TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Maschinenwesen

Robust and Efficient Discontinuous Galerkin Methods for Incompressible Flows

Niklas Fehn

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr.-Ing. Thomas Sattelmayer

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Wolfgang A. Wall
2. Prof. Dr. rer. nat. Gert Lube
   Georg-August-Universität Göttingen

Die Dissertation wurde am 18.03.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 31.08.2021 angenommen.

Abstract

The numerical solution of the incompressible Navier–Stokes equations requires sophisticated computational methods. Especially the accurate simulation of high-Reynolds-number turbulent flows is computationally demanding. This thesis contributes to the development of efficient numerical methods in computational fluid dynamics. A distinctive feature of this work is its holistic view on discretization methods, iterative solvers and preconditioners, and fast implementation techniques. The discontinuous Galerkin discretization of the incompressible Navier–Stokes equations uses mixed-order polynomials for velocity and pressure, upwind-like fluxes for the convective term, the symmetric interior penalty method for the viscous term, and central fluxes for the velocity–pressure coupling terms. A novel stabilized formulation is developed that enforces the divergence-free constraint and normal-continuity of the velocity between elements in a weak sense. By the use of consistent stabilization terms, robustness problems identified for standard L^2-conforming discretizations of the incompressible Navier–Stokes equations are overcome, which is attributed to improvements in terms of mass conservation, pressure-robustness, and energy stability. The discretization scheme is formally high-order accurate in space. More importantly, however, it exhibits appealing properties in terms of robustness and accuracy when applied to under-resolved turbulent flows, which are typical of engineering applications. This approach aims at combining the simplicity of L^2-conforming discretizations with the robustness and accuracy of H^{div}-conforming discretizations. This novel discontinuous Galerkin discretization is extended to coupled flow–transport problems and to problems on moving meshes. Moreover, applicability to fluid–structure interaction problems is demonstrated. For an efficient iterative solution of algebraic systems of equations, this work develops novel multigrid methods for high-order discontinuous Galerkin discretizations. The multigrid methods are hybrid in the sense that they exploit geometric and polynomial coarsening, an additional transfer to continuous finite element spaces, and the use of algebraic multigrid techniques as the coarse-level solver. To evaluate discretized differential operators and preconditioners efficiently, fast matrix-free evaluation techniques for tensor-product elements are used, which aim at optimal computational complexity by the use of the sum-factorization technique and optimal node-level performance by balancing arithmetic operations and data transfer. The combined efforts in terms of accurate discontinuous Galerkin discretizations, efficient Navier–Stokes time-stepping techniques, robust preconditioners, and efficient implementations allow substantial improvements over the state-of-the-art, which is demonstrated throughout the thesis. The efficiency of high-order discretizations is assessed critically in an error-vs-costs metric. By the example of the three-dimensional inviscid Taylor–Green problem, this work makes the interesting phenomenological observation that the temporal evolution of the kinetic energy seemingly converges to a dissipative solution under mesh refinement. These novel results contribute to key questions in fluid dynamics and turbulence research, and require careful discussions by the scientific community. Numerical evidence of anomalous energy dissipation in the inviscid limit or of the occurrence of finite-time singularities for the incompressible Euler equations according to Onsager's conjecture is unattained to date. The present work contributes to the development of incompressible flow solvers that exhibit the required robustness and efficiency to address these grand challenges in fluid dynamics on today's and next-generation's supercomputers. The software contributions of this work are made available to the scientific community.

Zusammenfassung

Die numerische Lösung der inkompressiblen Navier-Stokes-Gleichungen erfordert hochentwickelte Berechnungsmethoden. Insbesondere die genaue Simulation turbulenter Strömungen mit hohen Reynoldszahlen ist rechenintensiv. Die vorliegende Arbeit leistet einen Beitrag zur Entwicklung effizienter numerischer Methoden im Bereich der rechnergestützten Fluiddynamik. Ein charakteristisches Merkmal dieser Arbeit ist eine ganzheitliche Betrachtung von Diskretisierungsmethoden, iterativen Lösern und Vorkonditionierern sowie schnellen Implementierungstechniken. Die diskontinuierliche Galerkin-Diskretisierung der inkompressiblen Navier-Stokes-Gleichungen verwendet Polynome unterschiedlicher Ordnung für Geschwindigkeit und Druck, Aufwind-artige Verfahren für den konvektiven Term, die SIPG-Methode für den viskosen Term, und zentrale Flüsse für die Geschwindigkeits-Druck-Kopplungsterme. Eine neuartige stabilisierte Formulierung wird entwickelt, welche die Bedingung der Divergenzfreiheit sowie der Normalstetigkeit der Geschwindigkeit zwischen Elementen in einem schwachen Sinne aufbringt. Durch die Verwendung konsistenter Stabilisierungsterme werden Robustheitsprobleme überwunden, welche für gewöhnliche L^2-konforme Diskretisierungen beobachtet werden können, was sich auf Verbesserungen hinsichtlich Massenerhaltung, Druckrobustheit und Energiestabilität zurückführen lässt. Das Diskretisierungsschema ist formal von hoher Genauigkeitsordnung im Raum. Wichtiger erscheinen jedoch dessen ansprechende Eigenschaften hinsichtlich Robustheit und Genauigkeit bei Anwendung auf unteraufgelöste turbulente Strömungen, welche für Ingenieursanwendungen üblich sind. Dieser Ansatz verfolgt das Ziel, die Einfachheit von L^2-konformen Diskretisierungen mit der Robustheit und Genauigkeit von H^{div}-konformen Diskretisierungen zu verbinden. Diese neue diskontinuierliche Galerkin-Formulierung wird auf Strömungsprobleme mit gekoppelten Transportproblemen sowie auf Problemstellungen mit bewegten Rechengittern erweitert. Darüberhinaus wird eine Anwendung auf Probleme der Fluid-Struktur-Wechselwirkung gezeigt. Zur effizienten iterativen Lösung algebraischer Gleichungssysteme entwickelt die vorliegende Arbeit Mehrgitterverfahren für diskontinuierliche Galerkin-Diskretisierungen hoher Ordnung. Die Mehrgitterverfahren sind hybride Verfahren, da sie geometrische und polynomielle Vergröberung ausnutzen, einen zusätzlichen Transfer zu kontinuierlichen Funktionenräumen sowie algebraische Mehrgitterverfahren zur Lösung auf dem gröbsten Level verwenden. Zur effizienten Auswertung von diskretisierten Differentialoperatoren und Vorkonditionierern werden matrixfreie Auswertungstechniken für Elemente mit einer Tensorprodukt-Struktur angewendet, welche auf optimale Rechenkomplexität durch die Verwendung der Summenfaktorisierungstechnik abzielen sowie auf ein optimales Verhalten auf dem Level von Rechenknoten durch Algorithmen mit einem ausgewogenen Verhältnis von Rechenoperationen und Datentransfer. Die vereinten Bestrebungen nach genauen diskontinuierlichen Galerkin-Diskretisierungen, effizienten Zeitschrittverfahren für die Navier-Stokes-Gleichungen, robusten Vorkonditionierern und effizienten Implementierungen ermöglichen substanzielle Verbesserungen gegenüber dem aktuellen Stand der Technik, wie im Laufe der vorliegenden Arbeit gezeigt wird. Die Effizienz von Diskretisierungen hoher Ordnung wird hinsichtlich einer Genauigkeits-Kosten-Metrik kritisch hinterfragt. Am Beispiel des reibungsfreien Taylor-Green-Problems macht die vorliegende Arbeit die interessante phänomenologische Beobachtung, dass die zeitliche Entwicklung der kinetischen Energie bei Gitterverfeinerung scheinbar hin zu einer dissipativen Lösung konvergiert. Diese neuartigen Ergebnisse leisten einen Beitrag zu Kernfragen der Fluiddynamik und Turbulenzforschung und erfordern sorgfältige Diskussionen in der wissenschaft-

lichen Gemeinde. Ein numerischer Nachweis anomaler Energiedissipation im reibungsfreien Fall oder für das Auftreten von Singularitäten in endlicher Zeit bei den inkompressiblen Euler-Gleichungen gemäß der Hypothese nach Onsager ist bis heute unerreicht. Die vorliegende Arbeit trägt zur Entwicklung von inkompressiblen Strömungslösern bei, welche die nötige Robustheit und Effizienz aufweisen, um diese großen Herausforderungen der Fluiddynamik auf heutigen Supercomputern und denen der nächsten Generation anzugehen. Die Software-Beiträge der vorliegenden Arbeit werden der wissenschaftlichen Gemeinde zur Verfügung gestellt.

Danksagung

Zuerst möchte ich meinem Doktorvater, Wolfgang A. Wall, danken. Sie haben mir ermöglicht am Lehrstuhl für Numerische Mechanik zu promovieren und an einem spannenden Forschungsthema zu arbeiten. Ich danke Ihnen außerdem für die Freiheit, die Sie mir gegeben haben, und für das Vertrauen bei der Veröffentlichung zahlreicher wissenschaftlicher Artikel. Ich habe Sie kennengelernt als jemanden, der freie und ehrliche Wissenschaft lebt. Diese Überzeugungen haben mich während der Promotion motiviert, mich mit Energie der Wissenschaft zu widmen.

Ganz besonderer Dank gilt Martin Kronbichler, dem fachlichen Betreuer und Mentor meiner Doktorarbeit. Während meiner Masterarbeit konntest du mich für die Wissenschaft begeistern und hast mich zur Promotion am Lehrstuhl für Numerische Mechanik bewegt. Ich habe viel von dir gelernt und wir haben in den letzten Jahren viel Zeit gemeinsam verbringen und offen diskutieren können. Ich habe deine Rückendeckung, deinen Optimismus und deinen Pragmatismus sehr zu schätzen gelernt. Du hast auch fachlich in hohem Maße zum Erfolg dieser Promotion beigetragen, beispielsweise durch unzählige Softwareentwicklungen, sowie durch eine hilfreiche zweite Meinung und ein offenes Ohr.

Dank gilt auch all denjenigen, die dazu beigetragen haben, dass ich eine generöse technische Ausstattung vorgefunden habe und nutzen durfte, beispielsweise die Rechencluster am Lehrstuhl für Numerische Mechanik und vor allem die Großrechner des Leibniz-Rechenzentrums. Ohne diese Möglichkeiten wäre die vorliegende Doktorarbeit in dieser Form nicht möglich gewesen.

Ich danke allen Kollegen am Lehrstuhl für Numerische Mechanik für die zahlreichen schönen Stunden, etwa spannende Diskussionen beim Mittagessen und gemeinsame Zeit am Abend. Insbesondere möchte ich dem Admin-Team Danke sagen für die gegenseitige Unterstützung und konstruktive Zusammenarbeit. Ich danke ausdrücklich den beiden Kollegen Benjamin Krank und Peter Münch, mit denen ich fachlich eng zusammengearbeitet habe und die zum Gelingen dieser Arbeit beigetragen haben, sowie allen Studierenden, die ich betreuen durfte und die ebenfalls zur vorliegenden Arbeit beigetragen haben. Ebenso danke ich Julius Witte und Guido Kanschat aus Heidelberg, sowie Philipp Schröder, Christoph Lehrenfeld und Gert Lube aus Göttingen für die spannende Zusammenarbeit.

Zu guter Letzt möchte ich mich bei meiner Familie und meinen Freunden bedanken. Insbesondere möchte ich meinen Eltern und Großeltern meinen ganz herzlichen Dank aussprechen für die langjährige Unterstützung.

Garching, im September 2021 Niklas Fehn

Contents

Nomenclature

Notation – representation of scalars, vectors, tensors, and matrices

q, Q, λ, Λ	Scalar quantity
$\boldsymbol{q}, \boldsymbol{Q}, \boldsymbol{\lambda}, \boldsymbol{\Lambda}$	Vectorial quantity, second-order and higher-order tensors
$\boldsymbol{I}$	Identity tensor
$\boldsymbol{n}$ vs. $\boldsymbol{N}$	Spatial vs. material configuration
v	(Discrete) vector (linear algebra notation)
M	(Discrete) matrix (linear algebra notation)

Mathematics – operators and functions

$\partial(\bullet)/\partial t$	Partial time derivative
$\mathrm{d}(\bullet)/\mathrm{d}t$	Total time derivative
$\mathrm{D}(\bullet)/\mathrm{D}t$	Material time derivative
$\dot{(\bullet)}$	Time derivative (used if time t is the only variable)
$\partial(\bullet)/\partial x_i = (\bullet)_{,x_i}$	Partial derivative w.r.t. Cartesian coordinate direction x_i
$\nabla(\bullet) = \nabla_{\boldsymbol{x}}(\bullet)$	Gradient operator (w.r.t. spatial coordinates $\boldsymbol{x}$)
$\nabla_0(\bullet) = \nabla_{\boldsymbol{X}}(\bullet)$	Gradient operator w.r.t. material coordinates $\boldsymbol{X}$
$\nabla_{\boldsymbol{\chi}}(\bullet)$	Gradient operator w.r.t. mesh coordinates $\boldsymbol{\chi}$
$\nabla_{\boldsymbol{\xi}}(\bullet)$	Gradient operator w.r.t. reference coordinates $\boldsymbol{\xi}$
$\nabla \cdot (\bullet) = \mathrm{tr}(\nabla(\bullet))$	Divergence operator
$\nabla \times (\bullet)$	Curl operator
$\nabla^2(\bullet) = \nabla \cdot (\nabla(\bullet))$	Laplace operator
$\Delta(\bullet)$	Laplace operator
$\Delta(\bullet)$	Increment, difference to a reference state
$(\bullet) \odot (\bullet)$	Generic inner product
$(\bullet) \cdot (\bullet)$	Scalar product of two vectorial quantities, single tensor contraction ($\boldsymbol{a} \cdot \boldsymbol{b} = a_{ij} b_{jk} \boldsymbol{e}_i \otimes \boldsymbol{e}_k$)
$(\bullet) : (\bullet)$	double tensor contraction ($\boldsymbol{a} : \boldsymbol{b} = a_{ij} b_{ij}$)
$(\bullet) \otimes (\bullet)$	Tensor product
$(\bullet) \circ (\bullet)$	Function composition
$((\bullet), (\bullet))_{\{\cdot\}}$	L^2 inner product on domain, boundary, or interface specified by index $\{\cdot\}$
$\{\{\bullet\}\}$	Average operator
$[\![\bullet]\!]$	Jump operator
$[\bullet]$	Oriented jump operator
$\lvert \bullet \rvert$	Absolute value of a scalar quantity

$\| \bullet \| = \| \bullet \|_2$	Euclidean norm of a vector
$\| \bullet \|_\infty$	Infinity norm of a vector
$\overline{(\bullet)}$	Volume-averaged quantity
$\langle \bullet \rangle$	Ensemble average
$(\bullet)'$	Fluctuation
$(\bullet)^{-1}$	Inverse
$(\bullet)^{\mathsf{T}}$	Transpose
$(\bullet)^{-\mathsf{T}}$	Transpose of inverse operator
δ_{ij}	Kronecker δ
$\lambda(\bullet)$	Eigenvalue of a matrix
$\sigma(\bullet)$	Singular value of a matrix
const	Constant function
cos	Cosine function
det	Determinant
exp	Exponential function
lim	Limit
Lin	Linearization operator
log	Logarithm to base 10
min	Minimum function
max	Maximum function
rms	Root-mean-square value
sign	Sign function
sin	Sine function
tanh	Hyperbolic tangent function
tr	Trace operator

Physics – time

t	Physical time
T	Final time
$[0, T]$	Time interval

Physics – domains, boundaries, and coordinates

$\mathbb{R}^d$	Euclidean space in d space dimensions
Ω	domain in $\mathbb{R}^d$
Γ	boundary of domain in $\mathbb{R}^d$ ($\Gamma = \partial\Omega$)
Γ^{N}	Neumann boundary
Γ^{D}	Dirichlet boundary
d	Number of space dimensions (Euclidean space)
L	Characteristic length scale
a, A	Area
v, V	Volume

$k, \boldsymbol{k}$	Wavenumber and wavenumber vector
$\boldsymbol{n}, \boldsymbol{N}$	Outward pointing unit normal vector
$\boldsymbol{x} = (x_1, \ldots, x_d)^\mathsf{T}$	Cartesian coordinates in Euclidean space (Eulerian description)
$\boldsymbol{X} = (X_1, \ldots, X_d)^\mathsf{T}$	Material (Cartesian) coordinates in Euclidean space (Lagrangian description)
$\boldsymbol{\chi} = (\chi_1, \ldots, \chi_d)^\mathsf{T}$	Mesh (Cartesian) coordinates in Euclidean space (arbitrary Lagrangian–Eulerian description)

Physics – incompressible Navier–Stokes equations

$\boldsymbol{u} = (u_1, \ldots, u_d)^\mathsf{T}$	Velocity vector
p	Kinematic pressure
$\boldsymbol{f}$	Body force vector (force per unit mass)
$\boldsymbol{F}_\mathrm{c}$	Convective flux
$\boldsymbol{F}_\mathrm{v}$	Viscous flux
$\boldsymbol{g}_u$	Velocity Dirichlet boundary value
$\boldsymbol{h}$	Neumann boundary value
$\boldsymbol{t}$	Traction vector
$\boldsymbol{\omega}$	Vorticity vector
$\boldsymbol{\varepsilon}$	Symmetric part of velocity gradient
$\boldsymbol{\sigma}$	Cauchy stress tensor
ρ	Density
μ	Dynamic viscosity
ν	Kinematic viscosity
γ	Parameter distinguishing between divergence and Laplace formulation of viscous term
η	Kolmogorov length scale
ε	Dissipation rate
E	Kinetic energy
D	Kinetic energy dissipation rate
$\mathcal{E}$	Enstrophy
Re	Reynolds number

Physics – natural convection flows (Boussinesq approximation)

a	Thermal diffusivity
$\boldsymbol{g}$	Gravitational force
β	Thermal expansion coefficient
θ	Temperature
Gr	Grashof number
Pe	Péclet number
Pr	Prandtl number
Ra	Rayleigh number

Physics – non-linear structural mechanics

$\boldsymbol{d} = (d_1, \ldots, d_d)^\mathsf{T}$	Displacement vector
$\boldsymbol{v} = (v_1, \ldots, v_d)^\mathsf{T}$	Velocity vector
$\boldsymbol{a} = (a_1, \ldots, a_d)^\mathsf{T}$	Acceleration vector
$\boldsymbol{b}$	Body force vector (force per unit volume)
$\boldsymbol{g}$	Dirichlet boundary value
$\boldsymbol{t}$	Traction vector (Neumann boundary value)
$\boldsymbol{F}$	Deformation gradient
$\boldsymbol{E}$	Green–Lagrange strain tensor
$\boldsymbol{P}$	First Piola–Kirchhoff stress tensor
$\boldsymbol{S}$	Second Piola–Kirchhoff stress tensor
$\boldsymbol{C}$	Material tensor
$\boldsymbol{\varepsilon}$	Strain tensor
$\boldsymbol{\sigma}$	Cauchy stress tensor
Ψ	Strain energy function
ρ	Density
E	Young's modulus
ν	Poisson's ratio
λ, μ	Lamé coefficients

Numerics – temporal discretization

J	Order of time integration scheme
$\ell(t)$	Lagrange polynomial in time
n	Time step index
$N_{\Delta t}$	Total number of time steps
Δt	Time step size
γ_0, α_i	Coefficients of BDF time integration scheme
β_i	Coefficients of extrapolation scheme
$\alpha_\mathrm{f}, \alpha_\mathrm{m}, \beta, \gamma$	Coefficients of generalized-α time integration scheme
Cr	Courant number
r	Exponent of polynomial degree k in CFL condition
s	Exponent of mesh size h in explicit time step restriction
J_p	Order of extrapolation for projection-type schemes (pressure Neumann boundary condition in case of dual splitting scheme, pressure gradient term in case of incremental pressure-correction schemes)
$\hat{\boldsymbol{u}}, \hat{\hat{\boldsymbol{u}}}, \hat{\hat{\hat{\boldsymbol{u}}}}$	Intermediate velocity fields in projection-type schemes
ϕ	Pressure increment in pressure-correction scheme
χ	Parameter of pressure-correction scheme (standard formulation vs. rotational formulation)
$\boldsymbol{g}_{\hat{u}}$	Velocity Dirichlet boundary value for intermediate velocity in dual splitting scheme

$\boldsymbol{h}_u$	Velocity Neumann boundary value in projection-type schemes
g_p, g_ϕ	Pressure Dirichlet boundary value in projection-type schemes
h_p, h_ϕ	Pressure Neumann boundary value in projection-type schemes

Numerics – spatial discretization

$\mathcal{V}_h$	Space of test and trial functions
$L^2(\Omega)$	Sobolev space of square-integrable functions on Ω (discontinuous space)
$H^1(\Omega)$	Sobolev space of square-integrable functions with square-integrable first derivatives on Ω (continuous space)
$H^{\mathrm{div}}(\Omega)$	Sobolev space of $L^2(\Omega)$-functions for which their divergence is in $L^2(\Omega)$ (normal-continuous space)
Ω_h	Computational domain
$\Gamma_h = \partial\Omega_h$	Boundary of computational domain
Γ_h^{int}	Union of all interior faces between elements
$\boldsymbol{f}_{\mathrm{G}}(\boldsymbol{\chi}, t)$	Grid deformation function for time dependent domains (ALE formulation)
$\mathcal{I}_h$	Interpolation operator
Ω_e	Domain of element e
$\partial\Omega_e$	Boundary of element e
$\tilde{\Omega}_e = [0,1]^d$	Reference element (tensor product element)
$\mathcal{Q}_k(\tilde{\Omega}_e)$	Space of polynomials of tensor degree $\leq k$ on $\tilde{\Omega}_e$
$\mathbb{P}_k$	Space of polynomials of degree k in one space dimension
$\boldsymbol{\xi} = (\xi_1, \ldots, \xi_d)^{\mathsf{T}}$	Cartesian reference coordinates
$\ell^k_{i_1 \ldots i_d}(\boldsymbol{\xi})$	Multidimensional shape function of tensor degree k
$\ell^k_i(\xi)$	Lagrange polynomial of degree k
$\{\xi_j\}_{j=0}^k$	Set of nodal interpolation points defined on the unit interval $[0,1]$ in reference coordinates
$\boldsymbol{x}^e(\boldsymbol{\xi}), \boldsymbol{f}_{\mathrm{m}}(\boldsymbol{\xi})$	Mapping from reference space to physical space for element e
$\boldsymbol{x}^e_{i_1 \ldots i_d}$	Nodal coordinates of element e
$\boldsymbol{J}^e = \frac{\partial \boldsymbol{x}^e}{\partial \boldsymbol{\xi}}$	Jacobian
$\xi_q, \boldsymbol{\xi}_q$	Quadrature point in reference coordinates
w_q	Quadrature weight
e	Element index
f	Face index
h	Characteristic element length, mesh size
k	Polynomial degree of shape functions
k_{m}	Polynomial degree of mapping
l	Number of global mesh refinements
N_{DoFs}	Number of degrees of freedom
N_{el}	Number of elements

N_{faces}	Number of faces
$N_{\text{faces,int}}$	Number of interior faces
n_q	Number of one-dimensional quadrature points
$N_{q,e}$	Number of quadrature points per element
p	Polynomial degree of shape functions
q	Quadrature point index
$a^e_{\text{C},h}$	Continuity penalty operator
$a^e_{\text{D},h}$	Divergence penalty operator
b^e_h	Body force operator
c^e_h	Convective operator
d^e_h	Velocity divergence operator
g^e_h	Pressure gradient operator
l^e_h	Laplace operator
m^e_h	Mass matrix operator
v^e_h	Viscous operator
Λ	Stabilization parameter of Lax–Friedrichs flux
ζ_{LF}	Scaling factor of LF stabilization parameter
τ	Penalty parameter of SIPG method
ζ_{IP}	Scaling factor of interior penalty parameter
τ_{C}	Penalty parameter of continuity penalty term
ζ_{C}	Scaling factor of continuity penalty parameter
τ_{D}	Penalty parameter of divergence penalty term
ζ_{D}	Scaling factor of divergence penalty parameter

Numerics – algebraic systems of equations

$\mathbf{1}$	1-vector
$\mathbf{I}$	Identity matrix
$\mathbf{a}$	Vector of acceleration degrees of freedom
$\mathbf{a}_{\text{C}}$	Vector of continuity penalty operator
$\mathbf{A}_{\text{C}}$	Matrix representation of continuity penalty operator
$\mathbf{A}_{\text{D}}$	Matrix representation of divergence penalty operator
$\mathbf{A}_p$	Matrix of pressure convection–diffusion operator
$\mathbf{b}$	Body force vector
$\mathbf{c}$	Vector of non-linear convective operator
$\mathbf{C}_{\text{lin}}$	Matrix representation of linearized convective operator
$\mathbf{d}$	Vector of velocity divergence operator
$\mathbf{d}$	Vector of displacement degrees of freedom
$\mathbf{D}$	Matrix representation of homogeneous velocity divergence operator
$\mathbf{e}$	Vector of non-linear elasticity operator
$\mathbf{g}$	Vector of pressure gradient operator
$\mathbf{G}$	Matrix representation of homogeneous pressure gradient operator
$\mathbf{G}$	Velocity propagation matrix

$\mathbf{l}$	Vector of negative Laplace operator
$\mathbf{L}$	Matrix representation of negative Laplace operator
$\mathbf{M}$	Matrix representation of mass operator
$\mathbf{n}$	Null space vector
$\mathbf{p}$	Vector of pressure degrees of freedom
$\mathbf{P}$	Projector matrix
$\mathbf{r}$	Residual vector
$\mathbf{S}$	Schur complement matrix
$\mathbf{t}$	Traction vector
$\mathbf{u}$	Vector of velocity degrees of freedom
$\mathbf{v}$	Vector of velocity degrees of freedom
$\mathbf{v}$	Vector of viscous operator
$\mathbf{V}$	Matrix representation of homogeneous viscous operator
ϕ	Vector of degrees of freedom for pressure increment

Numerics – matrix-free implementation

$\mathbf{D}$	Differential operator
$\mathbf{I}$	Interpolation operator
$\mathbf{G}$	Gather operator
$\mathbf{S}$	Scatter operator

Numerics – iterative solvers, preconditioners, and multigrid

c	continuity parameter (DG vs. FE)
l	Index indicating current multigrid level
L	Number of multigrid levels (apart from coarse level)
n	Number of iterations
n_s	Number of smoothing steps
N	Problem size
$\mathbf{P}$	Preconditioner
$\mathbf{D}$	Diagonal matrix
$\mathbf{M}_e^l$	Elementwise mass matrix operator on level l
$\mathbf{M}_e^{l,l-1}$	Elementwise embedding operator from level $l-1$ to l
$\mathbf{P}^{l,l-1}$	Prolongation operator from level $l-1$ to l
$\mathbf{R}^{l-1,l}$	Restriction operator from level l to $l-1$
ε_abs	Absolute solver tolerance
ε_rel	Relative solver tolerance
ρ	Average convergence rate
τ, ω	Parameters of Newton solver
$\beta, \delta, \delta', \omega$	Parameters of conjugate gradient solver
σ_j, θ_j	Parameters of Chebyshev smoother

Numerics – fluid–structure interaction coupling

$\mathcal{F}$	Abstract notation of fluid solver
$\mathcal{M}$	Abstract notation of fluid mesh motion solver
$\mathcal{S}$	Abstract notation of structural mechanics solver
$\mathbf{d}$	Vector of degrees of freedom of solution
$\tilde{\mathbf{d}}$	Vector of degrees of freedom of preliminary solution
$\mathbf{f}_{\mathrm{FSI}}$	Operator representing one cycle of a partitioned FSI scheme
$\mathbf{r}$	Residual vector
$\mathbf{J}$	Jacobian
$\mathbf{D}$	Matrix composed of preliminary solution vectors $\tilde{\mathbf{d}}$
$\mathbf{R}$	Matrix composed of residual vectors $\mathbf{r}$
k	Iteration index
k_n	Number of iterations in time step n
m	Problem size of FSI system of equations
q	Number of reused time steps
ω_k	Relaxation parameter
$\boldsymbol{\alpha}_k$	Vector of k coefficients for solution update

Computational efficiency, performance engineering, parallel computations

$b_{\max}$	Maximum memory bandwidth
b_{c}	Code data transfer rate to/from main memory
c	Computational costs
E	Efficiency
I_{m}	Machine intensity (also Flop-to-Byte ratio)
I_{c}	Code intensity (also arithmetic/operational intensity)
N_{cores}	Number of cores
P	Number of processors (where processor denotes the smallest unit of a compute cluster, i.e., one node of a cluster)
P_{peak}	Peak floating point performance
P_{c}	Code floating point performance
S	Parallel speedup (strong scaling)
t_{wall}	Wall-clock time
η	Parallel efficiency

Symbols, subscripts, and superscripts

$\tilde{(\cdot)}$	Krylov projection
$\tilde{(\cdot)}$	Reference element
$\tilde{(\cdot)}$	Preliminary solution

$\hat{(\cdot)}$	Intermediate quantity (the same for multiple hats)
$\hat{(\cdot)}$	Approximation (used in the context of preconditioners)
$\hat{(\cdot)}$	Fourier transformation
$(\cdot)^-$	Interior
$(\cdot)^+$	Exterior
$(\cdot)^+$	Value in wall units
$(\cdot)^*$	Numerical flux function
$(\cdot)'$	Fluctuation
$(\cdot)^{\mathrm{D}}$	Dirichlet
$(\cdot)^{\mathrm{N}}$	Neumann
$(\cdot)^{\mathrm{I}}$	Interface
$(\cdot)^{\mathcal{F}}$	Fluid
$(\cdot)^{\mathcal{S}}$	Solid
$(\cdot)^{\mathcal{M}}$	Mesh (moving fluid mesh using ALE formulation)
$(\cdot)_0$	Indicates initial time $t = 0$ (initial/reference configuration)
$(\cdot)_{10}$	Refers to solver tolerance $\varepsilon_{10} = 10^{-10}$
$(\cdot)_{\mathrm{c}}$	Code (software)
$(\cdot)_{\mathrm{c}}$	Continuity equation
$(\cdot)_{\mathrm{c}}$	Convective term
$(\cdot)^{\mathrm{dst}}$	Destination
$(\cdot)_{\mathrm{DoFs}}$	Degrees of freedom
$(\cdot)_e$, $(\cdot)^e$	Element
$(\cdot)_{\mathrm{el}}$	Elements
$(\cdot)_{\mathrm{ex}}$	Explicit
$(\cdot)^{\mathrm{ex}}$	Extrapolation
$(\cdot)_f$	Face
$(\cdot)_{\mathrm{faces}}$	Faces
$(\cdot)_{\mathrm{FE}}$, $(\cdot)^{\mathrm{FE}}$	Continuous finite element space
$(\cdot)_{\mathrm{G}}$	Grid (ALE formulation)
$(\cdot)_h$	Indicates discrete version of a function (short for h, k)
$(\cdot)_{\mathrm{hom}}$	Homogeneous part of operator
$(\cdot)_{\mathrm{im}}$	Implicit
$(\cdot)_{\mathrm{inhom}}$	Inhomogeneous part of operator
$(\cdot)_{\mathrm{int}}$	Interior
$(\cdot)_k$, $(\cdot)^{(k)}$	Iteration index
$(\cdot)_l$, $(\cdot)^{(l)}$	Multigrid level
$(\cdot)_{\mathrm{lin}}$	Linearization
$(\cdot)_{\mathrm{m}}$	Machine (computer hardware)
$(\cdot)_{\mathrm{m}}$	Mapping
$(\cdot)_{\mathrm{m}}$	Momentum equation
$(\cdot)_{\mathrm{max}}$	Maximum value
$(\cdot)_{\mathrm{min}}$	Minimum value
$(\cdot)_n$, $(\cdot)^n$	Time step index
$(\cdot)_{\mathrm{opt}}$	Optimal value

$(\cdot)_p$, $(\cdot)^p$	Pressure
$(\cdot)_{\text{peak}}$	Peak performance (computer hardware)
$(\cdot)_q$	Quadrature point
$(\cdot)^{(q)}$	Indicates that q previous time steps are reused for FSI scheme
$(\cdot)_{\text{ref}}$	Reference
$(\cdot)_{\text{s}}$	Smoother
$(\cdot)^{\text{src}}$	Source
$(\cdot)_{\text{strong}}$	Strong formulation (integration-by-parts performed twice)
$(\cdot)_{\text{th}}$	Thermalization
$(\cdot)_u$, $(\cdot)^u$	Velocity
$(\cdot)_{\text{v}}$	Viscous term
$(\cdot)_{\text{w}}$	Wall (domain boundary)
$(\cdot)_{\text{wall}}$	Wall-clock time
$(\cdot)_{\text{weak}}$	Weak formulation (integration-by-parts performed once)
$(\cdot)_\phi$	Pressure increment
$(\cdot)_\tau$	Wall shear stress
$(\cdot)_\theta$	Temperature

Abbreviations

AC	Artificial Compressibility
ALDM	Adaptive Local Deconvolution Method
ALE	Arbitrary Lagrangian–Eulerian
AMG	Algebraic MultiGrid
AR	Aspect Ratio
ARM	Advanced RISC (Reduced Instruction Set Computer) Machines
AVM^4	Algebraic Variational Multiscale Multigrid Multifractal Method
AVX	Advanced Vector Extensions
BC	Boundary Condition
BR	Bassi–Rebay
BDF	Backward Differentiation Formula
BLAS	Basic Linear Algebra Subprograms
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy
CPU	Central Processing Unit
DBC	Dirichlet Boundary Condition
DFT	Discrete Fourier Transformation
DG	Discontinuous Galerkin
DGSEM	Discontinuous Galerkin Spectral Element Method
DNS	Direct Numerical Simulation
DoF	Degree Of Freedom
FEM	Finite Element Method
FDA	U.S. Food and Drug Administration

FMA	Fused Multiply Add
FSI	Fluid–Structure Interaction
GCL	Geometric Conservation Law
GMG	Geometric MultiGrid
GPU	Graphics Processing Unit
HDG	Hybridizable Discontinuous Galerkin
HPC	High-Performance Computing
ILU	Incomplete Lower Upper (decomposition)
IP	Interior Penalty
IQN-ILS	Interface Quasi-Newton Inverse Least-Squares
IQN-IMVJ	Interface Quasi-Netwon Inverse Multiple Vector update Jacobian
IQN-IMVLS	Interface Quasi-Newton Implicit Multi-Vector Least-Squares
K41	Kolmogorov's 1941 theory of turbulence
LDG	Local Discontinuous Galerkin
LES	Large-Eddy Simulation
LF	Lax–Friedrichs
LGL	Legendre–Gauss–Lobatto
MG	MultiGrid
MPI	Message Passing Interface
MRI	Magnetic Resonance Imaging
NBC	Neumann Boundary Condition
PDE	Partial Differential Equation
RANS	Reynolds-averaged Navier–Stokes
RT	Raviart–Thomas
SEM	Spectral Element Method
SIMD	Single Instruction Multiple Data
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SIPG	Symmetric Interior Penalty Galerkin
SSE	Streaming SIMD Extensions
SUPG	Streamline Upwind Petrov–Galerkin
SVD	Singular Value Decomposition
SVV	Spectral Vanishing Viscosity
TGV	Taylor–Green Vortex
WALE	Wall-Adapting Local Eddy-viscosity
WENO	Weighted Essentially Non-Oscillatory

1 Introduction

Fluid dynamics is ubiquitous. Classical engineering disciplines such as transportation, power plants, and energy convertion rely in a very fundamental way on fluid dynamics. In terms of transportation, fluid dynamics often has an adverse effect due to drag forces that need to be minimized for optimal usage of energy (trains and cars), but is the key element or enabling mechanism in other cases e.g. by providing the required lift or repulsion (airplanes and ships). In terms of power plants and green technologies, fluid dynamics is key to hydropower and wind energy, two of the main sources of regenerable energy. Also conventional forms of energy convertion such as chemical energy transformed into mechanical energy make use of principles of fluid dynamics. From the level of workstations to the level of supercomputers, the cooling of microprocessors is also a fluid dynamical problem. While the above examples are all technologies created by humans, there are fluid dynamical problems not directly influenced by mankind. These include phenomena in geodynamics such as convection in Earth's mantle, catastrophic events such as tsunamis and hurricanes, or just daily weather phenomena. Viruses are transmitted between human beings via aerosoles. The flow of air and transport of oxygen through the human lung or the flow of blood through the heart or veins and arteries are fluid dynamical problems. Finally, a multitude of sports fascinating human beings would be inconceivable without fluid dynamics, e.g., sailing, formula one racing, sky diving, gliding, or ski jumping.

1.1 Motivation

An appropriate physical model for the fluid flow problems mentioned above can be derived by a continuum description. Mathematically, this description leads to a set of partial differential equations, the Navier–Stokes equations. In most of the above examples, the fluid flow can be characterized as incompressible, i.e., the density does not change noticeably along the trajectory of a fluid element. The Navier–Stokes equations fascinate scientists from various disciplines, as exemplified by the Clay Millenium problem on the Navier–Stokes equations.[1] While the notion of turbulent flows is intuitively understandable, turbulence is at the same time one of the big unsolved problems. To exemplarily raise one question subject to controversial debates in physics, mathematics, and numerics: In the theoretical limit of vanishing viscosity, do turbulent flows still dissipate energy?

The discipline addressing the understanding of fluid dynamical problems by numerical techniques is called computational fluid dynamics (CFD). Despite immense progress in this field over the last decades, CFD simulations are still not feasible for many engineering applications even on today's supercomputers. A pressing issue in this field are scale-resolving simulations of high-Reynolds-number turbulent flows, i.e., being able to perform simulations of turbulent flows

[1] https://www.claymath.org/millennium-problems/

with billions to trillions of unknowns and millions of time steps in reasonable wall-time limits. Note that CFD software has somewhat fallen behind expectations dating back to the early times of large-eddy simulation, in the sense that LES is still not feasible as a standard design tool in an industrial context in the year of 2020, despite the ever-increasing power of computers. From such a perspective, a strong need for computationally efficient CFD techniques arises.

In retrospect, one might argue that computational fluid dynamics has often not been the technology paving the way. Mankind has been very successful in developing new technologies without a complete understanding of the underlying physical mechanisms. As an impressive example, airplanes were flying long before the first computer was built. However, from an engineering perspective, the common element of the applications mentioned in the very beginning is the optimization of a system according to a target function. This target function can be the efficiency of power plants, the lifetime of technical products, or the mortality rate or expectancy of life in the biomedical sector. This is exactly the perspective from which a main motivation for CFD originates. Often, flows could also be investigated by experimental techniques such as measurements in a wind tunnel. However, many applications are not amenable to measurements, because the scales of the flow might be too large or too small. Experimental setups might not be possible because certain parameters are not realizable, or because living objects are involved. Finally, an optimization loop might be easier to realize in a purely digital or virtual environment. Then, computational techniques might be the only viable approach to predict a system's behavior.

The development of computational methods enabling the efficient numerical solution of the incompressible Navier–Stokes equations and, thereby, the prediction and understanding of fluid dynamical problems is at the heart of the present thesis. To exemplify extensibility of the methodology developed in this thesis to multiphysics problems, coupled flow–transport problems and fluid–structure interaction are also covered by the present thesis. The broad applicability of the methods developed here is demonstrated by applications motivated from biomechanical problems such as simulating the flow of air through the human lung during the mechanical ventilation of neonates, to geodynamical problems such as earth mantle convection.

1.2 Background and embedding into priority programme for exa-scale computing

The above motivation has used the notion of *efficient* numerical methods, but what characterizes an efficient method in the context of PDE solvers? To explain this term, one first needs to internalize that the outcome of numerical simulations is accompanied by errors, in the very general sense of models representing an approximation of reality. The main source of errors addressed in the present work are discretization errors in space and time. Errors due to finite tolerances of iterative solvers or numerical round-off are also present, but they are typically negligible for the kind of problems discussed here. Another important class of errors emerges from modeling and associated uncertainties, which is also not the primary subject of the present work. Assuming that errors can be quantified, an accurate simulation is one that has a low error, i.e., a simulation for which the deviation from reality (such as experimental measurements, which are themselves subject to uncertainties and errors) is small. There is certainly a desire for accurate methods, and it is natural to ask for the price to pay for. Based on this line of argumentation, one may define

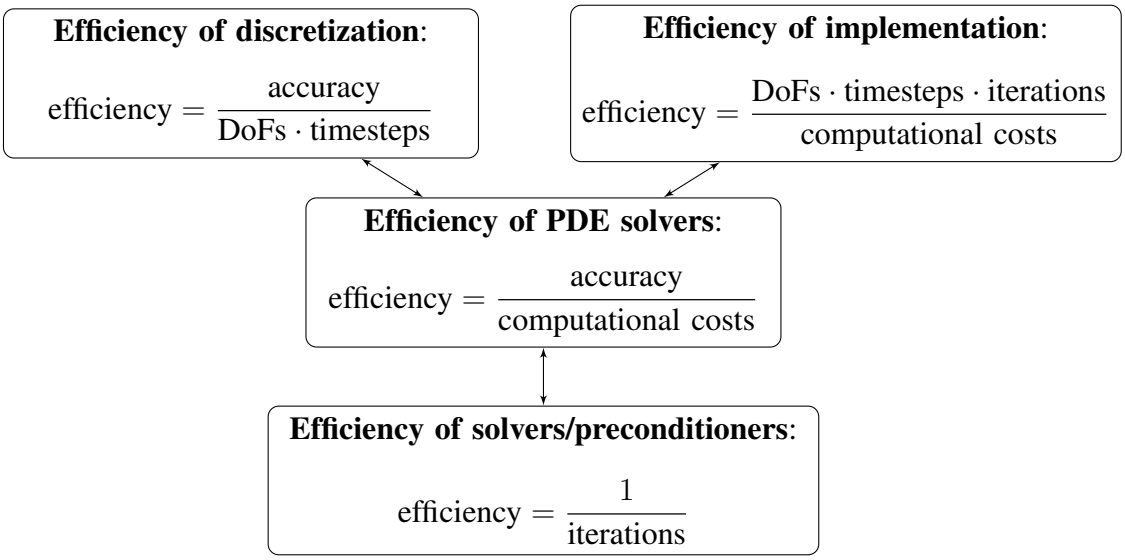


Figure 1.1: Discretization, iterative solvers/preconditioners, and implementation as the three main (multiplicative) contributions to the overall efficiency of PDE solvers.

efficiency as a combination of accuracy and computational costs. The notion of convergence of a numerical method is a well accepted metric for a method to qualify as reliable. Such a convergent method increases in accuracy when more computational effort is invested. While convergence is related to the mathematical notion of a limit process, the notion of efficiency contains an engineering component in the sense that resources are limited, that optimal accuracy is to be realized with a minimal or limited amount of computational costs.

In the context of PDE solvers for problems typically arising in natural sciences and modeled by continuum mechanics, three main contributors to efficiency can be identified according to Figure 1.1. This efficiency model assumes unsteady problems tackled by a method-of-lines discretization approach, where systems of equations arising in each time step are solved by iterative techniques (typically Krylov methods with some robust preconditioners such as multigrid to obtain mesh-independent convergence rates), for which the evaluation of discretized PDE operators forms a main algorithmic and performance-relevant ingredient of the PDE solver. The multiplicative splitting of the overall efficiency into three main contributions originates from the nested-loop character of CFD programs with an outer loop over all time steps, a loop over all iterations of iterative solvers, and an innermost loop over all elements of a mesh on which discretized versions of PDE operators need to be evaluated. The following discussion mainly refers to finite element and discontinuous Galerkin discretization techniques, for which the two main discretization parameters are the mesh size h and the polynomial degree k describing the space of polynomial shape functions on each element.

The separation of disciplines according to Figure 1.1 is well-accepted and underlines the multidisciplinary nature of PDE solver development (and computational fluid dynamics in particular). The point to make here is that these topics are, however, highly interconnected, requiring a holistic view to achieve optimal efficiency. From a software perspective, this separation of concerns is reflected in the common choice of using matrix-based iterative solvers that can be applied in a black-box fashion to the system of equations arising from the discretization of a PDE.

The price to pay for this high degree of generality is a rather poor performance when applied to high-order discretization methods, that are – under these circumstances – typically considered inefficient and to not pay off in terms of efficiency. Elementwise matrices are dense, leading to an at least quadratic complexity in terms of degrees of freedom per element and rendering high-order methods prohibitively expensive with this implementation technique. A methodology that tries to overcome these performance limitations of high-order methods is based on the idea of eliminating interior degrees of freedom prior to the solution of algebraic systems of equations (static condensation technique or hybridizable discontinuous Galerkin methods), a technique which aims at preserving a high degree of generality in terms of clearly separating the three disciplines from a software perspective and maintaining black-box interfaces to linear algebra packages for the solution of algebraic systems of equations. The present work argues that this optimization level is not "aggressive" enough to achieve optimal performance, especially for three-dimensional problems. The term *optimal* refers to the question whether e.g. high-order discontinuous Galerkin methods can be implemented as efficiently as their low-order counterparts or finite-difference methods with stencil-like matrix-free implementation. To achieve this goal, matrix-free implementation techniques are required, which can be realized most efficiently for element types that exhibit a tensor-product structure such as hexahedral elements in three space dimensions (Kronbichler and Kormann 2019), even though these techniques can also be realized for simplicial elements at somewhat reduced performance (Moxey et al. 2020a).

The reason why these different implementation techniques vary significantly in computational efficiency is that they exhibit different characteristics in terms of floating point operations, the amount of data transferred from main memory, and arithmetic intensity (the ratio of floating point operations to memory transfer). It should be emphasized that recent developments in computer hardware towards multi-core architectures with increasing SIMD capabilities and arithmetic performance increasing at a higher speed than memory bandwidth impact the question regarding an optimal implementation strategy. This question is further complicated by an increasing heterogeneity in the hardware landscape leading to different implementation techniques being most efficient on different hardware platforms. As shown in the course of this thesis, these design choices are not about rendering a method twice as fast, but essentially have an orders-of-magnitude impact on computational costs for high-order methods. Note that fast matrix-free implementation techniques also introduce several challenges. From the perspective of software design, the challenge is to preserve generality and a separation-of-concerns. Another main challenge is the development of matrix-free preconditioners, which is as important as matrix-free operator evaluation in order to realize fast PDE solvers for high-order discretizations.

This PhD project has been funded by the German Research Foundation (DFG) through the project *ExaDG – High-order Discontinuous Galerkin for the Exa-scale* (see Arndt et al. (2020b) for a summary), which has been part of the second phase of the German priority programme for Exa-scale[2] computing (SPPEXA[3] – Software for Exascale Computing). The main goals of the project ExaDG are novel contributions in terms of (i) robust and accurate discretization methods for PDE model problems arising in computational fluid dynamics with an emphasis on high-order discontinuous Galerkin discretization techniques, (ii) the development of fast matrix-free

[2] An Exa-scale computer is one that is able to perform 10^{18} floating-point operations per second. Currently, the world's largest supercomputer is Fugaku installed in Japan in the year of 2020 with a theoretical peak performance of approximately 0.5 EFlop/s, see https://www.top500.org/.

[3] See http://www.sppexa.de/ for details.

implementation techniques for high-order continuous and discontinuous Galerkin discretizations with a focus on node-level performance (i.e., maximizing the throughput of matrix-free operator evaluation or minimizing time-to-solution), and (iii) the development of robust and fast matrix-free preconditioners and iterative solvers for such high-order discretization techniques, see also Figure 1.1. The motivation for the use of discontinuous Galerkin methods originates primarily from the geometric flexibility and the sound mathematical foundation of finite element methods on the one hand, and advantageous stability properties for problems with dominant convection by adapting concepts from finite volume discretizations on the other hand. Discontinuous Galerkin methods are often motivated from the point of view of parallel scalability, but the grand challenge and question of scientific interest appears to be that of achieving good single-core or node-level performance as explained above. Regarding the latter aspect, high-order methods are often motivated with a higher arithmetic intensity. However, the absolute number of floating point operations is most often also higher per degree of freedom, i.e., implementations of high-order discretizations might achieve a significant fraction of a computer's peak performance, but might exhibit sub-optimal performance in terms of time-to-solution. In agreement with Figure 1.1, floating point performance can be considered a metric of subordinate importance. Instead, the metric of engineering interest is time-to-solution or computational costs.

1.3 Novel contributions of this work

The contributions of this work are twofold. They include software developments on the one hand, and new method developments in the field of discontinuous Galerkin methods for incompressible flows and natural convection flows, ALE methods and fluid–structure interaction, and fast matrix-free preconditioners and multigrid methods for high-order DG discretizations on the other hand. These developments have enabled the numerical investigation of several questions of scientific interest, for example the efficiency of high-order DG discretizations when applied to under-resolved turbulent flows by performing detailed mesh refinement studies for the viscous Taylor–Green problem, or the phenomenon of anomalous energy dissipation in incompressible Euler flows by performing high-resolution simulations of the challenging inviscid Taylor–Green problem. As demonstrated at several places in this thesis, these contributions have lead to significant performance improvements over many state-of-the-art methods and implementations in this field.

In the course of this thesis, a new CFD software has been developed, whose main focus is on efficient discontinuous Galerkin methods for incompressible flow problems, given that an incompressible flow solver is often a core module of CFD software. In particular, the software addresses scale-resolving simulations of high-Reynolds-number turbulent flows as a main challenge in CFD. The software project is called `ExaDG` and builds upon the generic finite element library `deal.II` (Arndt et al. 2020a, Bangerth et al. 2007). Both software projects are written in `C++` and make use of the object-oriented programming paradigm. The separation-of-concerns design principle allows `ExaDG` to concentrate on the physics, i.e., the project's credo is that fluid dynamics experts without a strong background in computer science should be able to write high-performance CFD software. The software `ExaDG` implements the discretized spatial derivative operators, numerical fluxes, boundary conditions, time integration schemes, and postprocessing routines for a particular PDE model problem at hand, while `deal.II` provides core finite ele-

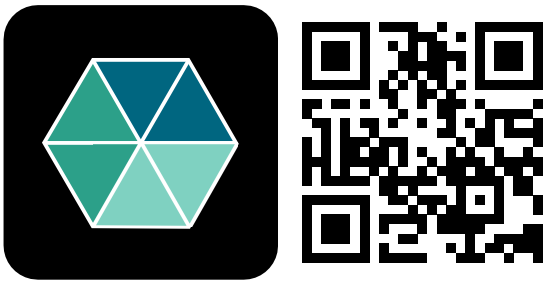


Figure 1.2: Code project `ExaDG` with icon (left) and QR code for https://github.com/exadg project (right). The project aims at fast PDE solvers by using matrix-free implementation techniques for tensor-product elements, as symbolized by the hexahedral element in the icon. Alternatively, the icon shows a triangular mesh that symbolizes the project's extensibility towards simplicial meshes as part of future work.

ment functionalities such as meshing and parallel domain decomposition, basic FE ingredients such as finite elements and shape functions, mappings, quadrature rules, the handling of degrees of freedom, Krylov solvers, and – most importantly in the context of the present thesis – integrators for the matrix-free evaluation of PDE operators. Regarding the latter aspect, the matrix-free implementation in `deal.II` provides high-performance kernels for the interpolation of shape functions and their derivatives into quadrature points, organizes the loops over the elements and faces of a mesh, and takes care of the parallelization in the form of a generic interface for matrix-free operator evaluation (Kronbichler and Kormann 2012). The library `deal.II` itself makes use of sophisticated third-party libraries. The `ExaDG` software is available on `github` (see Figure 1.2) to allow verification and reproducibility of results obtained in the present thesis, and to fulfill SPPEXA's mission of making software contributions for Exa-scale computing available to the scientific community.

The following publications (listed in chronological order) have been published (or submitted/accepted for publication) prior to the submission of this thesis:

(i) B. Krank, N. Fehn, W. A. Wall, and M. Kronbichler, A high-order semi-explicit discontinuous Galerkin solver for 3D incompressible flow with application to DNS and LES of turbulent channel flow, *Journal of Computational Physics* **348**, 634–659, 2017.

(ii) N. Fehn, W. A. Wall, and M. Kronbichler, On the stability of projection methods for the incompressible Navier–Stokes equations based on high-order discontinuous Galerkin discretizations, *Journal of Computational Physics* **351**, 392–421, 2017.

(iii) N. Fehn, W. A. Wall, and M. Kronbichler, Robust and efficient discontinuous Galerkin methods for under-resolved turbulent incompressible flows, *Journal of Computational Physics* **372**, 667–693, 2018.

(iv) N. Fehn, W. A. Wall, and M. Kronbichler, Efficiency of high-performance discontinuous Galerkin spectral element methods for under-resolved turbulent incompressible flows, *International Journal for Numerical Methods in Fluids* **88**, 32–54, 2018.

(v) N. Fehn, W. A. Wall, and M. Kronbichler, A matrix-free high-order discontinuous Galerkin compressible Navier–Stokes solver: A performance comparison of compressible and incompressible formulations for turbulent incompressible flows, *International Journal for Numerical Methods in Fluids* **89**, 71–102, 2019.

(vi) N. Fehn, M. Kronbichler, C. Lehrenfeld, G. Lube, and P. W. Schroeder, High-order DG solvers for under-resolved turbulent incompressible flows: A comparison of L^2 and $H(\text{div})$ methods, *International Journal for Numerical Methods in Fluids* **91**, 533–556, 2019.

(vii) N. Fehn, W. A. Wall, and M. Kronbichler, Modern discontinuous Galerkin methods for the simulation of transitional and turbulent flows in biomedical engineering: A comprehensive LES study of the FDA benchmark nozzle model, *International Journal for Numerical Methods in Biomedical Engineering* **35**, e3228, 2019.

(viii) N. Fehn, P. Munch, W. A. Wall, and M. Kronbichler, Hybrid multigrid methods for high-order discontinuous Galerkin discretizations, *Journal of Computational Physics* **415**, 109538, 2020.

(ix) D. Arndt, N. Fehn, G. Kanschat, K. Kormann, M. Kronbichler, P. Munch, W. A. Wall, and J. Witte, ExaDG: High-Order Discontinuous Galerkin for the Exa-Scale, In H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, and W. E. Nagel (eds.), *Software for Exascale Computing - SPPEXA 2016-2019*, pages 189–224, Cham, 2020, Springer International Publishing.

(x) N. Fehn, M. Kronbichler, P. Munch, and W. A. Wall, Numerical evidence of anomalous energy dissipation in incompressible Euler flows: Towards grid-converged results for the inviscid Taylor–Green problem, *Journal of Fluid Mechanics* **accepted**, arXiv preprint arXiv:2007.01656, 2021.

(xi) N. Fehn, J. Heinz, W. A. Wall, and M. Kronbichler, High-order arbitrary Lagrangian–Eulerian discontinuous Galerkin methods for the incompressible Navier–Stokes equations, *Journal of Computational Physics* **430**, 110040, 2021.

These articles constitute a main contribution of this work and content from many of these publications is reproduced in this thesis as summarized in Table 1.1. For clarity, the inverse map (chapter $\rightarrow$ publications) is additionally highlighted at the beginning of each chapter, to indicate which publications a particular chapter (or a part of a chapter) is based on. Chapters 3 and 9 present entirely new content that has not been published or submitted for publication prior to the submission of this thesis. For reasons of brevity, the content of the works (i), (v), (vi), (vii) is not presented in the present thesis. These works complement topics discussed in this thesis and are suggested as further reading material depending on the reader's interest. To highlight contributions by other PhD students, the work (i) has been developed in close collaboration with Benjamin Krank, PhD student at the Technical University of Munich, the work (vi) in close collaboration with Philipp Schroeder, PhD student at the Georg-August-Universität Göttingen, and the work (viii) in close collaboration with Peter Munch, PhD student at the Technical University of Munich. The work (ix) constitutes the final project report of the SPPEXA-ExaDG project, to which Julius Witte contributed as a PhD student at Heidelberg University.

Table 1.1: List of publications with references to chapters of this thesis that discuss or reproduce content of a given publication. The symbol '-' means that the content of a publication is not subject of the present thesis.

Publication	(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)	(x)	(xi)
Chapter(s)	-	2	2	4, 5, 6	-	-	-	4, 5	6	7	8

The contributions of these publications are as follows: The works (i), (ii), (iii) develop discontinuous Galerkin discretization methods with the goal to obtain robust incompressible flow solvers, overcoming several stability problems reported in previous works. The work (i) reviews state-of-the-art techniques from the literature and proposes remedies to address instabilities occurring for small time step sizes and for under-resolved problems typically encountered when simulating turbulent flows. The subsequent work (ii) clarifies the role of the velocity–pressure coupling terms and their DG discretization in the context of instabilities occurring for small time step sizes, and the work (iii) the role of consistent divergence and continuity penalty terms in terms of mass conservation and energy stability (and their stabilizing effect for under-resolved turbulent flows). To the best of the author's knowledge, the works (i) and (iii) represent the first successful application of L^2-conforming DG discretizations to the accurate simulation of turbulent incompressible flows with a robust behavior in under-resolved scenarios. The stabilized L^2-conforming DG discretization for incompressible flows discussed in this thesis exhibits many parallels to exactly divergence-free H^{div}-conforming discretizations. This topic has been explored in detail in the work (vi), where both approaches are compared regarding their accuracy and suitability to simulate turbulent flow problems in an implicit LES context without explicit sub-grid models. This study suggests that both discretization schemes are interesting candidates for generic turbulent flow solvers, and fosters a perspective where LES modeling might be understood as the problem of devising numerical discretization schemes with suitable mathematical properties. The work (vii) investigates the capabilities of the present incompressible DG solver to simulate transitional and turbulent flow problems by the example of the FDA benchmark nozzle problem, a flow configuration typically occurring in biomedical engineering. The study concludes that a sensitive behavior related to the definition of the FDA benchmark problem complicates a rigorous quantification of the accuracy and computational efficiency of CFD methods for the benchmark problem under investigation. An extension of the stabilized DG discretization for incompressible flows towards moving meshes based on the arbitrary Lagrangian–Eulerian technique is presented in the work (xi), which contributes to the state-of-the-art in the sense that an emphasis is put on temporal consistency for projection-type solvers as well as on aspects of energy stability and fulfillment of the geometric conservation law.

The work (iv) details the efficiency of this high-order DG discretization when applied to turbulent flows by the example of the three-dimensional Taylor–Green benchmark problem. This work benchmarks the present incompressible Navier–Stokes DG solver in terms of node-level performance and achieves a significant speed-up over state-of-the-art methods. This work also proposes a simple efficiency model for high-order PDE solvers and the results shown in this work suggest that high-order PDE solvers have to strive for optimal algorithmic complexity w.r.t. the

polynomial degree of the shape functions in order to render high-order methods more efficient in under-resolved application scenarios. The work (v) presents a high-order DG discretization for compressible flows using fast matrix-free implementation techniques. It addresses the question whether compressible solvers amenable to explicit time stepping pose a computationally efficient alternative to incompressible formulations when applied in a low Mach number regime. The incompressible DG solver is found to be significantly more efficient in terms of node-level performance than the explicit compressible DG solver, where the novel implementation of the compressible DG solver has been shown to outperform a state-of-the-art implementation from the literature substantially. The work (viii) develops novel multigrid techniques for high-order discontinuous Galerkin discretizations that exploit all levels of geometric, polynomial, and algebraic coarsening, which is particularly relevant to solve problems with complicated geometries in a computationally efficient way. In such a regime, pure h-multigrid methods typically show a rather poor performance for high polynomial degrees if not the number of mesh refinement level is large. The work (ix) investigates the parallel scalability of the present incompressible DG solver on the SuperMUC-NG supercomputer with more than 300k cores.

Finally, the study (x) builds upon the developments made in many previous contributions and addresses the challenging three-dimensional inviscid Taylor–Green vortex problem. An interesting result of this study is the phenomenological observation that the kinetic energy evolution seemingly converges to a dissipative solution with non-zero dissipation rate for increasing spatial resolution of the DG discretization scheme. The study raises the questions to which extent these results are related to weak dissipative solutions of the incompressible Euler equations and to which extent these results can be interpreted as a numerical confirmation of anomalous energy dissipation. An interesting idea is that observing such a dissipative behavior in numerical experiments might give indirect hints of finite-time singularities in incompressible Euler flows according to Onsager's conjecture. A main contribution to the state-of-the-art is the perspective elaborated in this work that the use of energy-conserving schemes which are widely used for the simulation of incompressible Euler flows might not be suitable to address the question regarding the occurrence of anomalous energy dissipation. As a conclusion, discretization methods such as those developed in this thesis might constitute a new tool in turbulence research and enable the numerical investigation of new classes of problems.

Apart from the contributions listed above, the author of this thesis has been involved in other projects that have led to the following publication, to which the author contributed partly:

- Kronbichler et al. (2018b)
- Kronbichler et al. (2019)
- Nitzler et al. (2020)

The work by Kronbichler et al. (2018b) summarizes early contributions in the field of stabilized DG methods for the simulation of incompressible turbulent flows and briefly outlines the algorithmic framework in terms of fast matrix-free implementation techniques. The work by Kronbichler et al. (2019) proposes an innovative Hermite-like polynomial basis for DG discretizations of PDE operators with second derivatives, e.g., the constant coefficient Poisson equation with symmetric interior penalty discretization. The use of a Hermite-like basis allows to improve the data access pattern of face integrals for discontinuous Galerkin discretizations with matrix-free implementation, and, thereby, results in an increased computational efficiency. The work

by Nitzler et al. (2020) addresses aspects of uncertainty propagation and highlights the need to simulate the forward problem efficiently. In this context, the incompressible Navier–Stokes DG solver developed in the present thesis is used.

New developments related to active scalar transport (natural convection flows) and fluid–structure interaction are presented in this thesis that have not been published prior to the submission of this thesis. By the example of natural convection flows, this thesis shows that the formulation of the convective term for active scalar transport plays a decisive role in terms of the stability of the discretization scheme for coupled flow–transport problems and that instabilities might occur in case that the velocity field is not exactly mass-conserving. An interesting aspect is that pressure-robust discretizations for the incompressible Navier–Stokes equations might also be prone to this instability since pressure-robustness does not imply that the velocity field is exactly mass-conserving. To preserve stability, the flow–transport coupling needs to fulfill a compatibility condition. In practice, this compatibility condition implies that the transport term should be written in convective formulation in general. With respect to fluid–structure interaction problems, this thesis proposes a new FSI solver based on an ALE-DG formulation for the fluid problem and a standard H^1-conforming discretization for the solid problem. The formulation is flexible w.r.t. non-matching grids at the fluid–structure interface. Moreover, the polynomial degrees used for the numerical approximation of the different fields can be chosen independently. First numerical results shown for this prototype FSI solver give indications that the paradigm of partitioned FSI algorithms with fast matrix-free single-field solvers might open new doors towards a next generation of fluid–structure interaction solvers, potentially overcoming performance limitations of state-of-the-art monolithic FSI algorithms.

1.4 Scope of this work

For reasons of brevity, this thesis does not give an introduction to fluid dynamics, turbulence phenomena, discretization methods in general, and finite element or discontinuous Galerkin techniques in particular. All of this is well-documented in the literature. The present thesis does also not cover the topic of wall-modeling relevant for the numerical solution of high-Reynolds-number wall-bounded turbulent flows, see for example the thesis by Krank (2019) in the context of discontinuous Galerkin discretization methods discussed here. Computational fluid dynamics has always been an interdisciplinary topic, requiring expertise from mathematics, physics, numerics, engineering, and computer science. This thesis is intended to catch readers from all of these disciplines. The red threat of this work is certainly the engineering effort striving for efficient PDE solvers in computational fluid dynamics, with *efficient* in the sense of robust, accurate, and fast numerical methods. This thesis might be particularly interesting for physicists and engineers sharing the desire for fluid dynamics solvers of the next generation enabling the efficient simulation of turbulent flows through novel discretization and implementation techniques. It might also be particularly relevant for computer scientists interested in the application field of PDE solvers. Finally, for those engaged in the emerging fields of machine learning, uncertainty propagation, optimization, etc. applied to PDE-based model problems, this work might be interesting in the sense that the speed at which the forward solution of a problem can be performed is often a main limiting factor.

1.5 Outline

This thesis is organized as follows: Chapter 2 discusses discretization methods for the incompressible Navier–Stokes equations, with a focus on efficient and accurate discretization techniques in both space and time. With respect to the temporal component, coupled (or monolithic) as well as projection-type solution techniques are discussed that are flexible in terms of fully-implicit or semi-implicit formulations and adaptive time stepping. With respect to the spatial component, high-order discontinuous Galerkin methods with suitable stabilization techniques are developed. Extensive numerical investigations detail the discretization properties of the proposed incompressible Navier–Stokes solvers. Here, a focus is on the robust and accurate simulation of turbulent flow problems. Chapter 3 provides an extension to coupled flow–transport problems. While this chapter covers both active and passive scalar transport, the focus of this chapter is on natural convection flows since this is the setting that is more challenging from a discretization point of view. Incompressible flow and scalar transport solvers are core modules of every CFD software, so that Chapters 2 and 3 are recommended as reading material for those readers interested in working with the `ExaDG` software or in learning more about its theoretical background in terms of discretization methods applied "under the hood".

The next three chapters shed light on the computational efficiency of this high-order discontinuous Galerkin discretization framework. Chapter 4 gives an introduction to fast matrix-free evaluation techniques as a general technique to evaluate discretized PDE operators in an abstract finite element context. Numerical results illustrate the performance of the main operators forming the kernels of high-order discontinuous Galerkin discretizations of the incompressible Navier–Stokes equations. Chapter 5 addresses iterative solvers and preconditioners. A main emphasis is put on efficient multigrid preconditioners for a Poisson-type model problem that is the key to a fast incompressible flow solver. In the context of high-order DG methods, the methodology of hybrid multigrid techniques exploiting polynomial, geometric, and algebraic coarsening strategies within the multigrid hierarchy is of particular importance and is explained in detail. Moreover, block-preconditioners for the saddle-point problem associated to coupled incompressible flow solvers are discussed. Chapter 6 takes an application perspective, with the goal to obtain accurate solutions of incompressible flow problems with a minimal amount of computational costs. This chapter also investigates whether high-order discretization methods can be expected to pay off in terms of computational efficiency. For practitioners and engineers in CFD, this might be the chapter that is most relevant since it benchmarks high-order DG methods in an error-vs-cost metric and might allow extrapolations whether this approach is worth giving it a try in an application context.

The remaining chapters cover advanced topics. Chapter 7 raises the question whether the proposed numerical discretization technique might be suitable to address a grand challenge in fluid dynamics and turbulence research, namely providing evidence of anomalous energy dissipation for inviscid incompressible flows and – related to this – addressing the question whether incompressible Euler flows might develop singularities in finite time in three space dimensions according to Onsager's conjecture. Turbulence researchers interested in this question might want to dive right into this chapter, which is why this content is presented as a stand-alone chapter in this thesis with pointers to previous chapters. Chapter 8 presents an extension of the discretization framework for the incompressible Navier–Stokes equations to problems on moving domains, which is realized by the well-known arbitrary Lagrangian–Eulerian technique. An important

field of application of this approach are fluid–structure interaction problems that are subject of Chapter 9. This chapter outlines the algorithmic ingredients and the computational efficiency of this new FSI solver. Finally, Chapter 10 summarizes this work and gives perspectives on future work.

2 Discretization methods for the incompressible Navier–Stokes equations

This chapter describes the main method developments of the present thesis in terms of stable and high-order accurate splitting solvers in combination with robust discontinuous Galerkin discretizations based on a novel stabilized L^2-conforming formulation for the incompressible Navier–Stokes equations. At the time when this PhD project was initiated, it was observed that state-of-the-art methods from the literature for L^2-conforming DG discretizations of the incompressible Navier–Stokes equations lack robustness when applied to practical flow problems. In the course of this thesis, it has been possible to trace this lack of robustness back to two sources of instabilities and to propose suitable remedies for each of them. Instabilities occurring for small time step sizes (especially for coarse spatial resolutions and large Reynolds numbers) were found to be related to the DG discretization of velocity–pressure coupling terms, requiring special care in case of projection-type solution algorithms. Independently of the type of temporal discretization approach, both monolithic and projection-type Navier–Stokes DG solvers where found to lead to instabilities when applied to under-resolved, high-Reynolds-number flows, preventing the robust simulation of turbulent flow problems that are of primary importance for engineering applications. This instability was found to originate from problems in mass conservation and energy stability, and a stabilized approach has been proposed enforcing stability in a weak sense. The goal is to obtain a robust and accurate discretization approach for LES and DNS of turbulent flows without the need for explicit turbulence models. These contributions have been published in a series of research articles (Fehn et al. 2017, 2018b, 2019a, Krank et al. 2017) prior to the submission of this thesis, and are an advancement of work originally initiated in Fehn (2015). The content of this chapter is mainly based on work that has already been published in Fehn et al. (2017) and Fehn et al. (2018b).

The outline of this chapter is as follows. Section 2.1 recapitulates the state-of-the-art in terms of high-order L^2-conforming discretizations for the incompressible Navier–Stokes equations and explains novel contributions of the present work. Section 2.2 briefly presents the model problem of the incompressible Navier–Stokes equations. Section 2.3 discusses temporal discretization methods, considering both monolithic and projection-type solution algorithms. Robust discontinuous Galerkin methods for discretization in space are subject of Section 2.4. Section 2.5 presents the fully-discrete formulation and discusses aspects related to inf–sup stability, numerical integration, and time step restrictions. Detailed numerical investigations on the robustness and accuracy of the proposed methods are shown in Section 2.6, and a conclusion is provided in Section 2.7.

2.1 Motivation

Discontinuous Galerkin methods are an emerging discretization technique for the numerical simulation of flow problems. With a focus on the model problem of the incompressible Navier–Stokes equations, this section gives a detailed summary of both the state-of-the-art and novel contributions of the present work, which mainly aim at improving the robustness and accuracy of this discretization approach. This chapter is not meant as a basic introduction to fluid mechanics, numerical time integration, and finite element or discontinuous Galerkin methods. The reader is, instead, referred to standard textbooks on computational fluid dynamics, or more specifically to the textbooks by Hesthaven and Warburton (2007), Karniadakis and Sherwin (2013), Kronbichler and Persson (2021) in the context of high-order methods discussed in this work.

2.1.1 State-of-the-art

The local discontinuous Galerkin method (LDG) is analyzed in Cockburn et al. (2002) for the steady Stokes equations, in Cockburn et al. (2004) for the Oseen equations, and a locally conservative LDG method for the steady incompressible Navier–Stokes equations is proposed in Cockburn et al. (2005). By using a pressure-stabilization term, a stable equal-order formulation for the steady Navier–Stokes equations is proposed in Cockburn et al. (2009), where an upwind flux formulation is considered for the convective term and the local discontinuous Galerkin method or the interior penalty method for the discretization of the viscous term. These early works revealed that spaces other than standard L^2-conforming approximations are necessary to exactly fulfill conservation properties such as mass conservation or energy stability, leading to H^{div}-conforming and exactly divergence-free approximations, see also Cockburn et al. (2007).

The DG method of Bassi et al. (2006, 2007) for unsteady incompressible flow solves a local Riemann problem (with artificial compressibility to recover hyperbolic equations) to compute the inviscid numerical fluxes. The BR2 scheme is used to discretize the viscous term. The authors report that the method allows the use of equal-order polynomials for velocity and pressure due to the stabilization inherent to this scheme.

The interior penalty method has found most widespread use for the discretization of second derivatives. The discontinuous Galerkin methods proposed in Girault et al. (2005a,b) consider both the symmetric and the non-symmetric interior penalty method for the viscous term, Lesaint–Raviart upwinding fluxes for the convective term, and central fluxes for the pressure gradient term and velocity divergence term. In Hesthaven and Warburton (2007), Shahbazi et al. (2007), the discretization of the viscous term is also based on the symmetric interior penalty Galerkin (SIPG) method, and the convective term is written in divergence form to ensure local conservativity and is discretized using the local Lax–Friedrichs flux. This DG discretization of the convective term and viscous term is also applied in Klein et al. (2015, 2013). The work by Ferrer and Willden (2011) uses the symmetric interior penalty method for the viscous term, while the convective term is written in convective form using Lesaint–Raviart fluxes. An important difference can be observed with respect to the discretization of the velocity–pressure coupling terms. While central fluxes are used for the velocity divergence term and pressure gradient term in Klein et al. (2015, 2013), Shahbazi et al. (2007), no integration by parts of these terms is considered in Ferrer and Willden (2011), Hesthaven and Warburton (2007). This turned out to be a crucial aspect for stability (Fehn et al. 2017) and is discussed in more detail below.

Most recent works by Chalmers et al. (2019), Fehn et al. (2017, 2018b), Krank et al. (2017) use the SIPG method for the viscous term, the Lax–Friedrichs flux for the convective term, and central fluxes for the velocity–pressure coupling terms.

Mixed discontinuous–continuous approximations for velocity and pressure are proposed in Botti and Pietro (2011), Gao et al. (2017), Xu et al. (2019). Staggered discontinuous Galerkin methods with velocity and pressure unknowns defined on staggered grids are proposed in Tavelli and Dumbser (2014). A hybridizable discontinuous Galerkin method is proposed in Nguyen et al. (2011) and hybrid discontinuous Galerkin methods are considered in Lehrenfeld and Schöberl (2016) using a standard DG discretization for the convective term and an H^{div}-conforming HDG discretization for the velocity occurring in the Stokes operator, which results in an exactly divergence-free velocity. Another HDG method with pointwise divergence-free velocity field is proposed in Rhebergen and Wells (2018). A fully-explicit divergence-free DG method that eliminates the pressure from the equations is proposed in Fu (2019).

A zoo of instabilities has been reported for DG discretizations of the incompressible Navier–Stokes equations. The work by Ferrer and Willden (2011) reports instabilities for small time step sizes when using an equal-order DG method in combination with the dual splitting scheme, which has first been proposed in Hesthaven and Warburton (2007). In a later work by Ferrer et al. (2014), these instabilities are associated to inf–sup instabilities. As explained in Ferrer and Willden (2011), Ferrer et al. (2014), these instabilities shift to larger time step sizes for coarse spatial resolutions and low viscosities, so that this lower bound might be in conflict with the CFL-type upper bound on the time step size, resulting in an impractical scheme under these circumstances. Similar instabilities occurring for small time step sizes are reported in Emamy (2014), Emamy et al. (2017), Piatkowski et al. (2018), always in combination with the use of projection-type methods. The works by Joshi et al. (2016), Steinmoeller et al. (2013) report instabilities for coarse spatial resolutions and low viscosities in combination with a projection scheme, and associate the instabilities to inaccuracies of the discrete pressure projection operator which yields a velocity that is not exactly divergence-free or mass conserving. The work by Shahbazi et al. (2007) reports instabilities for the two-dimensional Orr–Sommerfeld problem using mixed-order polynomials and an algebraic splitting scheme. Instabilities for the same problem are also reported in Klein et al. (2015) using a mixed-order formulation where the incompressible Navier–Stokes equations are solved by the SIMPLE algorithm. In Chalmers et al. (2019), instabilities are reported for a two-dimensional shear layer roll-up problem, where an algebraic splitting scheme similar to the one in Shahbazi et al. (2007) is used. It is remarkable that most of the state-of-the-art methods discussed above did not demonstrate robustness for three-dimensional turbulent flows in under-resolved scenarios. This points to robustness problems of standard L^2-conforming discretizations when applied in such a setting, as explained in more detail below.

2.1.2 Novel contributions of the present work

It has been a major effort of the present thesis to shed light on the various instabilities mentioned above, to explain their origin (which actually do not arise from the use of projection methods), and to propose suitable remedies in order to obtain a robust, L^2-conforming DG discretization for the incompressible Navier–Stokes equations. In a first contribution (Krank et al. 2017), instabilities reported in the literature have been summarized and reviewed, the numerical properties of

different remedies proposed so far have been investigated, and new improved stabilization techniques have been proposed. Two subsequent contributions (Fehn et al. 2017, 2018b) have been able to identify essentially two sources of instabilities as explained in more detail below. A key ingredient in this context has been to simultaneously investigate monolithic and projection-type Navier–Stokes solvers, in order to study similarities and observe differences for otherwise identical parameters. These investigations clearly reveal that the instabilities described above are not related to a particular temporal discretization scheme or operator-splitting technique and also not to the inf–sup problem, but are instabilities resulting from the DG discretization of the incompressible Navier–Stokes equations. In the following, the origin of these instabilities is discussed in detail.

2.1.2.1 Instabilities for small time step sizes: Velocity–pressure coupling terms

A discontinuous Galerkin formulation for the high-order dual splitting scheme (Karniadakis et al. 1991) has first been proposed in Hesthaven and Warburton (2007) using equal-order approximations for velocity and pressure. Instabilities of this method have been reported in Ferrer and Willden (2011) for small time step sizes. Similar instabilities are reported in Emamy (2014), Emamy et al. (2017), Piatkowski et al. (2018). Several remedies have been proposed in the literature. In Ferrer et al. (2014), the instabilities occurring for small time step sizes are related to inf–sup instabilities and a stabilization is proposed by scaling the penalty parameter of the interior penalty method used to discretize the pressure Poisson equation by the inverse of the time step size. The work by Krank et al. (2017) reports improved stability for this approach, but the simulations eventually become unstable for smaller time step sizes. The works by Joshi et al. (2016), Steinmoeller et al. (2013) suggest to postprocess the velocity field in the projection step in order to obtain an exactly or weakly divergence-free velocity and to stabilize the discrete pressure projection operator. The analysis in Emamy (2014) suggests that the instabilities might be related to the discretization of the velocity divergence term and pressure gradient term, but the formulation is unclear regarding the imposition of boundary conditions. A modification of the splitting scheme is proposed in Emamy et al. (2017) to overcome these instabilities. This scheme has already been proposed by Leriche and Labrosse (2000), Leriche et al. (2006) in a different context and has previously been analyzed in Krank et al. (2017) in the context of instabilities for small time step sizes and discontinuous Galerkin discretizations, where the approach is found to be stable for small time step sizes but to exhibit increased spatial errors and sub-optimal convergence in space. In Piatkowski et al. (2018), the div–div penalty based projection proposed in Krank et al. (2017) is compared to a postprocessing technique using H^{div}-reconstruction with Raviart–Thomas spaces. In retrospect, some of the remedies discussed in Krank et al. (2017), Piatkowski et al. (2018) might be better described in the context of mass conservation and energy stability than in the context of the small time steps problem (see the discussion in Section 2.1.2.2), but the picture has been unclear at that time.

For the DG formulation proposed in Hesthaven and Warburton (2007) and analyzed in Ferrer and Willden (2011), Ferrer et al. (2014), it is noticeable that the velocity divergence term and pressure gradient term are not integrated by parts when deriving the weak formulation. This might be due to the following reasons:

- For the high-order dual splitting scheme (and projection methods in general), one is not forced to perform integration by parts of these terms since they appear on the right-hand

side of the pressure Poisson equation and projection equation. Hence, the resulting systems of equations are still solvable without integration by parts. Note that this is fundamentally different for a monolithic solution approach. In that case, the system of equations is not solvable which becomes obvious when looking at the fact that performing integration by parts and defining numerical fluxes for the pressure gradient term is a necessary prerequisite to enforce continuity of the pressure solution in a weak sense.

- Performing integration by parts and defining numerical fluxes also requires a treatment of boundary conditions. In case of the high-order dual splitting scheme, this is not straightforward because the intermediate velocity does not fulfill the Dirichlet boundary conditions prescribed for the velocity. Integration by parts of the velocity divergence term is mentioned in Steinmoeller et al. (2013), however, without defining a numerical flux function and specifying boundary conditions. The work by Piatkowski et al. (2018) only performs integration by parts for the velocity divergence term with central flux. Integration by parts of both terms using central fluxes is considered in Emamy (2014), Emamy et al. (2017), Krank et al. (2017), but uncertainties with respect to the treatment of boundary conditions are avoided by defining exterior values on domain boundaries as a function of interior values only, or by using inconsistent velocity Dirichlet boundary conditions.

The work by Fehn et al. (2017) has eventually gained important insights, showing that the discontinuous Galerkin formulation of the velocity–pressure coupling terms on the right-hand side of the pressure Poisson equation and the projection step play a crucial role in terms of the instabilities occurring for small time step sizes. Integration by parts of these terms with consistent boundary conditions should be performed in order to obtain a stable and robust method. In Fehn et al. (2017), a stable and high-order accurate boundary condition is proposed for the intermediate velocity field within the high-order dual splitting scheme. By comparing this new formulation to the discontinuous Galerkin formulation originally proposed in Hesthaven and Warburton (2007), it is demonstrated that the instabilities analyzed in Ferrer and Willden (2011), Ferrer et al. (2014) for small time steps sizes can be reproduced and that these instabilities occur similarly for both equal-order and mixed-order approximations. Using integration by parts of the velocity–pressure coupling terms along with central fluxes and consistent boundary conditions, this new formulation is shown to be stable in the limit of small time steps for both equal-order and mixed-order approximations. An accompanying eigenvalue analysis supports these observations. The numerical investigations in Fehn et al. (2017) therefore indicate that these instabilities are neither related to inf–sup instabilities nor to inaccuracies of the spatially discretized projection operator resulting in velocity fields that do not exactly fulfill the divergence-free constraint. In addition, the work by Fehn et al. (2017) shows that inf–sup instabilities in form of spurious pressure oscillations are present and that rates of convergence in space are sub-optimal when using equal-order polynomial approximations. Although some projection methods inherently introduce inf–sup stabilizing terms, these effects will eventually show up depending on the parameters of the discretization. As a means of verification of the results, the work by Fehn et al. (2017) compares the results for the high-order dual splitting scheme to alternative solution strategies such as a fully coupled, monolithic solution approach and pressure-correction schemes. These results lead to significantly different conclusions than those drawn in Ferrer et al. (2014) and Emamy et al. (2017). Recently, an independent study by Xu et al. (2019) has been published supporting the conclusions from Fehn et al. (2017).

2.1.2.2 Robustness for under-resolved, high-Reynolds-number flows: Mass conservation and energy stability

The instabilities observed in Chalmers et al. (2019), Klein et al. (2015), Shahbazi et al. (2007) for high-order L^2-conforming DG discretizations applied to high-Reynolds-number two-dimensional flows can not be explained by the small-time-steps instability issue discussed above. Instead, standard L^2-conforming methods summarized in 2.1.1 suffer from another type of instability. Interestingly, it can be observed that robust DG discretizations applied to the solution of three-dimensional, high-Reynolds-number flows that are characterized by turbulent flow structures and severe under-resolution have not been available for a long time. For comparison, this vacuum has been filled much earlier by the compressible DG community, see for example the early works by Collis (2002), Ramakrishnan and Collis (2004). This gives first indications that the incompressible case is plagued by additional stability problems specifically related to the nature of the incompressible Navier–Stokes equations, apart from the well-known effect of aliasing generally occurring for non-linear terms (Hesthaven and Warburton 2007). In the incompressible case, the turbulent flow simulations shown in Bassi et al. (2016), Fambri and Dumbser (2016), Piatkowski et al. (2018), Tavelli and Dumbser (2016) appear promising, but a closer inspection might reveal that the spatial resolutions used there are so high that the flow is well-resolved and the simulation can be described as DNS. A conclusion regarding the robustness of high-order DG discretizations of the incompressible Navier–Stokes equations for strongly under-resolved scenarios is therefore not possible. There are many indications that L^2-conforming discretizations are not robust in the under-resolved regime. The approaches presented in Ferrer (2012), Marek et al. (2015) use an explicit, algebraic subgrid-scale model. The method proposed in Ferrer (2017) uses a DG discretization in two dimensions and a purely spectral Fourier approach in the third dimension. Stability for turbulent flow problems is realized by scaling the penalty parameter of the SIPG method of the viscous term and a spectral vanishing viscosity (SVV) method in the Fourier direction. In Joshi et al. (2016), Steinmoeller et al. (2013), instabilities are reported in under-resolved scenarios and for low viscosities and a postprocessing step is applied to the projected velocity in order to render the discrete pressure projection operator stable. Although Joshi et al. (2016), Steinmoeller et al. (2013) originally discussed this postprocessing in relation to projection methods, the basic idea turned out to be a very general and powerful stabilization approach (Fehn et al. 2018b). Based on these ideas, consistent penalty terms added to the weak formulation such as a divergence penalty term and a continuity penalty term are proposed in Krank et al. (2017) as a means to improve mass conservation. These penalty terms provide additional control over the incompressibility constraint as well as the continuity of the velocity field between elements in a weak sense. The div–div penalty term proposed in Krank et al. (2017) has similarities with the grad–div stabilization term established for continuous finite element methods (Franca and Hughes 1988, Olshanskii et al. 2009). Around the same time, a pure grad–div stabilization term for a DG discretization of incompressible natural convection flows has been proposed in Schroeder and Lube (2017), apparently for the first time in an L^2-conforming setting. Other works have used normal-continuity penalty terms for the velocity, see Guzmán et al. (2016), Montlaur et al. (2008), and their importance for the robust simulation of turbulent flows has long been unclear. The work by Fehn et al. (2018b) gives evidence that a continuity penalty term for the normal components of the velocity across elements in addition to the divergence penalty term is necessary for robustness. Moreover, the beneficial effect

of this stabilization approach in terms of energy stability is analyzed. Theoretical justification for the approach proposed in Fehn et al. (2018b), Krank et al. (2017) is provided in Akbas et al. (2018), where this approach is described as the analogue of grad–div stabilization for non-conforming discretizations. Furthermore, the importance of deriving the penalty parameters by dimensional analysis has been highlighted in Fehn et al. (2018b), an aspect that has been overlooked in mathematically oriented literature (Akbas et al. 2018, Guzmán et al. 2016, Montlaur et al. 2008, Schroeder and Lube 2017) where penalty terms do not exhibit physically consistent units. Through a suitable estimate of the penalty parameter as a function of characteristic quantities, a parameter-free flow solver can be obtained which can be interpreted as a robust and accurate implicit LES approach. The approach is generic since it realizes dissipation through consistent penalty terms and an otherwise consistent discretization. By comparing monolithic and projection-type Navier–Stokes solvers, the work by Fehn et al. (2018b) provides insight into the stabilization from Joshi et al. (2016), Steinmoeller et al. (2013), in the sense that the need for stabilization is not specifically related to the pressure projection operator, but more generally to the L^2-conforming nature of the function space.

The stabilized L^2-conforming approach proposed here has a strong analogy to exactly divergence-free H^{div}-conforming finite element spaces. An H^{div}-conforming velocity space implies that the velocity is normal-continuous between elements. In this sense, the continuity penalty term can be interpreted as an approach that weakly enforces H^{div}-conformity. Moreover, an H^{div}-conforming method yields an exactly divergence-free velocity if the divergence of the velocity lies within the pressure space, which holds for example for Raviart–Thomas elements (Raviart and Thomas 1977) in the case of tensor product elements. One can then argue that the divergence penalty term restricts the velocity to such an exactly divergence-free space in a weak sense. This analogy has been explored in Fehn et al. (2019a), where the suitability of both approaches for the simulation of turbulent flows has been investigated in a regime where the spatial discretization scheme is challenged through under-resolution. This study reveals that the two approaches show very similar results in such a regime, and that both approaches are promising candidates for no-model (implicit) large-eddy simulation of turbulent flows due to their inherent dissipation mechanisms, see also Fehn et al. (2018b), Schroeder (2019). An important observation is that the particular formulation of the convective term and its numerical flux function is not the decisive factor in terms of robustness and accuracy. An upwind-like discretization of the convective term without further measures is not able to render the overall discretization scheme energy-stable, which is due to the nonlinearity of the convective term. It is equally important to internalize that the robustness issue discussed here is not caused by the aliasing related to inexact integration of the non-linear term, i.e., an exact integration of the non-linear term still shows these problems. The study by Fehn et al. (2019a) suggests that both energy stability and mass conservation are essential for a robust and accurate method.

Against this background, DG discretizations for the incompressible Navier–Stokes equations published to date can be grouped into two categories: Methods fulfilling the properties of mass conservation and energy stability exactly through the use of tailored function spaces (H^{div}-conforming, Raviart–Thomas) that restrict the L^2-conforming space appropriately (Cockburn et al. 2005, 2007, 2009, Fu 2019, Guzmán et al. 2016, Lehrenfeld and Schöberl 2016, Montlaur et al. 2008, Rhebergen and Wells 2018, Schroeder and Lube 2018), and methods enforcing these properties weakly through suitable stabilization terms (Akbas et al. 2018, Fehn et al. 2018b, Joshi et al. 2016, Krank et al. 2017, Schroeder and Lube 2017), thereby also "restricting" cer-

tain degrees of freedom of the L^2-conforming space. Having said that, it can be suspected that robustness remains unclear for standard L^2-conforming schemes (Bassi et al. 2006, Botti and Pietro 2011, Ferrer and Willden 2011, Hesthaven and Warburton 2007, Klein et al. 2013, Shahbazi et al. 2007) when applied to high-Reynolds-number and under-resolved flows. In particular, no good theoretical unterstanding appears to be available in the literature whether the artificial compressibility flux presented in Bassi et al. (2006) – a formulation that has been applied to the solution of turbulent flows, see for example Bassi et al. (2016), Franciolini et al. (2017) – improves stability in the context of the mass-conservation and energy-stability problem compared to other unstabilized L^2-conforming DG formulations that have been shown to lack robustness.

2.2 The incompressible Navier–Stokes equations

This section summarizes the mathematical model of the incompressible Navier–Stokes equations governing incompressible fluid flow in a domain $\Omega \subset \mathbb{R}^d$ over a time interval $[0, T]$. The incompressible Navier–Stokes equations are a set of coupled partial differential equations consisting of the momentum equation written in conservative (divergence) formulation

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) + \nabla p = \boldsymbol{f} \text{ in } \Omega \times [0, T] \tag{2.1}$$

and the continuity equation (incompressibility constraint)

$$\nabla \cdot \boldsymbol{u} = 0 \text{ in } \Omega \times [0, T] \ , \tag{2.2}$$

where the unknowns are the velocity $\boldsymbol{u} = (u_1, ..., u_d)^{\mathsf{T}}$ and the kinematic pressure p. The body force vector is denoted by $\boldsymbol{f} = (f_1, ..., f_d)^{\mathsf{T}}$. The formulation of the convective term in equation (2.1) is known as the divergence or conservative formulation with $\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) = \boldsymbol{u} \otimes \boldsymbol{u}$. The convective term can alternatively be written in convective formulation $\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} + \boldsymbol{u} \nabla \cdot \boldsymbol{u} = (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u}$ since $\nabla \cdot \boldsymbol{u} = 0$. For ease of notation, the formulation in equation (2.1) may cover both variants in the sense of

$$\nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) = \begin{cases} \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) & \text{divergence formulation} \\ (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} & \text{convective formulation} \end{cases} ,$$

and it will be explicitly distinguished between the two formulations where necessary. Regarding the viscous term, the following two formulations of the viscous flux $\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u})$ are considered

$$\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) = \begin{cases} 2\nu \boldsymbol{\varepsilon}(\boldsymbol{u}) & \text{divergence formulation} \\ \nu \nabla \boldsymbol{u} & \text{Laplace formulation} \end{cases} ,$$

where $\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathsf{T}}\right)$ is the symmetric part of the velocity gradient and ν the kinematic viscosity, which is assumed to be constant in space in the following. The Laplace formulation can be derived from the divergence formulation by making use of the incompressibility constraint

$$\nabla \cdot 2\nu \boldsymbol{\varepsilon}(\boldsymbol{u}) = \nabla \cdot (\nu \nabla \boldsymbol{u} + \nu (\nabla \boldsymbol{u})^{\mathsf{T}}) = \nabla \cdot (\nu \nabla \boldsymbol{u}) + \nu \nabla (\underbrace{\nabla \cdot \boldsymbol{u}}_{=0}) = \nabla \cdot (\nu \nabla \boldsymbol{u}) = \nu \nabla^2 \boldsymbol{u} \ .$$

By introducing a parameter γ that takes a value of 1 for the divergence formulation and a value of 0 for the Laplace formulation, the viscous term can alternatively be written as

$$-\nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) = -\nu \nabla^2 \boldsymbol{u} - \gamma \nu \nabla (\nabla \cdot \boldsymbol{u}) \ . \tag{2.3}$$

The incompressible Navier–Stokes equations (2.1) and (2.2) are subject to the initial condition

$$\boldsymbol{u}(\boldsymbol{x}, t = t_0) = \boldsymbol{u}_0(\boldsymbol{x}) \text{ in } \Omega \ , \tag{2.4}$$

where $\boldsymbol{u}_0(\boldsymbol{x})$ has to be divergence-free, $\nabla \cdot \boldsymbol{u}_0(\boldsymbol{x}) = 0$ in Ω. On the boundary $\Gamma = \partial\Omega$, Dirichlet and Neumann boundary conditions are prescribed

$$\boldsymbol{u} = \boldsymbol{g}_u \text{ on } \Gamma^{\mathrm{D}} \times [0, T] \ , \tag{2.5}$$

$$(\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) - p\boldsymbol{I}) \cdot \boldsymbol{n} = \boldsymbol{h} \ \text{ on } \Gamma^{\mathrm{N}} \times [0, T] \ , \tag{2.6}$$

where the Dirichlet and Neumann part of the boundary are denoted by Γ^{D} and Γ^{N}, respectively, with $\Gamma = \Gamma^{\mathrm{D}} \cup \Gamma^{\mathrm{N}}$ and $\Gamma^{\mathrm{D}} \cap \Gamma^{\mathrm{N}} = \emptyset$. The outward pointing unit normal vector is denoted by $\boldsymbol{n}$. The present work also discusses projection methods (introduced in Section 2.3) to numerically solve the incompressible Navier–Stokes equations, where a splitting of the Neumann boundary condition according to $\boldsymbol{h} = \boldsymbol{h}_u - g_p \boldsymbol{n}$ into a viscous part $\boldsymbol{h}_u$ and a pressure part g_p is necessary due to the operator splitting. Accordingly, the viscous forces and the pressure have to be prescribed separately on Γ^{N}

$$\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) \cdot \boldsymbol{n} = \boldsymbol{h}_u \text{ on } \Gamma^{\mathrm{N}} \times [0, T] \ , \tag{2.7}$$

$$p = g_p \ \text{ on } \Gamma^{\mathrm{N}} \times [0, T] \ . \tag{2.8}$$

For the special case of pure Dirichlet boundary conditions, $\Gamma = \Gamma^{\mathrm{D}}$, the velocity Dirichlet boundary condition has to fulfill a constraint which can be derived from the incompressibility constraint by applying Gauss' divergence theorem

$$\int_{\Omega} \nabla \cdot \boldsymbol{u} \, \mathrm{d}\Omega = \int_{\Gamma = \Gamma^{\mathrm{D}}} \boldsymbol{u} \cdot \boldsymbol{n} \, \mathrm{d}\Gamma = \int_{\Gamma^{\mathrm{D}}} \boldsymbol{g}_u \cdot \boldsymbol{n} \, \mathrm{d}\Gamma = 0 \ . \tag{2.9}$$

Moreover, the pressure is only defined up to an additive constant in this case. To obtain a unique pressure solution, one can set the mean of the pressure to zero

$$\int_{\Omega} p \, \mathrm{d}\Omega = 0 \ . \tag{2.10}$$

The Reynolds number Re describes the ratio of inertial forces to viscous forces

$$\mathrm{Re} = \frac{U \, L}{\nu} \sim \frac{\text{inertial forces}}{\text{viscous forces}} \ , \tag{2.11}$$

where U is a characteristic velocity and L a characteristic length scale of the flow problem. Depending on the Reynolds number, two limiting cases can be distinguished. The unsteady (generalized) Stokes equations are obtained from the incompressible Navier–Stokes equations by neglecting the convective term, which corresponds to the limit $\mathrm{Re} \to 0$

$$\frac{\partial \boldsymbol{u}}{\partial t} - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) + \nabla p = \boldsymbol{f} \text{ in } \Omega \times [0, T] \ , \tag{2.12}$$

$$\nabla \cdot \boldsymbol{u} = 0 \ \text{ in } \Omega \times [0, T] \ . \tag{2.13}$$

Likewise, the incompressible Euler equations are obtained in the limit $\nu = 0$ ($\mathrm{Re} \to \infty$)

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) + \nabla p = \boldsymbol{f} \text{ in } \Omega \times [0, T] \ , \tag{2.14}$$

$$\nabla \cdot \boldsymbol{u} = 0 \text{ in } \Omega \times [0, T] \ . \tag{2.15}$$

2.3 Discretization in time

Methods for the numerical solution of the incompressible Navier–Stokes equations can be classified into two categories: (i) coupled or monolithic solution methods for which the global system of equations resulting from discretization in space and time involves both the velocity and pressure unknowns, and (ii) splitting methods which aim at separating the computation of pressure and velocity unknowns in the solution algorithm.

The coupling of velocity and pressure in the momentum equation in combination with the incompressibility constraint poses a major challenge in terms of the numerical solution of the incompressible Navier–Stokes equations (Guermond et al. 2006). The monolithic approach results in a system of equations of indefinite saddle-point type for both velocity and pressure unknowns that requires elaborate preconditioning techniques in order to obtain an efficient solution algorithm. The main motivation for using splitting methods is, therefore, to improve the computational efficiency of the solution algorithm. For splitting methods, the problem is decomposed into a set of equations – a convection–diffusion type problem for the velocity and a Poisson problem for the pressure – which can be solved more efficiently from a linear algebra point of view and for which optimal solver techniques are often readily available (Guermond et al. 2006, Karniadakis and Sherwin 2013).

The class of splitting methods may be subdivided into the four main groups of algebraic splitting schemes, pressure-correction schemes, velocity-correction schemes, and consistent splitting schemes, and the reader is referred to Guermond et al. (2006), Karniadakis and Sherwin (2013) for a comprehensive overview. Pressure-correction methods are studied in Guermond and Shen (2004), Minev and Gresho (1998), Timmermans et al. (1996) and velocity-correction methods in Guermond and Shen (2003), Karniadakis et al. (1991), Orszag et al. (1986). In the present work the focus is on velocity-correction and pressure-correction methods, which can be classified as projection methods as explained in more detail below. In case of velocity-correction and pressure-correction schemes and in contrast to algebraic splitting schemes, the temporal discretization and operator splitting are applied at the level of differential equations prior to discretization in space.

The idea of projection methods can be explained by the Helmholtz decomposition of a vector field $\boldsymbol{w} = \boldsymbol{u} + \nabla\phi$ into a divergence-free part $\boldsymbol{u}$ and an irrotational part $\nabla\phi$. Exploiting that $\nabla \cdot \boldsymbol{u} = 0$, the decomposition is a two-step process consisting of the solution of a Poisson problem and a subsequent projection onto the space of divergence free vectors

$$\nabla^2 \phi = \nabla \cdot \boldsymbol{w} \ , \tag{2.16}$$

$$\boldsymbol{u} = \boldsymbol{w} - \nabla\phi \ . \tag{2.17}$$

Writing the momentum equation of the incompressible Navier–Stokes equations in the form

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla p = \frac{\partial \hat{\boldsymbol{u}}}{\partial t} \,, \tag{2.18}$$

where $\hat{\boldsymbol{u}}$ is an intermediate velocity field agglomerating convective, viscous, and right-hand side terms, the term $\frac{\partial \boldsymbol{u}}{\partial t}$ can be interpreted as the divergence-free part and the term ∇p as the irrotational part. Applying the idea of a Helmholtz decomposition described above yields a pressure Poisson equation and a subsequent projection equation

$$\nabla^2 p = \nabla \cdot \left(\frac{\partial \hat{\boldsymbol{u}}}{\partial t} \right) \,, \tag{2.19}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} = \frac{\partial \hat{\boldsymbol{u}}}{\partial t} - \nabla p \,. \tag{2.20}$$

The convective and viscous terms are treated in other sub-steps of the projection scheme involving only the velocity as unknown field. Although the basic idea of projection methods is as simple as that, obtaining high-order accurate and stable projection methods in the time-discrete case is non-trivial as explained below.

A pressure-correction scheme has first been proposed by Chorin (1968). In the first step, the momentum equation is solved neglecting the pressure gradient term. The pressure and a divergence-free velocity field are obtained in the second step by projecting the intermediate velocity onto the space of divergence-free vectors. A modified scheme that uses an extrapolation of the pressure gradient term in the first substep is used by Hirt and Cook (1972), leading to so-called incremental pressure-correction schemes. The difference to the original splitting schemes has been noticed by Goda (1979), and Van Kan (1986) has shown that this modifications achieves second-order accuracy for the velocity as opposed to the original splitting scheme that is only first-order accurate. Timmermans et al. (1996) proposed the rotational formulation of the incremental pressure-correction scheme by adding a divergence correction term to the pressure increment in the pressure Poisson equation. Since this formulation leads to a consistent Neumann boundary condition for the pressure, it essentially reduces the formation of artificial boundary layers as analyzed in Guermond and Shen (2004).

In case of velocity-correction schemes, the pressure and a divergence-free velocity field are calculated in the first substep, while the velocity is corrected in the second substep taking the viscous term into account. Karniadakis et al. (1991) proposed a high-order accurate velocity-correction scheme. The formulation of the pressure Neumann boundary condition with the viscous term written in rotational form is based on the analysis of Orszag et al. (1986) and is of particular importance regarding the accuracy of this scheme. Guermond and Shen (2003) introduce different formulations of velocity-correction schemes in analogy to pressure-correction schemes, where the incremental velocity-correction scheme in rotational formulation is formally equivalent to the high-order dual splitting scheme of Karniadakis et al. (1991).

In case of projection methods such as the pressure-correction scheme and velocity-correction scheme considered in this work, a Neumann boundary condition has to be prescribed for the pressure on Dirichlet boundaries. Inconsistent formulations of this pressure Neumann boundary condition can cause unphysical boundary layers which also limit the temporal accuracy of projection schemes (Guermond et al. 2006). To obtain higher order accuracy with respect to the temporal

discretization, consistent formulations of the pressure Neumann boundary condition (leading to so-called rotational formulations) are crucial. This aspect has been addressed in Guermond and Shen (2003), Karniadakis et al. (1991), Orszag et al. (1986) for velocity-correction methods, and in Guermond and Shen (2004), Minev and Gresho (1998), Timmermans et al. (1996) for pressure-correction methods. Moreover, it is more difficult to obtain higher than second-order accuracy in time using splitting schemes compared to coupled solution methods. A scheme allowing to obtain third-order accuracy is, e.g., the high-order dual splitting scheme of Karniadakis et al. (1991) considered in this work. The pressure-correction method considered here is known to be only conditionally stable for higher than second-order time integration schemes, leading to a lower bound for the time step size in that case (Guermond et al. 2006).

In the following, the temporal discretization is presented for a coupled solution approach on the one hand, and two widely used projection methods on the other hand. For both solution approaches, the convective term can be formulated either explicitly or implicitly in time. An explicit treatment has the advantage that the discrete system of equations is linear and can be solved more efficiently. At the same time, an explicit formulation introduces a restriction of the time step size according to the CFL condition, which might require a time step size smaller than the physical time scale of the considered problem. An implicit treatment of the convective term does not introduce a time step restriction and is, therefore, more flexible in the selection of the time step size, at the cost of a nonlinear system of equations to be solved in each time step. The pressure-correction scheme is unconditionally stable only up to second-order, but allows an implicit formulation of the convective term. In contrast, the dual-splitting projection scheme requires an explicit formulation of the convective term, but the third-order accurate scheme has been reported to be stable. Against this background, different time integration schemes are discussed in the following in order to cover different aspects of Navier–Stokes solvers and to obtain a flexible solver framework providing solvers most efficient for a certain problem.

Remark 2.1 *There are also time-stepping techniques that aim at relaxing the CFL condition while still formulating the convective term explicitly in time, see Löhner (2004), Maday et al. (1990). These are specialized time stepping techniques that apply several time steps for the convective term per global or macro time step, each of the convective time steps obeying the CFL condition. The reader is referred to Riccius (2019) for a documentation and numerical investigation of this type of approach in the context of the present incompressible Navier–Stokes solvers with DG discretization in space. In the literature, these techniques have been used for example in Karakus et al. (2019), Lehrenfeld and Schöberl (2016) in combination with DG-based incompressible Navier–Stokes solvers.*

2.3.1 BDF time integration and extrapolation schemes

This work considers BDF (backward differentiation formula) time integration schemes to discretize the incompressible Navier–Stokes equations (2.1) and (2.2) in time. Extrapolation schemes are used in case of explicit formulations of certain terms of the equations. In order to introduce the BDF schemes, consider the numerical time integration of an ordinary differential equation of the form

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = f\left(\phi(t), t\right) \ , \tag{2.21}$$

with initial condition $\phi(t=0) = \phi_0$. The time interval $[0, T]$ is divided into $N_{\Delta t}$ time steps of variable size. With $n = 0, ..., N_{\Delta t} - 1$ denoting the time step number, the equations are advanced from time t_n to $t_{n+1} = t_n + \Delta t_n$ in time step n, leading to the time grid $\{t_i\}_{i=0}^{N_{\Delta t}} = \{\sum_{j=0}^{i-1} \Delta t_j\}_{i=0}^{N_{\Delta t}}$. BDF schemes of order $J = 1, 2, 3$ are considered. Although A-stability is only achieved for time integration schemes of order $J = 1, 2$, it is often reported that third-order accurate schemes are not affected by stability issues for practical problems. The above equation is advanced from time t_n to time t_{n+1} by solving the following time-discrete problem for ϕ^{n+1}

$$\frac{\gamma_0^n \phi^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \phi^{n-i}}{\Delta t_n} = f\left(\phi^{n+1}, t_{n+1}\right) \ , \tag{2.22}$$

where γ_0^n and α_i^n are the coefficients of the BDF scheme. For the general case of variable time step sizes, the coefficients γ_0^n, α_i^n, and β_i^n vary from one time step to the next and can be expressed as simple rational functions of the time step sizes $\Delta t_n, ..., \Delta t_{n-J+1}$ as derived below. To introduce the extrapolation scheme for explicit terms, consider a decomposition of the right-hand side $f = f_{\mathrm{ex}} + f_{\mathrm{im}}$ into terms that are treated explicitly (non-stiff terms) and terms that are treated implicitly (stiff terms). For the explicit terms, an extrapolation of the right-hand side of order J is used

$$\frac{\gamma_0^n \phi^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \phi^{n-i}}{\Delta t_n} = f_{\mathrm{im}}\left(\phi^{n+1}, t_{n+1}\right) + \sum_{i=0}^{J-1} \beta_i^n f_{\mathrm{ex}}\left(\phi^{n-i}, t_{n-i}\right) \ , \tag{2.23}$$

where β_i^n are the coefficients of the extrapolation scheme. To derive the BDF time integration coefficients, consider a function $\phi(t)$ that is approximated by a Lagrange polynomial of order J with support points at $t_{n+1}, t_n, ..., t_{n-J+1}$

$$\phi(t) \approx \sum_{j=0}^{J} \ell_j(t) \phi(t_{n+1-j}) \ , \ \ell_j(t) = \prod_{i=0, i\neq j}^{J} \frac{t - t_{n+1-i}}{t_{n+1-j} - t_{n+1-i}} \ . \tag{2.24}$$

By taking the derivative of the Lagrange interpolation polynomial, the time derivative of ϕ at time t_{n+1} can be approximated as follows

$$\begin{aligned} \left.\frac{\partial \phi(t)}{\partial t}\right|_{t=t_{n+1}} &\approx \sum_{j=0}^{J} \left.\frac{\partial \ell_j(t)}{\partial t}\right|_{t=t_{n+1}} \phi(t_{n+1-j}) \\ &= \frac{1}{\Delta t_n} \left(\underbrace{\Delta t_n \left.\frac{\partial \ell_0(t)}{\partial t}\right|_{t=t_{n+1}}}_{=\gamma_0^n} \phi(t_{n+1}) - \sum_{j=0}^{J-1} \underbrace{(-\Delta t_n) \left.\frac{\partial \ell_{j+1}(t)}{\partial t}\right|_{t=t_{n+1}}}_{=\alpha_j^n} \phi(t_{n-j}) \right) . \end{aligned} \tag{2.25}$$

Table 2.1: Coefficients of BDF time integration scheme and extrapolation scheme for constant time step size, see Karniadakis et al. (1991).

	γ_0	α_0	α_1	α_2	β_0	β_1	β_2
$J=1$	1	1	-	-	1	-	-
$J=2$	$3/2$	2	$-1/2$	-	2	-1	-
$J=3$	$11/6$	3	$-3/2$	$1/3$	3	-3	1

Table 2.2: Coefficients of BDF time integration scheme and extrapolation scheme for adaptive time step sizes.

	$J=1$	$J=2$	$J=3$
γ_0^n	1	$\frac{2\Delta t_n+\Delta t_{n-1}}{\Delta t_n+\Delta t_{n-1}}$	$1+\frac{\Delta t_n}{\Delta t_n+\Delta t_{n-1}}+\frac{\Delta t_n}{\Delta t_n+\Delta t_{n-1}+\Delta t_{n-2}}$
α_0^n	1	$\frac{\Delta t_n+\Delta t_{n-1}}{\Delta t_{n-1}}$	$\frac{(\Delta t_n+\Delta t_{n-1})(\Delta t_n+\Delta t_{n-1}+\Delta t_{n-2})}{\Delta t_{n-1}(\Delta t_{n-1}+\Delta t_{n-2})}$
α_1^n	-	$-\frac{\Delta t_n^2}{(\Delta t_n+\Delta t_{n-1})\Delta t_{n-1}}$	$-\frac{\Delta t_n^2(\Delta t_n+\Delta t_{n-1}+\Delta t_{n-2})}{(\Delta t_n+\Delta t_{n-1})\Delta t_{n-1}\Delta t_{n-2}}$
α_2^n	-	-	$\frac{\Delta t_n^2(\Delta t_n+\Delta t_{n-1})}{(\Delta t_n+\Delta t_{n-1}+\Delta t_{n-2})(\Delta t_{n-1}+\Delta t_{n-2})\Delta t_{n-2}}$
β_0^n	1	$\frac{\Delta t_n+\Delta t_{n-1}}{\Delta t_{n-1}}$	$\frac{(\Delta t_n+\Delta t_{n-1})(\Delta t_n+\Delta t_{n-1}+\Delta t_{n-2})}{\Delta t_{n-1}(\Delta t_{n-1}+\Delta t_{n-2})}$
β_1^n	-	$-\frac{\Delta t_n}{\Delta t_{n-1}}$	$-\frac{\Delta t_n(\Delta t_n+\Delta t_{n-1}+\Delta t_{n-2})}{\Delta t_{n-1}\Delta t_{n-2}}$
β_2^n	-	-	$\frac{\Delta t_n(\Delta t_n+\Delta t_{n-1})}{(\Delta t_{n-1}+\Delta t_{n-2})\Delta t_{n-2}}$

The BDF coefficients are therefore given by the derivative of the Lagrange polynomials evaluated at time t_{n+1}. The procedure is similar for the extrapolation scheme. Consider a Lagrange interpolation of order $J-1$ with support points at $t_n, t_{n-1}, ..., t_{n-J+1}$

$$f_{\text{ex}}(t) \approx \sum_{j=0}^{J-1} \ell_j(t) f_{\text{ex}}(t_{n-j}) \ , \ \ell_j(t) = \prod_{i=0, i\neq j}^{J-1} \frac{t-t_{n-i}}{t_{n-j}-t_{n-i}} \ . \tag{2.26}$$

Then, an approximation for $f_{\text{ex}}(t_{n+1})$ by an extrapolation scheme of order J with coefficients β_j^n is given as follows

$$f_{\text{ex}}(t_{n+1}) \approx \sum_{j=0}^{J-1} \underbrace{\ell_j(t_{n+1})}_{=\beta_j^n} f_{\text{ex}}(t_{n-j}) \ . \tag{2.27}$$

The coefficients γ_0 and α_i of the BDF time integration scheme as well as the coefficients β_i of the extrapolation scheme are listed in Table 2.1 for the case of a constant time step size $\Delta t = T/N_{\Delta t}$, see also Karniadakis et al. (1991). The time integration constants for adaptive time-stepping are summarized in Table 2.2, and the reader is referred to Wang and Ruuth (2008) for a discussion of this type of time integration schemes in a broader context.

BDF time integration schemes are multistep schemes, i.e., in order to calculate the current time step n, the solution at previous instants of time $t_{n-J+1}, ..., t_{n-1}$ is needed for $J > 1$. As a consequence, the method is not self-starting. There are two possibilities to start the time integration scheme. The first one is relevant if an analytical solution is available and has to be used to demonstrate optimal rates of convergence with respect to the time discretization. The solution at the discrete instants of time $t_{-J+1}, ..., t_{-1}$ for $J > 1$ is obtained by interpolation of the analytical solution. The time integration scheme of order J is then applied in the very first time step and all subsequent time steps. The second approach uses schemes of lower order in the first $J - 1$ time steps, i.e., starting with a first-order method in the first time step and successively increasing the order of the time integration scheme from one time step to the next until the desired order is reached. This does not necessarily imply a loss of accuracy for practical problems, given that those problems are typically started with zero initial fields and given that an accurate temporal resolution of the initial transient related to these initial conditions is typically not relevant.

2.3.2 Coupled solution approach

Applying the BDF time integration scheme introduced above to the incompressible Navier–Stokes equations (2.1) and (2.2) and using an implicit formulation of the convective term results in the time-discrete problem

$$\begin{aligned} \frac{\gamma_0^n \boldsymbol{u}^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n} + \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}^{n+1}) - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{n+1}) + \nabla p^{n+1} &= \boldsymbol{f}\left(t_{n+1}\right) \ , \\ \nabla \cdot \boldsymbol{u}^{n+1} &= 0 \ . \end{aligned} \tag{2.28}$$

The boundary conditions are

$$\boldsymbol{u}^{n+1} = \boldsymbol{g}_u^{n+1} \ \text{ on } \Gamma^{\mathrm{D}} \ , \tag{2.29}$$

$$\left(\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{n+1}) - p^{n+1}\boldsymbol{I}\right) \cdot \boldsymbol{n} = \boldsymbol{h}^{n+1} \ \text{ on } \Gamma^{\mathrm{N}} \ . \tag{2.30}$$

An alternative explicit formulation of the convective term leads to the time-discrete problem

$$\begin{aligned} \frac{\gamma_0^n \boldsymbol{u}^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n} + \sum_{i=0}^{J-1} \beta_i^n \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}^{n-i}) - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{n+1}) + \nabla p^{n+1} &= \boldsymbol{f}\left(t_{n+1}\right) \ , \\ \nabla \cdot \boldsymbol{u}^{n+1} &= 0 \ , \end{aligned} \tag{2.31}$$

where the following boundary condition is used in the convective term

$$\boldsymbol{u}^{n-i} = \boldsymbol{g}_u^{n-i} \ \text{ on } \Gamma^{\mathrm{D}} \ . \tag{2.32}$$

2.3.3 Dual splitting scheme

The high-order dual splitting scheme of Karniadakis et al. (1991) is an operator splitting method that belongs to the class of projection methods and is based on BDF time integration. The convective term, the pressure term, and the viscous term are treated separately in different sub-steps of the splitting scheme. While the convective term is formulated explicitly, the viscous term is formulated implicitly in time.

2.3.3.1 Convective step

In the first sub-step, the convective term and the body force term are considered. An intermediate velocity field $\hat{\boldsymbol{u}}$ is obtained from the following equation

$$\frac{\gamma_0^n \hat{\boldsymbol{u}} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n} = -\sum_{i=0}^{J-1} \beta_i^n \nabla \cdot \boldsymbol{F}_{\mathrm{c}}\left(\boldsymbol{u}^{n-i}\right) + \boldsymbol{f}\left(t_{n+1}\right) \ , \tag{2.33}$$

where Dirichlet boundary conditions are imposed for the velocity field on Γ^{D} at old time instants t_{n-i}

$$\boldsymbol{u} = \boldsymbol{g}_u \text{ on } \Gamma^{\mathrm{D}} \ . \tag{2.34}$$

2.3.3.2 Pressure step and projection step

In the next sub-step, the pressure solution p^{n+1} at time t_{n+1} as well as a second intermediate velocity field $\hat{\hat{\boldsymbol{u}}}$ are computed by decomposing the intermediate velocity $\hat{\boldsymbol{u}}$ into an irrotational part ∇p^{n+1} and a solenoidal part $\hat{\hat{\boldsymbol{u}}}$ (projection method)

$$\frac{\gamma_0^n}{\Delta t_n} \hat{\hat{\boldsymbol{u}}} + \nabla p^{n+1} = \frac{\gamma_0^n}{\Delta t_n} \hat{\boldsymbol{u}} \ , \tag{2.35}$$

$$\nabla \cdot \hat{\hat{\boldsymbol{u}}} = 0 \ . \tag{2.36}$$

The fact that $\hat{\hat{\boldsymbol{u}}}$ has to be divergence-free is exploited to derive a Poisson equation for the pressure by taking the divergence of equation (2.35)

$$-\nabla^2 p^{n+1} = -\frac{\gamma_0^n}{\Delta t_n} \nabla \cdot \hat{\boldsymbol{u}} \ , \tag{2.37}$$

subject to the boundary conditions

$$\nabla p^{n+1} \cdot \boldsymbol{n} = h_p\left(t_{n+1}\right) \quad \text{on } \Gamma^{\mathrm{D}} \ , \tag{2.38}$$

$$p^{n+1} = g_p\left(t_{n+1}\right) \quad \text{on } \Gamma^{\mathrm{N}} \ . \tag{2.39}$$

The consistent Neumann boundary condition h_p is derived by multiplying the momentum equation of the incompressible Navier–Stokes equations by the normal vector $\boldsymbol{n}$ and solving for the pressure term (Karniadakis and Sherwin 2013, Karniadakis et al. 1991)

$$h_p\left(t_{n+1}\right) = -\left[\frac{\partial \boldsymbol{g}_u\left(t_{n+1}\right)}{\partial t} + \sum_{i=0}^{J_p-1} \beta_i^n \left(\nabla \cdot \boldsymbol{F}_{\mathrm{c}}\left(\boldsymbol{u}^{n-i}\right) + \nu \nabla \times \boldsymbol{\omega}^{n-i}\right) - \boldsymbol{f}\left(t_{n+1}\right)\right] \cdot \boldsymbol{n} \ . \tag{2.40}$$

The time derivative term and the body force term have been added compared to the formulation in Karniadakis et al. (1991) in order to extend the formulation to the more general case of time dependent boundary conditions and right-hand side vectors $\boldsymbol{f} \neq \boldsymbol{0}$. Since the velocity $\boldsymbol{u}$ is prescribed on Γ^{D}, the partial derivative with respect to t occurring in the time derivative term can be calculated from the given boundary data $\boldsymbol{g}_u$. However, there are scenarios where the

time derivative is not known analytically.[1] In this case, the analytical time derivative term in equation (2.40) is replaced by a BDF time derivative

$$\frac{\partial \boldsymbol{g}_u\left(t_{n+1}\right)}{\partial t} \rightarrow \frac{\gamma_0^n \boldsymbol{g}_u^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{g}_u^{n-i}}{\Delta t_n} \,. \tag{2.41}$$

The body force term is evaluated at time t_{n+1} as it does not depend on the velocity or pressure solution at time t_{n+1}. The convective term and the viscous term are formulated explicitly using an extrapolation scheme of order J_p since $\boldsymbol{u}^{n+1}$ is unknown at this point of the projection algorithm. Moreover, the viscous term is written in rotational formulation $\nabla \times \boldsymbol{\omega}$, where $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$ denotes the vorticity. This formulation is obtained by using the Laplace formulation of the viscous term, $\boldsymbol{F}_{\mathrm{v}}\left(\boldsymbol{u}\right) = \nu \nabla^2 \boldsymbol{u}$, which can be derived from the divergence formulation using $\nabla \cdot \boldsymbol{u} = 0$, and applying the vector identity $\nabla^2 \boldsymbol{u} = \nabla(\nabla \cdot \boldsymbol{u}) - \nabla \times (\nabla \times \boldsymbol{u}) = -\nabla \times (\nabla \times \boldsymbol{u})$, again making use of the incompressibility constraint $\nabla \cdot \boldsymbol{u} = 0$. The rotational formulation has first been proposed and analyzed in Karniadakis et al. (1991), Orszag et al. (1986). It is well known that the rotational formulation effectively reduces boundary divergence errors as compared to the Laplace formulation and is essential in obtaining high-order accuracy in time, see also Guermond et al. (2006), Karniadakis and Sherwin (2013). An alternative point of view is provided in Leriche and Labrosse (2000), where it is shown that the ellipticity of the Stokes operator is lost with the viscous term written in Laplace formulation.

For the intermediate velocity field $\hat{\boldsymbol{u}}_h$, a consistent boundary condition $\boldsymbol{g}_{\hat{u}}(t_{n+1})$ on Γ^{D} has to be specified in order to evaluate the divergence operator on the right-hand side of the pressure Poisson equation (2.37). As first proposed in Fehn et al. (2017), the boundary condition $\boldsymbol{g}_{\hat{u}}(t_{n+1})$ is derived by solving equation (2.33) for the intermediate velocity and using the fact that $\boldsymbol{u} = \boldsymbol{g}_u$ on Γ^{D}

$$\boldsymbol{g}_{\hat{u}}\left(t_{n+1}\right) = \sum_{i=0}^{J-1} \frac{\alpha_i^n}{\gamma_0^n} \boldsymbol{g}_u(t_{n-i}) - \frac{\Delta t_n}{\gamma_0^n} \sum_{i=0}^{J-1} \beta_i^n \nabla \cdot \boldsymbol{F}_c\left(\boldsymbol{u}^{n-i}\right) + \frac{\Delta t_n}{\gamma_0^n} \boldsymbol{f}\left(t_{n+1}\right) \,. \tag{2.42}$$

Note that applying $\boldsymbol{g}_u(t_{n+1})$ as boundary condition is inconsistent and, hence, does not yield optimal rates of convergence with respect to the temporal discretization. In combination with the spatial DG discretization discussed in Section 2.4, the consistent boundary condition is therefore essential in order to obtain a method that (i) is stable in the limit of small time step sizes, and that (ii) achieves high-order temporal accuracy.

Regarding the convective term, it appears to be less clear from the literature which formulation to choose and whether the convective form $\left(\boldsymbol{u}^{n-i} \cdot \nabla\right) \boldsymbol{u}^{n-i}$ exploiting $\nabla \cdot \boldsymbol{u} = 0$ should be preferred over the divergence form in equations (2.40) and (2.42). In the present implementation, the formulation used in the boundary conditions can be selected independently of the formulation used for the convective term in equation (2.33), and the convective formulation is used as default value.

[1] This is for example the case when using a precursor simulation strategy to prescribe inflow boundary conditions, or in case of fluid–structure interaction problems with a partitioned solution according to the Dirichlet–Neumann scheme, where the Dirichlet boundary data is only known at discrete instants of time.

The second intermediate velocity $\hat{\hat{\boldsymbol{u}}}$ is then obtained from equation (2.35) by projecting $\hat{\boldsymbol{u}}$ onto the space of divergence-free vectors

$$\hat{\hat{\boldsymbol{u}}} = \hat{\boldsymbol{u}} - \frac{\Delta t_n}{\gamma_0^n} \nabla p^{n+1} \,, \tag{2.43}$$

with the following boundary condition for the pressure

$$p^{n+1} = g_p\left(t_{n+1}\right) \text{ on } \Gamma^{\mathrm{N}} \,. \tag{2.44}$$

2.3.3.3 Viscous step

In the final step of the dual splitting scheme, the viscous term is considered leading to the following Helmholtz-like equation

$$\frac{\gamma_0^n}{\Delta t_n} \boldsymbol{u}^{n+1} - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}\left(\boldsymbol{u}^{n+1}\right) = \frac{\gamma_0^n}{\Delta t_n} \hat{\hat{\boldsymbol{u}}} \,, \tag{2.45}$$

where the velocity $\boldsymbol{u}^{n+1}$ has to fulfill the following boundary conditions

$$\boldsymbol{u}^{n+1} = \boldsymbol{g}_u\left(t_{n+1}\right) \text{ on } \Gamma^{\mathrm{D}} \,, \tag{2.46}$$

$$\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{n+1}) \cdot \boldsymbol{n} = \boldsymbol{h}_u\left(t_{n+1}\right) \text{ on } \Gamma^{\mathrm{N}} \,. \tag{2.47}$$

Remark 2.2 *For velocity-correction schemes, theoretical rates of convergence are available for the case of pure Dirichlet boundary conditions. As shown in Guermond and Shen (2003), the high-order dual splitting scheme with $J = 2$ and $J_p = 1$ is formally equivalent to the rotational velocity-correction scheme proposed by Guermond and Shen (2003) who proved stability and theoretical rates of convergence of order Δt^2 in the L^2-norm of the velocity and $\Delta t^{3/2}$ in the L^2-norm of the pressure for the velocity-correction scheme in rotational form. Numerical investigations in Leriche et al. (2006) show that the rate of convergence of $\Delta t^{3/2}$ for the pressure is related to the first order extrapolation $J_p = 1$ in the pressure Neumann boundary condition and that $J_p = 2$ has to be used to obtain optimal rates of convergence (of order Δt^2) also for the pressure. Moreover, an eigenvalue analysis in Leriche et al. (2006) reveals that the high-order dual splitting scheme is only conditionally stable for $J_p > 2$, while it is unconditionally stable for $J_p \leq 2$, independently of the order $1 \leq J \leq 4$ of the BDF scheme. According to that analysis, among the schemes that provide unconditional stability, the method with $J_p = 2$ (and $J = 3$) achieves the highest rates of convergence of order Δt^3 for the velocity and $\Delta t^{5/2}$ for the pressure.*

2.3.4 Pressure-correction scheme

This section describes the operator splitting technique and the temporal discretization of pressure-correction schemes, presenting different formulations of pressure-correction schemes that are summarized in Guermond et al. (2006). In contrast to the dual splitting scheme, the convective term and the viscous term are treated in the same sub-step of the splitting scheme, where the convective term can be formulated either explicitly or implicitly.

2.3.4.1 Momentum step

In the first sub-step, the momentum equation is solved by either neglecting the pressure gradient term (non-incremental formulation) or by using an extrapolation of the pressure gradient term based on the pressure solution at previous instants of time (incremental formulation). Using an implicit formulation of the convective term, an intermediate velocity field $\hat{\boldsymbol{u}}$ is calculated in the momentum step by solving the following nonlinear equation

$$\frac{\gamma_0^n \hat{\boldsymbol{u}} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n} + \nabla \cdot \boldsymbol{F}_{\mathrm{c}}\left(\hat{\boldsymbol{u}}\right) - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}\left(\hat{\boldsymbol{u}}\right) = -\sum_{i=0}^{J_p-1} \beta_i^n \nabla p^{n-i} + \boldsymbol{f}\left(t_{n+1}\right) \ . \tag{2.48}$$

In case of an explicit formulation of the convective term, the momentum step reads

$$\begin{aligned} \frac{\gamma_0^n \hat{\boldsymbol{u}} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n} - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}\left(\hat{\boldsymbol{u}}\right) = &-\sum_{i=0}^{J-1} \beta_i^n \nabla \cdot \boldsymbol{F}_{\mathrm{c}}\left(\boldsymbol{u}^{n-i}\right) \\ &- \sum_{i=0}^{J_p-1} \beta_i^n \nabla p^{n-i} + \boldsymbol{f}\left(t_{n+1}\right) \ . \end{aligned} \tag{2.49}$$

The order of the extrapolation of the pressure gradient term is denoted by J_p. Schemes with $J_p = 0$ are called non-incremental pressure-correction schemes, while schemes with $J_p \geq 1$ are called incremental pressure-correction schemes

$$J_p = \begin{cases} 0 & \text{non-incremental formulation,} \\ \geq 1 & \text{incremental formulation .} \end{cases} \tag{2.50}$$

As shown in Section 2.3.4.2, this is due to the fact that a Poisson equation has to be solved for the *pressure* in case of the non-incremental formulation, and for the *pressure increment* in case of the incremental formulation. The boundary conditions are

$$\hat{\boldsymbol{u}} = \boldsymbol{g}_u\left(t_{n+1}\right) \quad \text{on } \Gamma^{\mathrm{D}} \ , \tag{2.51}$$

$$\boldsymbol{F}_{\mathrm{v}}(\hat{\boldsymbol{u}}) \cdot \boldsymbol{n} = \boldsymbol{h}_u\left(t_{n+1}\right) \quad \text{on } \Gamma^{\mathrm{N}} \tag{2.52}$$

for the intermediate velocity field $\hat{\boldsymbol{u}}$, and

$$p^{n-i} = g_p\left(t_{n-i}\right) \quad \text{on } \Gamma^{\mathrm{N}} \tag{2.53}$$

for the pressure p^{n-i}.

2.3.4.2 Pressure step and projection step

In the second sub-step, the velocity $\boldsymbol{u}^{n+1}$ and the pressure p^{n+1} at time t_{n+1} are obtained as the solution of the following projection method

$$\frac{\gamma_0^n}{\Delta t_n} \boldsymbol{u}^{n+1} + \nabla \phi^{n+1} = \frac{\gamma_0^n}{\Delta t_n} \hat{\boldsymbol{u}} \ , \tag{2.54}$$

$$\nabla \cdot \boldsymbol{u}^{n+1} = 0 \ , \tag{2.55}$$

where ϕ^{n+1} is defined as

$$\phi^{n+1} = p^{n+1} - \sum_{i=0}^{J_p-1} \beta_i^n p^{n-i} + \chi \nu \nabla \cdot \hat{\boldsymbol{u}} \,, \tag{2.56}$$

explaining the incremental/non-incremental terminology. Depending on the parameter χ, the formulation is called standard ($\chi = 0$) or rotational ($\chi = 1 + \gamma$). In Guermond et al. (2006), only the Laplace formulation of the viscous term, $\gamma = 0$, is considered, for which the rotational formulation corresponds to $\chi = 1$. As derived in Section 2.3.4.3, a value of $\chi = 2$ yields the rotational formulation in case that the viscous term is written in divergence formulation, $\gamma = 1$.

To obtain the pressure solution, a Poisson equation is solved for the pressure increment which can be derived by taking the divergence of equation (2.54) and making use of the divergence-free condition (2.55)

$$-\nabla^2 \phi^{n+1} = -\frac{\gamma_0^n}{\Delta t_n} \nabla \cdot \hat{\boldsymbol{u}} \,, \tag{2.57}$$

with boundary conditions

$$\nabla \phi^{n+1} \cdot \boldsymbol{n} = h_\phi(t_{n+1}) = 0 \text{ on } \Gamma^{\mathrm{D}} \,, \tag{2.58}$$

$$\phi^{n+1} = g_\phi(t_{n+1}) = g_p\left(t_{n+1}\right) - \sum_{i=0}^{J_p-1} \beta_i^n g_p\left(t_{n-i}\right) \text{ on } \Gamma^{\mathrm{N}} \,. \tag{2.59}$$

The boundary conditions (2.58) and (2.59) are in line with (Guermond et al. 2006, boundary conditions (10.3)), except that the formulation is extended towards a more general treatment of boundary conditions including also inhomogeneous pressure boundary conditions on the Neumann part Γ^{N} of the boundary. Subsequently, p^{n+1} is calculated from equation (2.56)

$$p^{n+1} = \phi^{n+1} + \sum_{i=0}^{J_p-1} \beta_i^n p^{n-i} - \chi \nu \nabla \cdot \hat{\boldsymbol{u}} \,, \tag{2.60}$$

where boundary condition (2.51) is prescribed when evaluating the velocity divergence operator in the DG context.

The final velocity $\boldsymbol{u}^{n+1}$ is then obtained from equation (2.54) by projecting $\hat{\boldsymbol{u}}$ onto the space of divergence-free vectors

$$\boldsymbol{u}^{n+1} = \hat{\boldsymbol{u}} - \frac{\Delta t_n}{\gamma_0^n} \nabla \phi^{n+1} \,. \tag{2.61}$$

The evaluation of the gradient operator applied to the pressure increment ϕ^{n+1} on the right-hand side of the above equation requires boundary condition (2.59) to be prescribed.

2.3.4.3 Rotational formulation and pressure Neumann boundary condition

If the standard formulation of the pressure correction scheme is used ($\chi = 0$), an unphysical Neumann boundary condition, $\nabla p^{n+1} \cdot \boldsymbol{n} = \sum_{i=0}^{J_p-1} \beta_i^n \nabla p^{n-i} \cdot \boldsymbol{n}$ on Γ^{D}, is imposed for the

presssure according to equations (2.58) and (2.56), leading to $\nabla p^{n+1} \cdot \boldsymbol{n} = 0$ for the non-incremental formulation and $\nabla p^{n+1} \cdot \boldsymbol{n} = \nabla p^{n} \cdot \boldsymbol{n} = ... = \nabla p^{0} \cdot \boldsymbol{n}$ for the incremental formulation. As argued in Guermond et al. (2006), this unphysical Neumann boundary condition induces the formation of numerical boundary layers and limits the temporal accuracy of the projection method.

Alternatively, by inserting equations (2.48) and (2.56) into equation (2.54) and multiplying the resulting equation by the normal vector $\boldsymbol{n}$, the pressure Neumann boundary condition that is imposed on Γ^{D} reads

$$\begin{aligned} \nabla p^{n+1} \cdot \boldsymbol{n} = & - \left[\frac{\gamma_0^n \boldsymbol{u}^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n} + \nabla \cdot \boldsymbol{F}_{\mathrm{c}}\left(\hat{\boldsymbol{u}}\right) - \boldsymbol{f}\left(t_{n+1}\right) \right] \cdot \boldsymbol{n} \\ & - \left[-\nabla \cdot \boldsymbol{F}_{\mathrm{v}}\left(\hat{\boldsymbol{u}}\right) + \chi \nu \nabla (\nabla \cdot \hat{\boldsymbol{u}}) \right] \cdot \boldsymbol{n} \, . \end{aligned} \tag{2.62}$$

Equation (2.62) can be seen in analogy to equation (2.40) for the dual splitting scheme. The above equation highlights that the standard formulation of the viscous term $-\nabla \cdot \boldsymbol{F}_{\mathrm{v}}\left(\hat{\boldsymbol{u}}\right)$ is used on the right-hand side of the pressure Neumann boundary condition for $\chi = 0$, while the rotational formulation $\nu \nabla \times (\nabla \times \hat{\boldsymbol{u}})$ is applied for $\chi = 1 + \gamma$

$$\begin{aligned} -\nabla \cdot \boldsymbol{F}_{\mathrm{v}}\left(\hat{\boldsymbol{u}}\right) + \chi \nu \nabla (\nabla \cdot \hat{\boldsymbol{u}}) &= -\nu \nabla^2 \hat{\boldsymbol{u}} - \gamma \nu \nabla (\nabla \cdot \hat{\boldsymbol{u}}) + \chi \nu \nabla (\nabla \cdot \hat{\boldsymbol{u}}) \\ &= \nu \nabla \times (\nabla \times \hat{\boldsymbol{u}}) - (1+\gamma) \nu \nabla (\nabla \cdot \hat{\boldsymbol{u}}) + \chi \nu \nabla (\nabla \cdot \hat{\boldsymbol{u}}) \\ &= \nu \nabla \times (\nabla \times \hat{\boldsymbol{u}}) \quad \text{if} \quad \chi = 1 + \gamma \, . \end{aligned} \tag{2.63}$$

In the first step of the above derivation, equation (2.3) is inserted, while the vector identity $\nabla^2 \boldsymbol{u} = \nabla(\nabla \cdot \boldsymbol{u}) - \nabla \times (\nabla \times \boldsymbol{u})$ is used in the second step. The last step highlights that $\chi = 1 + \gamma$ (and especially $\chi = 2$ in case of the divergence formulation of the viscous term) has to be used to obtain the rotational formulation.

Remark 2.3 *Theoretical rates of convergence of pressure-correction schemes are derived in Guermond and Shen (2004) and summarized in Guermond et al. (2006). The non-incremental pressure-correction scheme with $J = 1$, $J_p = 0$ in standard form is Δt accurate in the L^2-norm of the velocity and $\Delta t^{1/2}$ accurate in the L^2-norm of the pressure. The incremental pressure-correction scheme with $J = 2$, $J_p = 1$ is Δt^2 accurate in the L^2-norm of the velocity for both the standard formulation and the rotational formulation. While the standard formulation achieves an accuracy of order Δt in the L^2-norm of the pressure, the rotational form is $\Delta t^{3/2}$ accurate in the L^2-norm of the pressure. As reported in Guermond et al. (2006), numerical results give evidence that pressure-correction schemes are only conditionally stable for $J_p > 1$. For this reason, only schemes with $J_p \leq 1$ (and $J \leq 2$), which are unconditionally stable, are considered in the present work.*

2.4 Discretization in space

This section discusses discontinuous Galerkin – or more precisely – L^2-conforming discretizations of the incompressible Navier–Stokes equations.

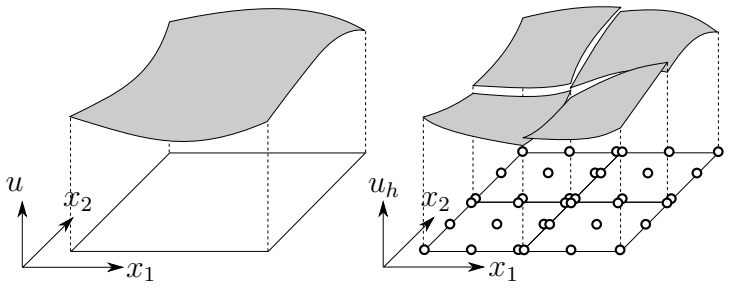


Figure 2.1: Illustration of L^2-conforming function space in two space dimensions ($d = 2$) for tensor-product elements of polynomial degree $k = 2$.

2.4.1 Notation

The physical domain Ω is approximated by the computational domain $\Omega_h \in \mathbb{R}^d$ with boundary $\Gamma_h = \partial\Omega_h$, where $\Gamma_h = \Gamma_h^{\mathrm{D}} \cup \Gamma_h^{\mathrm{N}}$ and $\Gamma_h^{\mathrm{D}} \cap \Gamma_h^{\mathrm{N}} = \emptyset$. The computational domain Ω_h consists of N_{el} non-overlapping finite elements Ω_e

$$\Omega_h = \bigcup_{e=1}^{N_{\mathrm{el}}} \Omega_e \ , \tag{2.64}$$

where quadrilateral/hexahedral element geometries with a tensor-product structure are considered in this work. The velocity $\boldsymbol{u}(\boldsymbol{x},t)$ and pressure $p(\boldsymbol{x},t)$ are approximated by functions $\boldsymbol{u}_h(\boldsymbol{x},t) \in \mathcal{V}_h^u$ and $p_h(\boldsymbol{x},t) \in \mathcal{V}_h^p$. In the context of discontinuous Galerkin finite element methods, the solution is polynomial inside elements but discontinuous between elements, and the global solution is typically written as the direct sum of element-local solutions (Hesthaven and Warburton 2007)

$$\boldsymbol{u}_h\left(\boldsymbol{x},t\right) = \bigoplus_{e=1}^{N_{\mathrm{el}}} \boldsymbol{u}_h^e\left(\boldsymbol{x},t\right) \ , \quad p_h\left(\boldsymbol{x},t\right) = \bigoplus_{e=1}^{N_{\mathrm{el}}} p_h^e\left(\boldsymbol{x},t\right) \ . \tag{2.65}$$

The spaces of test and trial functions for velocity and pressure are defined as

$$\mathcal{V}_h^u = \left\{ \boldsymbol{u}_h \in \left[L^2(\Omega_h)\right]^d \ : \ \boldsymbol{u}_h\left(\boldsymbol{x}^e(\boldsymbol{\xi})\right)|_{\Omega_e} = \tilde{\boldsymbol{u}}_h^e(\boldsymbol{\xi})|_{\tilde{\Omega}_e} \in \mathcal{V}_{h,e}^u = [\mathcal{Q}_{k_u}(\tilde{\Omega}_e)]^d \ \forall e \right\} \ , \tag{2.66}$$

$$\mathcal{V}_h^p = \left\{ p_h \in L^2(\Omega_h) \ : \ p_h\left(\boldsymbol{x}^e(\boldsymbol{\xi})\right)|_{\Omega_e} = \tilde{p}_h^e(\boldsymbol{\xi})|_{\tilde{\Omega}_e} \in \mathcal{V}_{h,e}^p = \mathcal{Q}_{k_p}(\tilde{\Omega}_e) \ \forall e \right\} \ , \tag{2.67}$$

respectively, where $\mathcal{Q}_k(\tilde{\Omega}_e) = \mathbb{P}_k \otimes \ldots \otimes \mathbb{P}_k$ denotes the space of polynomials of tensor degree $\leq k$ on the reference element $\tilde{\Omega}_e = [0,1]^d$ with reference coordinates $\boldsymbol{\xi} = (\xi_1, ..., \xi_d)^{\mathsf{T}}$. An illustration of the L^2-conforming space is shown in Figure 2.1. A nodal approach is used so that the approximate solutions of velocity and pressure on element e can be written as

$$\tilde{\boldsymbol{u}}_h^e(\boldsymbol{\xi},t) = \sum_{i_1,\ldots,i_d=0}^{k_u} \ell_{i_1\ldots i_d}^{k_u}(\boldsymbol{\xi}) \boldsymbol{u}_{i_1\ldots i_d}^e(t) \ , \ \tilde{p}_h^e(\boldsymbol{\xi},t) = \sum_{i_1,\ldots,i_d=0}^{k_p} \ell_{i_1\ldots i_d}^{k_p}(\boldsymbol{\xi}) p_{i_1\ldots i_d}^e(t) \ , \tag{2.68}$$

where $\boldsymbol{u}_{i_1\ldots i_d}^e$ and $p_{i_1\ldots i_d}^e$ denote the nodal degrees of freedom of the velocity and pressure solution on element e, respectively. The multidimensional shape functions $\ell_{i_1\ldots i_d}^k$ are given as the tensor

product of one-dimensional shape functions, $\ell^k_{i_1\dots i_d}(\boldsymbol{\xi}) = \prod_{n=1}^{d} \ell^k_{i_n}(\xi_n)$, where $\ell^k_i(\xi) \in \mathbb{P}_k$ are the Lagrange polynomials of degree k

$$\ell^k_i(\xi) = \prod_{\substack{j=0 \\ j\neq i}}^{k} \frac{\xi - \xi_j}{\xi_i - \xi_j} \ , \quad i = 0, \dots, k \ , \tag{2.69}$$

where $\{\xi_j\}_{j=0}^k$ is the set of nodes defined on the unit interval $[0, 1]$. The Lagrange polynomials satisfy the property $\ell^k_i(\xi_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta. It is well-known that the choice of equidistant interpolation points leads to poor conditioning for high polynomial degrees. In the course of this thesis, the node distribution is therefore based on the Legendre–Gauss–Lobatto (LGL) nodes, which ensure optimal interpolation quality by minimizing the Lebesque constant (Hesthaven and Warburton 2007)

$$\Lambda = \max_{\xi} \sum_{i=0}^{k} \left| l^k_i(\xi) \right| \ . \tag{2.70}$$

Compared to equidistant points, the nodes are distributed more densely towards the edges of the unit interval for LGL nodes. Nodal Lagrange polynomials based on LGL nodes show beneficial properties in terms of iteration counts for linear solvers with point-Jacobi-type preconditioners, such as multigrid methods with Chebyshev-accelerated Jacobi smoothing that are employed in this work (see Chapter 5). This is due to a better diagonal dominance of the associated linear operator as compared to a modal basis, used for example in Ferrer and Willden (2011), or a nodal Lagrange basis with Legendre–Gauss quadrature points as interpolation points.

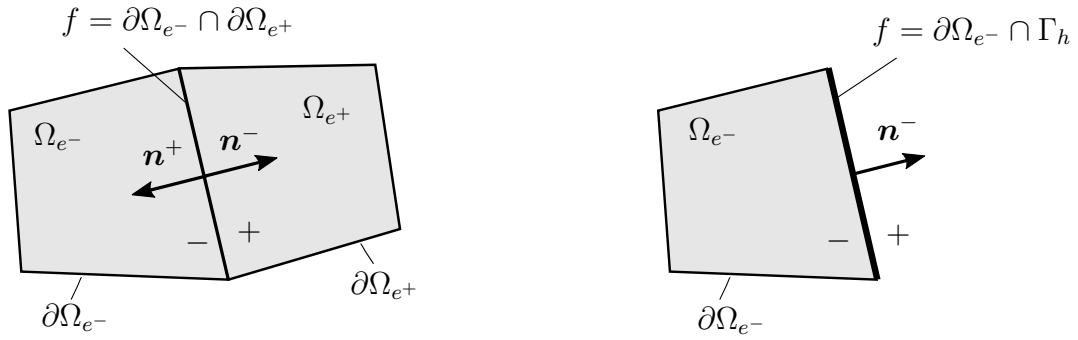
In the above equations, $\boldsymbol{x}^e(\boldsymbol{\xi}) : \tilde{\Omega}_e \rightarrow \Omega_e$ denotes the mapping from reference space to physical space. For the mapping, the same ansatz is used as for approximating the solution, but with an independent polynomial degree k_m (typically $k_\mathrm{m} \leq k_u$) and continuity between elements

$$\boldsymbol{x}^e(\boldsymbol{\xi}) = \sum_{i_1,\dots,i_d=0}^{k_\mathrm{m}} \ell^{k_\mathrm{m}}_{i_1\dots i_d}(\boldsymbol{\xi}) \boldsymbol{x}^e_{i_1\dots i_d} \ . \tag{2.71}$$

Remark 2.4 *Despite the H^1-conforming nature of the mapping space compared to the L^2-conforming solution space, the mapping might be denoted as isoparametric if $k_\mathrm{m} = k_u$.*

Unless specified otherwise, mixed-order polynomials of degree $(k_u, k_p) = (k, k-1)$ for velocity and pressure are used for reasons of inf–sup stability. Projection-type Navier–Stokes solvers might implicitly introduce inf–sup stabilizing terms so that also equal-order polynomials are sometimes used in this case. The reader is referred to Section 2.5.4 for a more detailed discussion of this aspect.

The interface of two adjacent elements Ω_{e^-} and Ω_{e^+} is denoted by $f_{e^-/e^+} = \partial\Omega_{e^-} \cap \partial\Omega_{e^+}$ where the outward pointing normal vectors on f_{e^-/e^+} are denoted by $\boldsymbol{n}^-$ for Ω_{e^-} and $\boldsymbol{n}^+$ for Ω_{e^+}. Furthermore, let u_h^- and u_h^+ denote the solution u_h on f_{e^-/e^+} evaluated from the interior of element e^- and element e^+, respectively. The set of all interior faces is denoted by Γ_h^int. The average operator $\{\{\bullet\}\}$ and jump operator $[\![\bullet]\!]$ are defined as $\{\{u\}\} = (u^- + u^+)/2$ and $[\![u]\!] = u^- \otimes \boldsymbol{n}^- + u^+ \otimes \boldsymbol{n}^+$, respectively. Note that both operators can be applied to a scalar, vectorial



(a) Interior face

(b) Boundary face

Figure 2.2: Introduction of notation used to formulate numerical fluxes.

or tensorial quantity u. Moreover, the oriented jump operator $[u]$ is introduced as $[u] = u^- - u^+$. This work makes use of the convention that interior information on the current element Ω_e is denoted by the superscript $(\cdot)^-$ and exterior information from neighboring elements by the superscript $(\cdot)^+$. Accordingly, the normal vector $\boldsymbol{n}$ of the current element Ω_e is equal to $\boldsymbol{n}^-$, while $\boldsymbol{n}^+ = -\boldsymbol{n}^- = -\boldsymbol{n}$. Following Arnold et al. (2000), some basic properties of a numerical flux F^* approximating the physical flux $F(u)$ can be introduced:

- Local: The numerical flux depends on the solution u_h^- and u_h^+ only, $F^* = F^*(u_h^-, u_h^+)$.
- Consistent: The physical flux is recovered when inserting the exact solution $u_h^- = u_h^+ = u$ into the numerical flux function, $F^*(u_h^- = u, u_h^+ = u) = F(u)$.
- Conservative: The flux is the same independently of the side of the interface from which the flux is evaluated, $F^{*,-}(u_h^-, u_h^+) = F^{*,+}(u_h^+, u_h^-)$.

Defining numerical fluxes in terms of $\{\{u\}\}$ and $[\![u]\!]$ guarantees conservativity of the numerical flux. Adding jump terms to a numerical flux does not change consistency.

An element-by-element formulation is used when deriving the weak formulation, i.e., volume integrals are performed over the current element Ω_e and face integrals over the boundary $\partial\Omega_e$ of element e. Integrals over Ω_e and $\partial\Omega_e$ are abbreviated by using the shorthand notation for L^2-products

$$(v,u)_{\Omega_e} = \int_{\Omega_e} v \odot u \, \mathrm{d}\Omega \, , \; (v,u)_{\partial\Omega_e} = \int_{\partial\Omega_e} v \odot u \, \mathrm{d}\Gamma \, , \tag{2.72}$$

where the operator $\odot$ symbolizes inner products, i.e., vu for rank-0 tensors, $\boldsymbol{v} \cdot \boldsymbol{u} = \sum_i v_i u_i$ for rank-1 tensors, and $\boldsymbol{v} : \boldsymbol{u} = \sum_{i,j} v_{ij} u_{ij}$ for rank-2 tensors. An integral over the computational domain is to be understood as $(v,u)_{\Omega_h} = \sum_{e=1}^{N_\mathrm{el}} (v,u)_{\Omega_e}$, and similarly for integrals over all interior faces, e.g., $(\{\{v\}\}, u^*)_{\Gamma_h^\mathrm{int}} = \sum_{e=1}^{N_\mathrm{el}} \left(\frac{1}{2}v, u^*\right)_{\partial\Omega_e \setminus \Gamma_h}$ if u^* is single-valued.

Remark 2.5 *In the literature, methods of degree $k \geq 2$ are typically denoted as high-order methods (Wang et al. 2013). In the course of this thesis, polynomial degrees in the range $2 \leq k \leq$*

15 are typically studied, where degrees of $k = 2, 3$ are denoted as low, degrees of $k = 3, \dots, 7$ as moderately high, and degrees of $k = 8, \dots, 15$ as very high polynomial degrees. This categorization is qualitative in nature and is motivated from aspects of accuracy and computational efficiency.

2.4.2 Derivation of discontinuous Galerkin formulation

This section derives the weak discontinuous Galerkin formulation of the incompressible Navier–Stokes equations. The general procedure consists of two steps, and is illustrated by applying it to the incompressible Navier–Stokes equations (2.1) and (2.2) with the convective term written in divergence formulation. The variational form is derived by

(i) requiring the residuals of the momentum and continuity equations to be orthogonal to all test functions $\boldsymbol{v}_h \in \mathcal{V}_h^u$ and $q_h \in \mathcal{V}_h^p$, respectively. Multiplying the residual of the momentum equation by test functions $\boldsymbol{v}_h$ and the residual of the continuity equation by test functions q_h as well as integration over Ω_h yields the following set of equations

$$\begin{aligned} \left(\boldsymbol{v}_h, \frac{\partial \boldsymbol{u}_h}{\partial t}\right)_{\Omega_e} + \left(\boldsymbol{v}_h, \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h)\right)_{\Omega_e} \\ - \left(\boldsymbol{v}_h, \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}_h)\right)_{\Omega_e} + \left(\boldsymbol{v}_h, \nabla p_h\right)_{\Omega_e} - \left(\boldsymbol{v_h}, \boldsymbol{f}(t)\right)_{\Omega_e} = 0 \;\; \forall \boldsymbol{v}_h \in \mathcal{V}_{h,e}^u \,, \end{aligned} \tag{2.73}$$

$$\left(q_h, -\nabla \cdot \boldsymbol{u}_h\right)_{\Omega_e} = 0 \;\; \forall q_h \in \mathcal{V}_{h,e}^p \,, \tag{2.74}$$

for all elements $e = 1, \dots N_{\mathrm{el}}$. The problem is stated in an element-wise manner due to the discontinuity of shape functions between elements.

(ii) and by performing integration by parts. In this step, Gauss' divergence theorem is applied in order to transform volume integrals into surface integrals. Subsequently, physical fluxes are replaced by numerical fluxes in order to enforce continuity in a weak sense. Numerical fluxes are defined as a function of the approximate solution on both elements adjacent to an interior face and as a function of the interior solution and prescribed boundary data on boundary faces. By the example of the convective term, this second step can be generically written as

$$\left(\boldsymbol{v}_h, \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h)\right)_{\Omega_e} \quad \rightarrow \quad c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h; \boldsymbol{g}_u\right) \,. \tag{2.75}$$

This results in the following discontinuous Galerkin formulation: Find $\boldsymbol{u}_h \in \mathcal{V}_h^u$, $p_h \in \mathcal{V}_h^p$ such that

$$\begin{aligned} m_{h,u}^e\left(\boldsymbol{v}_h, \frac{\partial \boldsymbol{u}_h}{\partial t}\right) + c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h; \boldsymbol{g}_u\right) \\ + v_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h; \boldsymbol{g}_u, \boldsymbol{h}_u\right) + g_h^e\left(\boldsymbol{v}_h, p_h; g_p\right) - b_h^e\left(\boldsymbol{v}_h, \boldsymbol{f}(t)\right) = 0 \,, \end{aligned} \tag{2.76}$$

$$-d_h^e(q_h, \boldsymbol{u}_h; \boldsymbol{g}_u) = 0 \,, \tag{2.77}$$

for all $(\boldsymbol{v}_h, q_h) \in \mathcal{V}_{h,e}^u \times \mathcal{V}_{h,e}^p$ and for all elements $e = 1, \dots, N_{\mathrm{el}}$. The minus sign is inserted in equation (2.74) and equation (2.77) to ensure that the matrix representation of the (linearized)

Table 2.3: Choice of exterior values $(\cdot)^+$ on domain boundaries as a function of interior values $(\cdot)^-$ and prescribed boundary data for velocity and pressure in order to weakly impose boundary conditions according to the mirror principle.

	Γ_h^{D}	Γ_h^{N}
velocity	$\boldsymbol{u}_h^+ = -\boldsymbol{u}_h^- + 2\boldsymbol{g}_u$	$\boldsymbol{u}_h^+ = \boldsymbol{u}_h^-$
	$\boldsymbol{F}_{\mathrm{v}}^+(\boldsymbol{u}_h^+)\cdot\boldsymbol{n} = \boldsymbol{F}_{\mathrm{v}}^-(\boldsymbol{u}_h^-)\cdot\boldsymbol{n}$	$\boldsymbol{F}_{\mathrm{v}}^+(\boldsymbol{u}_h^+)\cdot\boldsymbol{n} = -\boldsymbol{F}_{\mathrm{v}}^-(\boldsymbol{u}_h^-)\cdot\boldsymbol{n} + 2\boldsymbol{h}_u$
pressure	$p_h^+ = p_h^-$	$p_h^+ = -p_h^- + 2g_p$
	$\nabla p_h^+\cdot\boldsymbol{n} = -\nabla p_h^-\cdot\boldsymbol{n} + 2h_p$	$\nabla p_h^+\cdot\boldsymbol{n} = \nabla p_h^-\cdot\boldsymbol{n}$

system of equations corresponding to the weak formulation (2.76) and (2.77) is symmetric with respect to the pressure gradient term and the velocity divergence term.

The mass matrix term and body force term do not contain spatial derivative operators. Hence, there is no need to perform step (ii) described above. The velocity mass matrix operator is given in elementwise notation as

$$m_{h,u}^e(\boldsymbol{v}_h, \boldsymbol{u}_h) = (\boldsymbol{v}_h, \boldsymbol{u}_h)_{\Omega_e} \ , \tag{2.78}$$

and the body force operator as

$$b_h^e(\boldsymbol{v}_h, \boldsymbol{f}) = (\boldsymbol{v}_h, \boldsymbol{f})_{\Omega_e} \ . \tag{2.79}$$

A detailed description of the convective term c_h^e, viscous term v_h^e, pressure gradient term g_h^e, and velocity divergence term d_h^e is given below. Apart from these operators, the DG formulation of the negative Laplace operator l_h^e is introduced, which is required when discretizing the projection-type solution methods in space.

2.4.2.1 Convective term

This section derives DG formulations for the divergence formulation and convective formulation of the nonlinear transport term.

2.4.2.1.1 Divergence formulation The discontinuous Galerkin formulation of the convective term written in divergence formulation is derived by performing step (ii) described above. Integration by parts of the convective term $(\boldsymbol{v}_h, \nabla\cdot\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h))_{\Omega_e}$ and replacing the physical flux $\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h)$ by the numerical flux $\boldsymbol{F}_{\mathrm{c}}^*(\boldsymbol{u}_h)$ yields

$$c_h^e(\boldsymbol{v}_h, \boldsymbol{u}_h; \boldsymbol{g}_u) = -(\nabla\boldsymbol{v}_h, \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h))_{\Omega_e} + (\boldsymbol{v}_h, \boldsymbol{F}_{\mathrm{c}}^*(\boldsymbol{u}_h)\cdot\boldsymbol{n})_{\partial\Omega_e} \ . \tag{2.80}$$

The local Lax–Friedrichs flux is defined as (Hesthaven and Warburton 2007, Klein et al. 2013, Shahbazi et al. 2007)

$$\boldsymbol{F}_{\mathrm{c}}^*(\boldsymbol{u}_h) = \{\{\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h)\}\} + \frac{\Lambda}{2}[\![\boldsymbol{u}_h]\!] \ , \tag{2.81}$$

where $\Lambda = \max\left(\lambda^-, \lambda^+\right)$, and where λ is the maximum eigenvalue (in terms of absolute values) of the flux Jacobian

$$\lambda^{\pm} = \max_i \left| \lambda_i \left(\left. \frac{\partial \boldsymbol{F}(\boldsymbol{u}) \cdot \boldsymbol{n}}{\partial \boldsymbol{u}} \right|_{\boldsymbol{u}_h^{\pm}} \right) \right| = 2|\boldsymbol{u}_h^{\pm} \cdot \boldsymbol{n}| \ . \tag{2.82}$$

The flux Jacobian is given as

$$\frac{\partial \boldsymbol{F}(\boldsymbol{u}) \cdot \boldsymbol{n}}{\partial \boldsymbol{u}} = (\boldsymbol{u} \cdot \boldsymbol{n})\, \boldsymbol{I} + \boldsymbol{u} \otimes \boldsymbol{n} \ . \tag{2.83}$$

The eigenvectors of this tensor are the velocity vector $\boldsymbol{u}$ and the $d-1$ vectors $\boldsymbol{n}^{\perp}_{1,\ldots,d-1}$ that are perpendicular to the normal vector $\boldsymbol{n}$. The corresponding eigenvalues are

$$\frac{\partial \boldsymbol{F}(\boldsymbol{u}) \cdot \boldsymbol{n}}{\partial \boldsymbol{u}} \cdot \boldsymbol{n}^{\perp}_{1,\ldots,d-1} = (\boldsymbol{u} \cdot \boldsymbol{n})\, \boldsymbol{n}^{\perp}_{1,\ldots,d-1} + \boldsymbol{u} \left(\boldsymbol{n} \cdot \boldsymbol{n}^{\perp}_{1,\ldots,d-1} \right) = \underbrace{(\boldsymbol{u} \cdot \boldsymbol{n})}_{=\lambda_{1,\ldots,d-1}} \boldsymbol{n}^{\perp}_{1,\ldots,d-1} \ , \tag{2.84}$$

$$\frac{\partial \boldsymbol{F}(\boldsymbol{u}) \cdot \boldsymbol{n}}{\partial \boldsymbol{u}} \cdot \boldsymbol{u} = (\boldsymbol{u} \cdot \boldsymbol{n})\, \boldsymbol{u} + \boldsymbol{u} \left(\boldsymbol{n} \cdot \boldsymbol{u} \right) = \underbrace{2\,(\boldsymbol{u} \cdot \boldsymbol{n})}_{=\lambda_d} \boldsymbol{u} \ , \tag{2.85}$$

revealing $\lambda = \max_i |\lambda_i| = |2\,(\boldsymbol{u} \cdot \boldsymbol{n})\,|$. In the above equation, $\boldsymbol{u}_h^{\pm}$ is the local velocity evaluated in each quadrature point, while mean values of the velocity are used in Hesthaven and Warburton (2007), Klein et al. (2013), Shahbazi et al. (2007). A different value of $|\boldsymbol{u} \cdot \boldsymbol{n}|$ is also used in some works (Hesthaven and Warburton 2007, Xu et al. 2019). This value for the Lax–Friedrichs stabilization term turns out to be beneficial in terms of the maximum time step size according to the CFL condition, enlarging the stability region as compared to the larger value derived from the maximum eigenvalue of the flux Jacobian. Therefore, an additional parameter ζ_{LF} is introduced, to obtain $\lambda = 2\zeta_{\mathrm{LF}}|\boldsymbol{u} \cdot \boldsymbol{n}|$, where a value of $\zeta_{\mathrm{LF}} \approx \frac{1}{2}$ appears to be the sweet spot in terms of the maximum possible time step size. Finally, note that also different formulations such as $\lambda = 2\|\boldsymbol{u}\|$ are for example used in Marek et al. (2015). Boundary conditions are imposed by calculating exterior values $\boldsymbol{u}_h^+$ on Γ_h as defined in Table 2.3 according to the so-called mirror principle, see Hesthaven and Warburton (2007). In order to highlight that the convective term depends on the prescribed boundary data $\boldsymbol{g}_u$ on Dirichlet boundaries Γ_h^{D}, the notation $c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h; \boldsymbol{g}_u\right)$ is used.

For implicit formulations of the convective term, with the resulting non-linear system of equations solved by a Newton–Krylov approach, the linearization of the convective term is required

$$c^e_{h,\mathrm{lin}}(\boldsymbol{v}_h, \boldsymbol{u}_{h,\mathrm{lin}}, \Delta\boldsymbol{u}_h) = \left. \frac{\partial c_h^e(\boldsymbol{v}_h, \boldsymbol{u}_h)}{\partial \boldsymbol{u}_h} \right|_{\boldsymbol{u}_{h,\mathrm{lin}}} \cdot \Delta\boldsymbol{u}_h \ , \tag{2.86}$$

where $\boldsymbol{u}_{h,\mathrm{lin}}$ is the point of linearization and $\Delta\boldsymbol{u}_h$ the solution increment. To derive the linearization, the numerical flux definition (2.81) and the convective flux $\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) = \boldsymbol{u} \otimes \boldsymbol{u}$ are inserted into equation (2.80)

$$c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h\right) = -\left(\nabla \boldsymbol{v}_h, \boldsymbol{u}_h \otimes \boldsymbol{u}_h\right)_{\Omega_e} + \left(\boldsymbol{v}_h, \{\{\boldsymbol{u}_h \otimes \boldsymbol{u}_h\}\} \cdot \boldsymbol{n} + \frac{\Lambda}{2} [\![\boldsymbol{u}_h]\!] \cdot \boldsymbol{n} \right)_{\partial\Omega_e} .$$

In the following, it is assumed that Λ does not depend on $\boldsymbol{u}_h$ when linearizing the convective term. This asumption is motivated by the estimate $\|[\![\boldsymbol{u}_h]\!] \cdot \boldsymbol{n}\| \ll |\boldsymbol{u}_h \cdot \boldsymbol{n}|$. Accordingly, the term that is neglected as a consequence of this assumption is significantly smaller than the other terms. The linearization of the convective term is then given as

$$\begin{aligned} c^e_{h,\mathrm{lin}}(\boldsymbol{v}_h, \boldsymbol{u}_{h,\mathrm{lin}}, \Delta\boldsymbol{u}_h) = &- (\nabla\boldsymbol{v}_h, \boldsymbol{u}_{h,\mathrm{lin}} \otimes \Delta\boldsymbol{u}_h + \Delta\boldsymbol{u}_h \otimes \boldsymbol{u}_{h,\mathrm{lin}})_{\Omega_e} \\ &+ \left(\boldsymbol{v}_h, (\{\{\boldsymbol{u}_{h,\mathrm{lin}} \otimes \Delta\boldsymbol{u}_h\}\} + \{\{\Delta\boldsymbol{u}_h \otimes \boldsymbol{u}_{h,\mathrm{lin}}\}\}) \cdot \boldsymbol{n} + \frac{\Lambda}{2}[\![\Delta\boldsymbol{u}_h]\!] \cdot \boldsymbol{n}\right)_{\partial\Omega_e} , \end{aligned} \tag{2.87}$$

where $\Lambda = 2\zeta_{\mathrm{LF}} \max\left(|\boldsymbol{u}^-_{h,\mathrm{lin}} \cdot \boldsymbol{n}|, |\boldsymbol{u}^+_{h,\mathrm{lin}} \cdot \boldsymbol{n}|\right)$. On domain boundaries, exterior values for $\boldsymbol{u}_{h,\mathrm{lin}}$ and $\Delta\boldsymbol{u}_h$ are calculated as

$$\boldsymbol{u}^+_{h,\mathrm{lin}} = \begin{cases} -\boldsymbol{u}^-_{h,\mathrm{lin}} + 2\boldsymbol{g}_u & \text{on } \Gamma^{\mathrm{D}}_h \\ +\boldsymbol{u}^-_{h,\mathrm{lin}} & \text{on } \Gamma^{\mathrm{N}}_h \end{cases} , \quad \Delta\boldsymbol{u}^+_h = \begin{cases} -\Delta\boldsymbol{u}^-_h & \text{on } \Gamma^{\mathrm{D}}_h \\ +\Delta\boldsymbol{u}^-_h & \text{on } \Gamma^{\mathrm{N}}_h \end{cases} . \tag{2.88}$$

2.4.2.1.2 Convective formulation The starting point is the convective formulation of the convective term $(\boldsymbol{w}_h \cdot \nabla)\,\boldsymbol{u}_h$, i.e., the field $\boldsymbol{u}_h$ is transported by the velocity $\boldsymbol{w}_h$ (Oseen equation). For the time being, assume that $\boldsymbol{w}_h$ is a velocity field that is continuous across elements. Integration by parts yields

$$\begin{aligned} (\boldsymbol{v}_h, (\boldsymbol{w}_h \cdot \nabla)\,\boldsymbol{u}_h)_{\Omega_e} &= (\boldsymbol{v}_h, (\nabla\boldsymbol{u}_h) \cdot \boldsymbol{w}_h)_{\Omega_e} = (\boldsymbol{v}_h \otimes \boldsymbol{w}_h, \nabla\boldsymbol{u}_h)_{\Omega_e} \\ &= -(\nabla \cdot (\boldsymbol{v}_h \otimes \boldsymbol{w}_h), \boldsymbol{u}_h)_{\Omega_e} + (\boldsymbol{v}_h \otimes \boldsymbol{w}_h, \boldsymbol{u}^*_h \otimes \boldsymbol{n})_{\partial\Omega_e} \\ &= -(\nabla \cdot (\boldsymbol{v}_h \otimes \boldsymbol{w}_h), \boldsymbol{u}_h)_{\Omega_e} + (\boldsymbol{v}_h, (\boldsymbol{w}_h \cdot \boldsymbol{n})\,\boldsymbol{u}^*_h)_{\partial\Omega_e} . \end{aligned} \tag{2.89}$$

In accordance with Hesthaven and Warburton (2007), this formulation is denoted as weak formulation

$$\begin{aligned} c^e_{h,\mathrm{weak}}(\boldsymbol{v}_h, \boldsymbol{w}_h, \boldsymbol{u}_h) &= -(\nabla \cdot (\boldsymbol{v}_h \otimes \boldsymbol{w}_h), \boldsymbol{u}_h)_{\Omega_e} + (\boldsymbol{v}_h, (\boldsymbol{w}_h \cdot \boldsymbol{n})\,\boldsymbol{u}^*_h)_{\partial\Omega_e} \\ &= -(\nabla\boldsymbol{v}_h, \boldsymbol{u}_h \otimes \boldsymbol{w}_h)_{\Omega_e} - (\boldsymbol{v}_h, (\nabla \cdot \boldsymbol{w}_h)\,\boldsymbol{u}_h)_{\Omega_e} \\ &\quad + (\boldsymbol{v}_h, (\boldsymbol{w}_h \cdot \boldsymbol{n})\,\boldsymbol{u}^*_h)_{\partial\Omega_e} . \end{aligned} \tag{2.90}$$

The strong formulation $c^e_{h,\mathrm{strong}} = c^e_h$ is obtained by performing integration by parts once again, without inserting a numerical flux $\boldsymbol{u}^*_h$

$$\begin{aligned} c^e_h(\boldsymbol{v}_h, \boldsymbol{w}_h, \boldsymbol{u}_h) = &+ (\boldsymbol{v}_h, (\nabla\boldsymbol{u}_h) \cdot \boldsymbol{w}_h)_{\Omega_e} - (\boldsymbol{v}_h, (\boldsymbol{w}_h \cdot \boldsymbol{n})\,\boldsymbol{u}_h)_{\partial\Omega_e} \\ &+ (\boldsymbol{v}_h, (\boldsymbol{w}_h \cdot \boldsymbol{n})\,\boldsymbol{u}^*_h)_{\partial\Omega_e} . \end{aligned} \tag{2.91}$$

Using an upwind flux as numerical flux function $\boldsymbol{u}^*_h$,

$$\boldsymbol{u}^*_h = \{\{\boldsymbol{u}_h\}\} + \frac{1}{2}\mathrm{sign}\,(\boldsymbol{w_h} \cdot \boldsymbol{n})\,[\boldsymbol{u}_h] , \tag{2.92}$$

yields

$$\begin{aligned} c^e_h(\boldsymbol{v}_h, \boldsymbol{w}_h, \boldsymbol{u}_h) = &+ (\boldsymbol{v}_h, (\nabla\boldsymbol{u}_h) \cdot \boldsymbol{w}_h)_{\Omega_e} - (\boldsymbol{v}_h, (\boldsymbol{w}_h \cdot \boldsymbol{n})\,\boldsymbol{u}_h)_{\partial\Omega_e} \\ &+ \left(\boldsymbol{v}_h, (\boldsymbol{w}_h \cdot \boldsymbol{n})\,\{\{\boldsymbol{u}_h\}\} + \frac{1}{2}|\boldsymbol{w}_h \cdot \boldsymbol{n}|\,[\boldsymbol{u}_h]\right)_{\partial\Omega_e} . \end{aligned} \tag{2.93}$$

The non-liner term of the Navier–Stokes equations is $(\boldsymbol{u}_h \cdot \nabla)\,\boldsymbol{u}_h$ instead of $(\boldsymbol{w}_h \cdot \nabla)\,\boldsymbol{u}_h$. Since the numerical velocity $\boldsymbol{u}_h$ is discontinuous between elements, $\boldsymbol{w}_h$ is replaced by $\{\{\boldsymbol{u}_h\}\}$ for all face integrals (Fehn et al. 2019a), to obtain

$$\begin{aligned} c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h\right) = &+ \left(\boldsymbol{v}_h, \left(\nabla \boldsymbol{u}_h\right) \cdot \boldsymbol{u}_h\right)_{\Omega_e} - \left(\boldsymbol{v}_h, \left(\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right) \boldsymbol{u}_h\right)_{\partial\Omega_e} \\ &+ \left(\boldsymbol{v}_h, \left(\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right) \{\{\boldsymbol{u}_h\}\} + \frac{1}{2}|\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}|\,[\boldsymbol{u}_h]\right)_{\partial\Omega_e} . \end{aligned} \tag{2.94}$$

Boundary conditions are imposed according to Table 2.3. Following the procedure shown above for the divergence formulation, the linearization of the convective formulation yields

$$\begin{aligned} c_{h,\mathrm{lin}}^e(\boldsymbol{v}_h, \boldsymbol{u}_{h,\mathrm{lin}}, \Delta\boldsymbol{u}_h) = &\left(\boldsymbol{v}_h, \left(\nabla \boldsymbol{u}_{h,\mathrm{lin}}\right) \cdot \Delta\boldsymbol{u}_h\right)_{\Omega_e} + \left(\boldsymbol{v}_h, \left(\nabla(\Delta\boldsymbol{u}_h)\right) \cdot \boldsymbol{u}_{h,\mathrm{lin}}\right)_{\Omega_e} \\ &- \left(\boldsymbol{v}_h, \left(\{\{\boldsymbol{u}_{h,\mathrm{lin}}\}\} \cdot \boldsymbol{n}\right) \Delta\boldsymbol{u}_h\right)_{\partial\Omega_e} - \left(\boldsymbol{v}_h, \left(\{\{\Delta\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right) \boldsymbol{u}_{h,\mathrm{lin}}\right)_{\partial\Omega_e} \\ &+ \left(\boldsymbol{v}_h, \left(\{\{\boldsymbol{u}_{h,\mathrm{lin}}\}\} \cdot \boldsymbol{n}\right) \{\{\Delta\boldsymbol{u}_h\}\} + \left(\{\{\Delta\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right) \{\{\boldsymbol{u}_{h,\mathrm{lin}}\}\}\right)_{\partial\Omega_e} \\ &+ \left(\boldsymbol{v}_h, \frac{1}{2}|\{\{\boldsymbol{u}_{h,\mathrm{lin}}\}\} \cdot \boldsymbol{n}|\,[\Delta\boldsymbol{u}_h]\right)_{\partial\Omega_e} . \end{aligned} \tag{2.95}$$

Remark 2.6 *Numerical investigations revealed that the strong formulation should be used to obtain optimal rates of convergence. For the weak formulation, sub-optimal rates of convergence of order k_u (instead of k_u+1) have been observed for the velocity. For this reason, only the strong formulation is investigated in this work.*

2.4.2.2 Velocity–pressure coupling terms

2.4.2.2.1 Velocity divergence term The DG formulation of the velocity divergence term is derived by performing integration by parts of $(q_h, \nabla \cdot \boldsymbol{u}_h)_{\Omega_e}$ and replacing the physical flux $\boldsymbol{u}_h$ by the numerical flux $\boldsymbol{u}_h^*$, to obtain

$$d_{h,\mathrm{weak}}^e\left(q_h, \boldsymbol{u}_h; \boldsymbol{g}_u\right) = -\left(\nabla q_h, \boldsymbol{u}_h\right)_{\Omega_e} + \left(q_h, \boldsymbol{u}_h^* \cdot \boldsymbol{n}\right)_{\partial\Omega_e} . \tag{2.96}$$

Using the central flux $\boldsymbol{u}_h^* = \{\{\boldsymbol{u}_h\}\}$ yields

$$d_{h,\mathrm{weak}}^e\left(q_h, \boldsymbol{u}_h\right) = -\left(\nabla q_h, \boldsymbol{u}_h\right)_{\Omega_e} + \left(q_h, \{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right)_{\partial\Omega_e} . \tag{2.97}$$

Under the assumption of an exact evaluation of integrals, the weak formulation is equivalent to the strong formulation

$$d_{h,\mathrm{strong}}^e\left(q_h, \boldsymbol{u}_h\right) = \left(q_h, \nabla \cdot \boldsymbol{u}_h\right)_{\Omega_e} - \left(q_h, \frac{1}{2}\left[\boldsymbol{u}_h\right] \cdot \boldsymbol{n}\right)_{\partial\Omega_e} . \tag{2.98}$$

Boundary conditions are imposed according to Table 2.3, so that the velocity divergence term can alternatively be written as

$$\begin{aligned} d_{h,\mathrm{weak}}^e\left(q_h, \boldsymbol{u}_h; \boldsymbol{g}_u\right) = &\underbrace{-\left(\nabla q_h, \boldsymbol{u}_h\right)_{\Omega_e} + \left(q_h, \{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right)_{\partial\Omega_e \setminus \Gamma_h} + \left(q_h, \boldsymbol{u}_h \cdot \boldsymbol{n}\right)_{\partial\Omega_e \cap \Gamma_h^{\mathrm{N}}}}_{= d_{h,\mathrm{weak,hom}}^e(q_h, \boldsymbol{u}_h)} \\ &\underbrace{+\left(q_h, \boldsymbol{g}_u \cdot \boldsymbol{n}\right)_{\partial\Omega_e \cap \Gamma_h^{\mathrm{D}}}}_{= d_{h,\mathrm{inhom}}^e(q_h; \boldsymbol{g}_u)} , \end{aligned} \tag{2.99}$$

where a decomposition into a homogeneous part (depending on the solution) and an inhomogeneous part (depending on inhomogeneous boundary data) has been introduced.

Remark 2.7 *As a reference formulation, consider the modified formulation of the velocity divergence term used in Hesthaven and Warburton (2007) in the context of the high-order dual splitting scheme*

$$d^e_{h,\mathrm{ref}}\left(q_h, \boldsymbol{u}_h\right) = \left(q_h, \nabla \cdot \boldsymbol{u}_h\right)_{\Omega_e} \ . \tag{2.100}$$

This formulation does not perform integration by parts as described in step (ii) above. Accordingly, this formulation does not depend on boundary conditions prescribed for the velocity.

2.4.2.2.2 Pressure gradient term The procedure detailed above for the velocity divergence term is applied to derive the DG formulation $g^e_h\left(\boldsymbol{v}_h, p_h\right)$ of the pressure gradient term

$$g^e_{h,\mathrm{weak}}\left(\boldsymbol{v}_h, p_h; g_p\right) = -\left(\nabla \cdot \boldsymbol{v}_h, p_h\right)_{\Omega_e} + \left(\boldsymbol{v}_h, p^*_h \boldsymbol{n}\right)_{\partial\Omega_e} \ . \tag{2.101}$$

Using again the central flux $p^*_h = \{\{p_h\}\}$ yields the weak formulation

$$g^e_{h,\mathrm{weak}}\left(\boldsymbol{v}_h, p_h; g_p\right) = -\left(\nabla \cdot \boldsymbol{v}_h, p_h\right)_{\Omega_e} + \left(\boldsymbol{v}_h, \{\{p_h\}\}\boldsymbol{n}\right)_{\partial\Omega_e} \ . \tag{2.102}$$

Under the assumption of an exact evaluation of integrals, the weak formulation is equivalent to the strong formulation

$$g^e_{h,\mathrm{strong}}\left(\boldsymbol{v}_h, p_h\right) = \left(\boldsymbol{v}_h, \nabla p_h\right)_{\Omega_e} - \left(\boldsymbol{v}_h, \frac{1}{2}\left[p_h\right]\boldsymbol{n}\right)_{\partial\Omega_e} \ . \tag{2.103}$$

By imposing boundary conditions according to Table 2.3, the pressure gradient term can be decomposed into homogeneous and inhomogeneous contributions

$$\begin{aligned} g^e_{h,\mathrm{weak}}\left(\boldsymbol{v}_h, p_h; g_p\right) = & \underbrace{-\left(\nabla \cdot \boldsymbol{v}_h, p_h\right)_{\Omega_e} + \left(\boldsymbol{v}_h, \{\{p_h\}\}\boldsymbol{n}\right)_{\partial\Omega_e \setminus \Gamma_h} + \left(\boldsymbol{v}_h, p_h \boldsymbol{n}\right)_{\partial\Omega_e \cap \Gamma^{\mathrm{D}}_h}}_{= g^e_{h,\mathrm{weak,hom}}\left(\boldsymbol{v}_h, p_h\right)} \\ & \underbrace{+\left(\boldsymbol{v}_h, g_p \boldsymbol{n}\right)_{\partial\Omega_e \cap \Gamma^{\mathrm{N}}_h}}_{= g^e_{h,\mathrm{inhom}}\left(\boldsymbol{v}_h; g_p\right)} \ . \end{aligned} \tag{2.104}$$

Remark 2.8 *As a reference formulation, consider the modified formulation of the pressure gradient term used in Hesthaven and Warburton (2007) in the context of the high-order dual splitting scheme*

$$g^e_{h,\mathrm{ref}}\left(\boldsymbol{v}_h, p_h\right) = \left(\boldsymbol{v}_h, \nabla p_h\right)_{\Omega_e} \ . \tag{2.105}$$

This formulation does not perform integration by parts as described in (ii) above. Accordingly, this formulation does not depend on boundary conditions prescribed for the pressure.

2.4.2.2.3 A note on the symmetry of the velocity–pressure coupling terms In order to ensure that the (non-)linear system of equations (2.76) and (2.77) is symmetric with respect to the pressure gradient term and velocity divergence term, a negative sign has been introduced in front of the velocity divergence term in the weak formulation, $-d_h^e(q_h, \boldsymbol{u}_h)$. This can be verified by comparing the homogeneous part of the weak formulations of the pressure gradient term and the velocity divergence term. In a first step, the velocity divergence term in weak formulation is integrated by parts once again to obtain the strong formulation

$$d_{h,\mathrm{strong,hom}}^e(q_h, \boldsymbol{u}_h) = (q_h, \nabla \cdot \boldsymbol{u}_h)_{\Omega_e} - \left(q_h \boldsymbol{n}, \frac{1}{2}[\![\boldsymbol{u}_h]\!] \cdot \boldsymbol{n} \right)_{\partial\Omega_e \setminus \Gamma_h} - (q_h \boldsymbol{n}, \boldsymbol{u}_h)_{\partial\Omega_e \cap \Gamma_h^{\mathrm{D}}} .$$

While the previous derivations used an elementwise formulation, a global formulation that performs integrals over all elements and all faces is introduced by adding the contributions from both elements Ω_{e^-} and Ω_{e^+} adjacent to an interior face f

$$- \left(q_h^- \boldsymbol{n}^-, \frac{1}{2}[\![\boldsymbol{u}_h]\!] \cdot \boldsymbol{n}^- \right)_{\partial\Omega_{e^-} \cap f} - \left(q_h^+ \boldsymbol{n}^+, \frac{1}{2}[\![\boldsymbol{u}_h]\!] \cdot \boldsymbol{n}^+ \right)_{\partial\Omega_{e^+} \cap f} = - \left(\{\{q_h\}\} \boldsymbol{n}, [\![\boldsymbol{u}_h]\!] \cdot \boldsymbol{n} \right)_f$$

Accordingly, the homogeneous part of the velocity divergence term can be written as

$$d_{h,\mathrm{strong,hom}}(q_h, \boldsymbol{u}_h) = (q_h, \nabla \cdot \boldsymbol{u}_h)_{\Omega_h} - (\{\{q_h\}\} \boldsymbol{n}, [\![\boldsymbol{u}_h]\!] \cdot \boldsymbol{n})_{\Gamma_h^{\mathrm{int}}} - (q_h \boldsymbol{n}, \boldsymbol{u}_h)_{\Gamma_h^{\mathrm{D}}} . \tag{2.106}$$

For the pressure gradient term, one obtains

$$\left(\boldsymbol{v}_h^-, \{\{p_h\}\} \boldsymbol{n}^- \right)_{\partial\Omega_{e^-} \cap f} + \left(\boldsymbol{v}_h^+, \{\{p_h\}\} \boldsymbol{n}^+ \right)_{\partial\Omega_{e^+} \cap f} = \left([\![\boldsymbol{v}_h]\!] \cdot \boldsymbol{n}, \{\{p_h\}\} \boldsymbol{n} \right)_f$$

and

$$g_{h,\mathrm{weak,hom}}(\boldsymbol{v}_h, p_h) = - (\nabla \cdot \boldsymbol{v}_h, p_h)_{\Omega_h} + ([\![\boldsymbol{v}_h]\!] \cdot \boldsymbol{n}, \{\{p_h\}\} \boldsymbol{n})_{\Gamma_h^{\mathrm{int}}} + (\boldsymbol{v}_h, p_h \boldsymbol{n})_{\Gamma_h^{\mathrm{D}}} . \tag{2.107}$$

A comparison of equation (2.106) and equation (2.107) shows that symmetry of both terms is obtained if $-d_h^e(q_h, \boldsymbol{u}_h)$ is used in the continuity equation. While the weak and strong formulations are equivalent in case of exact integration, only weak–strong combinations of the velocity–pressure coupling terms ($d_{h,\mathrm{strong}}$ with $g_{h,\mathrm{weak}}$, or $d_{h,\mathrm{weak}}$ with $g_{h,\mathrm{strong}}$) yield exact symmetry in the presence of quadrature errors. The above equations also reveal that exterior values on domain boundaries have to be chosen according to Table 2.3 to ensure symmetry.

2.4.2.3 Negative Laplace operator

The DG formulations derived above have in common that the underlying operators only involve derivatives of first order. Another important class are operators with second derivatives, such as viscous or diffusive terms in computational fluid dynamics. Projection methods for the incompressible Navier–Stokes equations lead to a Poisson equation for the pressure, another example of a PDE with second derivatives. This section discusses the DG formulation of the Laplace operator. Consider the Poisson-type model problem

$$-\nabla^2 u = f \ \text{ in } \Omega , \tag{2.108}$$

subject to boundary conditions

$$u = g \ \text{ on } \Gamma^{\mathrm{D}} \ , \tag{2.109}$$

$$\nabla u \cdot \boldsymbol{n} = h \ \text{ on } \Gamma^{\mathrm{N}} \ . \tag{2.110}$$

Various DG methods have been formulated for this problem and the reader is referred to Arnold et al. (2002) for a comprehensive overview. The derivation shown here follows the standard procedure of decomposing the problem into a system of equations with first derivatives, see also Hesthaven and Warburton (2007), to which integration by parts is applied with suitable numerical fluxes. This system of equations is then combined to an equation in the primal variable only, the so-called primal formulation. In terms of numerical fluxes, the present work exclusively studies the symmetric interior penalty Galerkin (SIPG) method (Arnold 1982, Arnold et al. 2000, 2002). Rewriting equation (2.108) as a system of first-order equations yields

$$-\nabla \cdot \boldsymbol{\sigma} = f \ , \tag{2.111}$$

$$\boldsymbol{\sigma} = \nabla u \ , \tag{2.112}$$

where the auxiliary variable $\boldsymbol{\sigma}$ has been introduced. The exact solution is approximated by $u_h \in \mathcal{V}_h$ and $\boldsymbol{\sigma}_h \in \mathcal{V}_h^d$, where the space of test and trial functions $\mathcal{V}_h$ is defined as

$$\mathcal{V}_h = \left\{ u_h \in L_2(\Omega_h) \ : \ u_h\left(\boldsymbol{x}^e(\boldsymbol{\xi})\right)|_{\Omega_e} = \tilde{u}_h^e(\boldsymbol{\xi})|_{\tilde{\Omega}_e} \in \mathcal{V}_{h,e} = \mathcal{Q}_k(\tilde{\Omega}_e) \ \forall e \right\} \ . \tag{2.113}$$

Replacing u and $\boldsymbol{\sigma}$ in equation (2.111) and equation (2.112) by the approximate solutions u_h and $\boldsymbol{\sigma}_h$, multiplying the residuals by test functions $v_h \in \mathcal{V}_h$ and $\boldsymbol{\tau}_h \in \mathcal{V}_h^d$, and integration over element Ω_e yields

$$\left(v_h, -\nabla \cdot \boldsymbol{\sigma}_h\right)_{\Omega_e} = \left(v_h, f\right)_{\Omega_e} \ , \tag{2.114}$$

$$\left(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h\right)_{\Omega_e} = \left(\boldsymbol{\tau}_h, \nabla u_h\right)_{\Omega_e} \ . \tag{2.115}$$

The weak formulation of the problem written in flux formulation is obtained by integrating equations (2.114) and (2.115) by parts and introducing numerical flux functions u_h^* and $\boldsymbol{\sigma}_h^*$

$$\left(\nabla v_h, \boldsymbol{\sigma}_h\right)_{\Omega_e} - \left(v_h, \boldsymbol{\sigma}_h^* \cdot \boldsymbol{n}\right)_{\partial\Omega_e} = \left(v_h, f\right)_{\Omega_e} \ , \tag{2.116}$$

$$\left(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h\right)_{\Omega_e} = -\left(\nabla \cdot \boldsymbol{\tau}_h, u_h\right)_{\Omega_e} + \left(\boldsymbol{\tau}_h, u_h^* \boldsymbol{n}\right)_{\partial\Omega_e} \ . \tag{2.117}$$

The primal formulation is obtained by eliminating the auxiliary variable $\boldsymbol{\sigma}_h$. The right-hand side of equation (2.117) is first integrated by parts once again

$$\left(\boldsymbol{\tau}_h, \boldsymbol{\sigma}_h\right)_{\Omega_e} = \left(\boldsymbol{\tau}_h, \nabla u_h\right)_{\Omega_e} - \left(\boldsymbol{\tau}_h, \left(u_h - u_h^*\right)\boldsymbol{n}\right)_{\partial\Omega_e} \ . \tag{2.118}$$

Using $\boldsymbol{\tau}_h = \nabla v_h$ and inserting equation (2.118) into equation (2.116) yields the primal formulation: Find $u_h \in \mathcal{V}_h$ such that

$$l_h^e\left(v_h, u_h\right) = \left(\nabla v_h, \nabla u_h\right)_{\Omega_e} - \left(\nabla v_h, \left(u_h - u_h^*\right)\boldsymbol{n}\right)_{\partial\Omega_e} - \left(v_h, \boldsymbol{\sigma}_h^* \cdot \boldsymbol{n}\right)_{\partial\Omega_e} = \left(v_h, f\right)_{\Omega_e} \ , \tag{2.119}$$

for all $v_h \in \mathcal{V}_{h,e}$ and all elements Ω_e, $e = 1, ..., N_{\mathrm{el}}$. For the SIPG method (Arnold 1982, Arnold et al. 2000, 2002), the numerical fluxes are defined as

$$u_h^* = \{\{u_h\}\} \,, \tag{2.120}$$

$$\boldsymbol{\sigma}_h^* = \{\{\nabla u_h\}\} - \tau [\![u_h]\!] \,. \tag{2.121}$$

The penalty parameter of the SIPG method is denoted by τ and has to be large enough to ensure coercivity of the bilinear form. Essentially, the penalty parameter depends on the polynomial degree k and a characteristic element length h. An explicit expression for the penalty parameter of the SIPG method is derived in Shahbazi (2005) for triangular/tetrahedral elements and in Hillewaert (2013) for other element geometries. For quadrilateral/hexahedral elements the penalty parameter τ_e associated to element e is defined as (Hillewaert 2013)

$$\tau_e = (k+1)^2 \frac{A\left(\partial\Omega_e \setminus \Gamma_h\right)/2 + A\left(\partial\Omega_e \cap \Gamma_h\right)}{V\left(\Omega_e\right)} \,, \tag{2.122}$$

with the element volume $V\left(\Omega_e\right) = \int_{\Omega_e} \mathrm{d}\Omega$ and the surface area $A(f) = \int_{f \subset \partial\Omega_e} \mathrm{d}\Gamma$. Conservativity of the numerical flux definition is fulfilled by choosing the maximum value from both elements adjacent to an interior face f according to

$$\tau = \begin{cases} \max\left(\tau_{e^-}, \tau_{e^+}\right) & \text{if face } f \subseteq \partial\Omega_e \setminus \Gamma_h \,, \\ \tau_e & \text{if face } f \subseteq \partial\Omega_e \cap \Gamma_h \,. \end{cases} \tag{2.123}$$

Inserting the numerical fluxes (2.120) and (2.121) into equation (2.119) yields

$$\begin{aligned} l_h^e\left(v_h, u_h\right) = &+ \left(\nabla v_h, \nabla u_h\right)_{\Omega_e} - \left(\nabla v_h, \frac{1}{2}[\![u_h]\!]\right)_{\partial\Omega_e} - \left(v_h, \{\{\nabla u_h\}\} \cdot \boldsymbol{n}\right)_{\partial\Omega_e} \\ &+ \left(v_h, \tau [\![u_h]\!] \cdot \boldsymbol{n}\right)_{\partial\Omega_e} \,. \end{aligned} \tag{2.124}$$

To complete the formulation, exterior values have to be prescribed on Γ_h in order to weakly impose boundary conditions. Since equation (2.108) contains second derivatives, both the solution u_h and the gradient in normal direction $\nabla u_h \cdot \boldsymbol{n}$ have to be prescribed

$$u_h^+ = \begin{cases} -u_h^- + 2g & \text{on } \Gamma_h^{\mathrm{D}} \\ +u_h^- & \text{on } \Gamma_h^{\mathrm{N}} \end{cases} \,, \quad \nabla u_h^+ \cdot \boldsymbol{n} = \begin{cases} +\nabla u_h^- \cdot \boldsymbol{n} & \text{on } \Gamma_h^{\mathrm{D}} \\ -\nabla u_h^- \cdot \boldsymbol{n} + 2h & \text{on } \Gamma_h^{\mathrm{N}} \end{cases} \,. \tag{2.125}$$

By inserting the boundary conditions from equation (2.125) into equation (2.124), the weak formulation of the Laplace operator l_h^e can be separated into a homogeneous part $l_{h,\mathrm{hom}}^e$ and an inhomogeneous part $l_{h,\mathrm{inhom}}^e$

$$l_h^e\left(v_h, u_h; g, h\right) = l_{h,\mathrm{hom}}^e(v_h, u_h) + l_{h,\mathrm{inhom}}^e(v_h; g, h) \,, \tag{2.126}$$

where

$$\begin{aligned} l_{h,\mathrm{hom}}^e(v_h, u_h) = &+ \left(\nabla v_h, \nabla u_h\right)_{\Omega_e} \\ &- \left(\nabla v_h, \frac{1}{2}[\![u_h]\!]\right)_{\partial\Omega_e \setminus \Gamma_h} - \left(\nabla v_h, u_h \boldsymbol{n}\right)_{\partial\Omega_e \cap \Gamma_h^{\mathrm{D}}} \\ &- \left(v_h, \{\{\nabla u_h\}\} \cdot \boldsymbol{n}\right)_{\partial\Omega_e \setminus \Gamma_h} - \left(v_h, \nabla u_h \cdot \boldsymbol{n}\right)_{\partial\Omega_e \cap \Gamma_h^{\mathrm{D}}} \\ &+ \left(v_h, \tau [\![u_h]\!] \cdot \boldsymbol{n}\right)_{\partial\Omega_e \setminus \Gamma_h} + \left(v_h, 2\tau u_h\right)_{\partial\Omega_e \cap \Gamma_h^{\mathrm{D}}} \end{aligned} \tag{2.127}$$

and

$$l^e_{h,\mathrm{inhom}}(v_h; g, h) = +(\nabla v_h, g\boldsymbol{n})_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} - (v_h, h)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{N}}} - (v_h, 2\tau g)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} \,. \tag{2.128}$$

Then, the weak formulation of problem (2.108) reads: Find $u_h \in \mathcal{V}_h$ such that

$$l^e_{h,\mathrm{hom}}(v_h, u_h) = (v_h, f)_{\Omega_e} - l^e_{h,\mathrm{inhom}}(v_h; g, h) \;\; \forall v_h \in \mathcal{V}_{h,e} \tag{2.129}$$

and for all elements Ω_e, $e = 1, ..., N_{\mathrm{el}}$. Terms arising from inhomogeneous boundary conditions have been shifted to the right-hand side of the equation.

2.4.2.4 Viscous term

This section derives DG formulations for the Laplace formulation and divergence formulation of the viscous term.

2.4.2.4.1 Laplace formulation The viscous operator represents a generalization of the Laplace operator to vectorial quantities with the viscosity ν as a scaling factor. In analogy to the Poisson problem considered in Section 2.4.2.3, the primal formulation of the viscous term is given as

$$\begin{aligned} v_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h, \boldsymbol{g}_u, \boldsymbol{h}_u\right) = &+ \left(\nabla\boldsymbol{v}_h, \nu\nabla\boldsymbol{u}_h\right)_{\Omega_e} - \left(\nabla\boldsymbol{v}_h, \nu\left(\boldsymbol{u}_h - \boldsymbol{u}_h^*\right)\otimes\boldsymbol{n}\right)_{\partial\Omega_e} \\ &- \left(\boldsymbol{v}_h, \boldsymbol{F}^*_{\mathrm{v},h}\cdot\boldsymbol{n}\right)_{\partial\Omega_e} \,. \end{aligned} \tag{2.130}$$

For the symmetric interior penalty Galerkin (SIPG) method, the numerical fluxes are defined as

$$\boldsymbol{u}_h^* = \{\{\boldsymbol{u}_h\}\} \,, \tag{2.131}$$

$$\boldsymbol{F}^*_{\mathrm{v},h} = \nu\{\{\nabla\boldsymbol{u}_h\}\} - \nu\tau[\![\boldsymbol{u}_h]\!] \,, \tag{2.132}$$

where the interior penalty parameter is defined as in equation (2.123) and equation (2.122). Again, boundary conditions are incorporated into the formulation by defining exterior values for the velocity $\boldsymbol{u}_h^+$ and the velocity gradient in normal direction $\nabla\boldsymbol{u}_h^+\cdot\boldsymbol{n}$. Inserting the numerical fluxes (2.131) and (2.132) into equation (2.130) yields

$$\begin{aligned} v_h^e(\boldsymbol{v}_h, \boldsymbol{u}_h) = &+ \left(\nabla\boldsymbol{v}_h, \nu\nabla\boldsymbol{u}_h\right)_{\Omega_e} - \left(\nabla\boldsymbol{v}_h, \frac{\nu}{2}[\![\boldsymbol{u}_h]\!]\right)_{\partial\Omega_e} \\ &- \left(\boldsymbol{v}_h, \nu\{\{\nabla\boldsymbol{u}_h\}\}\cdot\boldsymbol{n}\right)_{\partial\Omega_e} + \left(\boldsymbol{v}_h, \nu\tau[\![\boldsymbol{u}_h]\!]\cdot\boldsymbol{n}\right)_{\partial\Omega_e} \,. \end{aligned} \tag{2.133}$$

By inserting boundary conditions according to Table 2.3 into equation (2.133), the viscous term can be decomposed into

$$v_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h; \boldsymbol{g}_u, \boldsymbol{h}_u\right) = v^e_{h,\mathrm{hom}}(\boldsymbol{v}_h, \boldsymbol{u}_h) + v^e_{h,\mathrm{inhom}}(\boldsymbol{v}_h; \boldsymbol{g}_u, \boldsymbol{h}_u) \,, \tag{2.134}$$

where the homogeneous part is

$$\begin{aligned} v^e_{h,\mathrm{hom}}(\boldsymbol{v}_h, \boldsymbol{u}_h) = &+ \left(\nabla\boldsymbol{v}_h, \nu\nabla\boldsymbol{u}_h\right)_{\Omega_e} \\ &- \left(\nabla\boldsymbol{v}_h, \frac{\nu}{2}[\![\boldsymbol{u}_h]\!]\right)_{\partial\Omega_e\setminus\Gamma_h} - \left(\nabla\boldsymbol{v}_h, \nu\,\boldsymbol{u}_h\otimes\boldsymbol{n}\right)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} \\ &- \left(\boldsymbol{v}_h, \nu\{\{\nabla\boldsymbol{u}_h\}\}\cdot\boldsymbol{n}\right)_{\partial\Omega_e\setminus\Gamma_h} - \left(\boldsymbol{v}_h, \nu\,\nabla\boldsymbol{u}_h\cdot\boldsymbol{n}\right)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} \\ &+ \left(\boldsymbol{v}_h, \nu\tau[\![\boldsymbol{u}_h]\!]\cdot\boldsymbol{n}\right)_{\partial\Omega_e\setminus\Gamma_h} + \left(\boldsymbol{v}_h, 2\nu\tau\boldsymbol{u}_h\right)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} \end{aligned} \tag{2.135}$$

and the inhomogeneous part

$$\begin{aligned} v^e_{h,\mathrm{inhom}}(\boldsymbol{v}_h, \boldsymbol{g}_u, \boldsymbol{h}_u) = &+ (\nabla \boldsymbol{v}_h, \nu\, \boldsymbol{g}_u \otimes \boldsymbol{n})_{\partial\Omega_e \cap \Gamma_h^{\mathrm{D}}} - (\boldsymbol{v}_h, \boldsymbol{h}_u)_{\partial\Omega_e \cap \Gamma_h^{\mathrm{N}}} \\ &- (\boldsymbol{v}_h, 2\nu\tau \boldsymbol{g}_u)_{\partial\Omega_e \cap \Gamma_h^{\mathrm{D}}} \ . \end{aligned} \tag{2.136}$$

2.4.2.4.2 Divergence formulation This section discusses the DG discretization of the viscous term written in divergence formulation by extracting the model problem

$$\begin{aligned} -\nabla \cdot \boldsymbol{F}_{\mathrm{v}} &= \boldsymbol{f} \ , \\ \boldsymbol{F}_{\mathrm{v}} &= \nu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathsf{T}} \right) \ . \end{aligned} \tag{2.137}$$

The starting point is to split the viscous flux into two contributions $\boldsymbol{F}_{1,\mathrm{v}} + \boldsymbol{F}_{2,\mathrm{v}} = \boldsymbol{F}_{\mathrm{v}}$, in order to rewrite the system of first-order equations (2.137) in an alternative way

$$\begin{aligned} -\nabla \cdot \boldsymbol{F}_{1,\mathrm{v}} - \nabla \cdot \boldsymbol{F}_{2,\mathrm{v}} &= \boldsymbol{f} \ , \\ \boldsymbol{F}_{1,\mathrm{v}} &= \nu \nabla \boldsymbol{u} \ , \\ \boldsymbol{F}_{2,\mathrm{v}} &= \nu \left(\nabla \boldsymbol{u} \right)^{\mathsf{T}} \ . \end{aligned} \tag{2.138}$$

The equations are first multiplied by weighting functions and integrated over one element

$$(\boldsymbol{v}_h, -\nabla \cdot \boldsymbol{F}_{1,\mathrm{v},h})_{\Omega_e} + (\boldsymbol{v}_h, -\nabla \cdot \boldsymbol{F}_{2,\mathrm{v},h})_{\Omega_e} = (\boldsymbol{v}_h, \boldsymbol{f})_{\Omega_e} \ , \tag{2.139}$$

$$(\boldsymbol{w}_h, \boldsymbol{F}_{1,\mathrm{v},h})_{\Omega_e} = (\boldsymbol{w}_h, \nu \nabla \boldsymbol{u}_h)_{\Omega_e} \ , \tag{2.140}$$

$$\begin{aligned} (\boldsymbol{w}_h, \boldsymbol{F}_{2,\mathrm{v},h})_{\Omega_e} &= \left(\boldsymbol{w}_h, \nu \left(\nabla \boldsymbol{u}_h \right)^{\mathsf{T}} \right)_{\Omega_e} \\ &= \left(\boldsymbol{w}_h^{\mathsf{T}}, \nu \nabla \boldsymbol{u}_h \right)_{\Omega_e} \ . \end{aligned} \tag{2.141}$$

Then, following the procedure described in Section 2.4.2.3 yields

$$\begin{aligned} (\nabla \boldsymbol{v}_h, \boldsymbol{F}_{1,\mathrm{v}}(\boldsymbol{u}_h))_{\Omega_e} + (\nabla \boldsymbol{v}_h, \boldsymbol{F}_{2,\mathrm{v}}(\boldsymbol{u}_h))_{\Omega_e} - \left(\boldsymbol{v}_h, \boldsymbol{F}^*_{1,\mathrm{v},h} \cdot \boldsymbol{n} \right)_{\partial\Omega_e} \\ - \left(\boldsymbol{v}_h, \boldsymbol{F}^*_{2,\mathrm{v},h} \cdot \boldsymbol{n} \right)_{\partial\Omega_e} = (\boldsymbol{v}_h, \boldsymbol{f})_{\Omega_e} \ , \end{aligned} \tag{2.142}$$

and

$$(\boldsymbol{w}_h, \boldsymbol{F}_{1,\mathrm{v},h})_{\Omega_e} = (\boldsymbol{w}_h, \nu \nabla \boldsymbol{u}_h)_{\Omega_e} - (\boldsymbol{w}_h, \nu \left(\boldsymbol{u}_h - \boldsymbol{u}^*_h \right) \otimes \boldsymbol{n})_{\partial\Omega_e} \ , \tag{2.143}$$

and

$$\begin{aligned} (\boldsymbol{w}_h, \boldsymbol{F}_{2,\mathrm{v},h})_{\Omega_e} &= \left(\boldsymbol{w}_h^{\mathsf{T}}, \nu \nabla \boldsymbol{u}_h \right)_{\Omega_e} - \left(\boldsymbol{w}_h^{\mathsf{T}}, \nu \left(\boldsymbol{u}_h - \boldsymbol{u}^*_h \right) \otimes \boldsymbol{n} \right)_{\partial\Omega_e} = \\ &= \left(\boldsymbol{w}_h, \nu \left(\nabla \boldsymbol{u}_h \right)^{\mathsf{T}} \right)_{\Omega_e} - \left(\boldsymbol{w}_h, \nu \left(\left(\boldsymbol{u}_h - \boldsymbol{u}^*_h \right) \otimes \boldsymbol{n} \right)^{\mathsf{T}} \right)_{\partial\Omega_e} \ . \end{aligned} \tag{2.144}$$

The primal formulation is obtained by setting $\boldsymbol{w}_h = \nabla \boldsymbol{v}_h$ and inserting equations (2.143) and (2.144) into equation (2.142)

$$\begin{aligned} v^e_h(\boldsymbol{v}_h, \boldsymbol{u}_h) = &+ \left(\nabla \boldsymbol{v}_h, \nu \left(\nabla \boldsymbol{u}_h + (\nabla \boldsymbol{u}_h)^{\mathsf{T}} \right) \right)_{\Omega_e} \\ &- \left(\nabla \boldsymbol{v}_h, \nu \left((\boldsymbol{u}_h - \boldsymbol{u}^*_h) \otimes \boldsymbol{n} + \left((\boldsymbol{u}_h - \boldsymbol{u}^*_h) \otimes \boldsymbol{n} \right)^{\mathsf{T}} \right) \right)_{\partial\Omega_e} \\ &- \left(\boldsymbol{v}_h, \left(\boldsymbol{F}^*_{1,\mathrm{v},h} + \boldsymbol{F}^*_{2,\mathrm{v},h} \right) \cdot \boldsymbol{n} \right)_{\partial\Omega_e} = (\boldsymbol{v}_h, \boldsymbol{f})_{\Omega_e} \ . \end{aligned} \tag{2.145}$$

The symmetric interior penalty fluxes are chosen for the standard Laplace term $\boldsymbol{F}_{1,\mathrm{v}}$

$$\boldsymbol{u}_h^* = \{\{\boldsymbol{u}_h\}\} \,, \tag{2.146}$$

$$\boldsymbol{F}_{1,\mathrm{v},h}^* = \nu\{\{\nabla \boldsymbol{u}_h\}\} - \nu\tau[\![\boldsymbol{u}_h]\!] \,, \tag{2.147}$$

and due to $\boldsymbol{F}_{2,\mathrm{v}} = \boldsymbol{F}_{1,\mathrm{v}}^{\mathsf{T}}$, the following flux is used for $\boldsymbol{F}_{2,\mathrm{v}}$

$$\boldsymbol{F}_{2,\mathrm{v},h}^* = \left(\boldsymbol{F}_{1,\mathrm{v},h}^*\right)^{\mathsf{T}} = \nu\{\{(\nabla \boldsymbol{u}_h)^{\mathsf{T}}\}\} - \nu\tau[\![\boldsymbol{u}_h]\!]^{\mathsf{T}} \,. \tag{2.148}$$

The interior penalty parameter is defined as in equation (2.123) and equation (2.122). Inserting the numerical fluxes, equations (2.146), (2.147), and (2.148), into equation (2.145) yields

$$\begin{aligned} v_h^e(\boldsymbol{v}_h, \boldsymbol{u}_h) = &+ \left(\nabla \boldsymbol{v}_h, \nu\left(\nabla \boldsymbol{u}_h + (\nabla \boldsymbol{u}_h)^{\mathsf{T}}\right)\right)_{\Omega_e} - \left(\nabla \boldsymbol{v}_h, \frac{\nu}{2}\left([\![\boldsymbol{u}_h]\!] + [\![\boldsymbol{u}_h]\!]^{\mathsf{T}}\right)\right)_{\partial\Omega_e} \\ &- \left(\boldsymbol{v}_h, \nu\{\{\nabla \boldsymbol{u}_h + (\nabla \boldsymbol{u}_h)^{\mathsf{T}}\}\} \cdot \boldsymbol{n}\right)_{\partial\Omega_e} + \left(\boldsymbol{v}_h, \nu\tau\left([\![\boldsymbol{u}_h]\!] + [\![\boldsymbol{u}_h]\!]^{\mathsf{T}}\right) \cdot \boldsymbol{n}\right)_{\partial\Omega_e} \,. \end{aligned} \tag{2.149}$$

By inserting boundary conditions according to Table 2.3 into equation (2.149), the weak formulation of the viscous operator v_h^e can be decomposed into

$$v_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h, \boldsymbol{g}_u, \boldsymbol{h}_u\right) = v_{h,\mathrm{hom}}^e(\boldsymbol{v}_h, \boldsymbol{u}_h) + v_{h,\mathrm{inhom}}^e(\boldsymbol{v}_h, \boldsymbol{g}_u, \boldsymbol{h}_u) \,, \tag{2.150}$$

where the homogeneous part is

$$\begin{aligned} v_{h,\mathrm{hom}}^e(\boldsymbol{v}_h, \boldsymbol{u}_h) = &+ \left(\nabla \boldsymbol{v}_h, \nu\left(\nabla \boldsymbol{u}_h + (\nabla \boldsymbol{u}_h)^{\mathsf{T}}\right)\right)_{\Omega_e} \\ &- \left(\nabla \boldsymbol{v}_h, \frac{\nu}{2}\left([\![\boldsymbol{u}_h]\!] + [\![\boldsymbol{u}_h]\!]^{\mathsf{T}}\right)\right)_{\partial\Omega_e\setminus\Gamma_h} - \left(\nabla \boldsymbol{v}_h, \nu\left(\boldsymbol{u}_h \otimes \boldsymbol{n} + (\boldsymbol{u}_h \otimes \boldsymbol{n})^{\mathsf{T}}\right)\right)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} \\ &- \left(\boldsymbol{v}_h, \nu\{\{\nabla \boldsymbol{u}_h + (\nabla \boldsymbol{u}_h)^{\mathsf{T}}\}\} \cdot \boldsymbol{n}\right)_{\partial\Omega_e\setminus\Gamma_h} - \left(\boldsymbol{v}_h, \nu\left(\nabla \boldsymbol{u}_h + (\nabla \boldsymbol{u}_h)^{\mathsf{T}}\right) \cdot \boldsymbol{n}\right)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} \\ &+ \left(\boldsymbol{v}_h, \nu\tau\left([\![\boldsymbol{u}_h]\!] + [\![\boldsymbol{u}_h]\!]^{\mathsf{T}}\right) \cdot \boldsymbol{n}\right)_{\partial\Omega_e\setminus\Gamma_h} + \left(\boldsymbol{v}_h, 2\nu\tau\left(\boldsymbol{u}_h \otimes \boldsymbol{n} + (\boldsymbol{u}_h \otimes \boldsymbol{n})^{\mathsf{T}}\right) \cdot \boldsymbol{n}\right)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} \end{aligned} \tag{2.151}$$

and the inhomogeneous part

$$\begin{aligned} v_{h,\mathrm{inhom}}^e(\boldsymbol{v}_h; \boldsymbol{g}_u, \boldsymbol{h}_u) = &+ \left(\nabla \boldsymbol{v}_h, \nu\left(\boldsymbol{g}_u \otimes \boldsymbol{n} + (\boldsymbol{g}_u \otimes \boldsymbol{n})^{\mathsf{T}}\right)\right)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} \\ &- (\boldsymbol{v}_h, \boldsymbol{h}_u)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{N}}} \\ &- \left(\boldsymbol{v}_h, 2\nu\tau\left(\boldsymbol{g}_u \otimes \boldsymbol{n} + (\boldsymbol{g}_u \otimes \boldsymbol{n})^{\mathsf{T}}\right) \cdot \boldsymbol{n}\right)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}} \,. \end{aligned} \tag{2.152}$$

2.4.2.4.3 A note on the correct imposition of Neumann boundary conditions In the weak formulation of the momentum equation (2.76) and during the derivation of the weak formulation of the viscous term and the pressure gradient term, a splitting of the Neumann boundary condition $\boldsymbol{h} = \boldsymbol{h}_u - g_p\boldsymbol{n}$ into a viscous part and a pressure part according to equations (2.7) and (2.8) has been assumed. However, it is actually equation (2.6) that defines the

Neumann boundary condition to be prescribed in case of the coupled solution approach. This section shows that the splitting of boundary conditions does not impact the discrete system of equations for the coupled solution approach, and that the boundary conditions are imposed correctly.

The inhomogeneous boundary face integrals of the viscous term, equation (2.136) for the Laplace formulation and equation (2.152) for the divergence formulation, and the pressure gradient term according to equation (2.104) are added in the monolithic system, equation (2.76), so that any decomposition $\boldsymbol{h} = \boldsymbol{h}_u - g_p \boldsymbol{n}$ ensures a correct imposition of the Neumann boundary condition (2.6)

$$-(\boldsymbol{v}_h, \boldsymbol{h}_u)_{\partial\Omega_e \cap \Gamma_h^{\mathrm{N}}} + (\boldsymbol{v}_h, g_p \boldsymbol{n})_{\partial\Omega_e \cap \Gamma_h^{\mathrm{N}}} = -(\boldsymbol{v}_h, \boldsymbol{h}_u - g_p \boldsymbol{n})_{\partial\Omega_e \cap \Gamma_h^{\mathrm{N}}} = -(\boldsymbol{v}_h, \boldsymbol{h})_{\partial\Omega_e \cap \Gamma_h^{\mathrm{N}}} \,. \quad (2.153)$$

Without loss of generality, one can for example use $\boldsymbol{h}_u = \boldsymbol{h}$ and $g_p = 0$ for the coupled solution approach. A decoupled treatment of the Neumann boundary condition according to equations (2.7) and (2.8) is, however, necessary for the projection methods considered in this work.

2.4.2.5 Consistent stabilization terms

Numerical results give evidence that the L^2-conforming discretization introduced above lacks robustness for small values of the viscosity and coarse spatial resolutions. In particular, this problem prevents the successful simulation of three-dimensional turbulent flows in under-resolved scenarios and might even lead to numerical blow-up of the simulation. This section proposes two additional terms in the variational formulation in order to obtain a robust and accurate discretization for L^2-conforming discontinuous Galerkin formulations. These terms are consistent stabilizations terms that are motivated from the point of view of mass conservation and energy stability. This section deals with the first aspect of mass conservation, while Section 2.4.5 below sheds light on the beneficial effect of these terms in terms of energy stability.

The divergence and continuity penalty terms discussed here have their origin in the works by Joshi et al. (2016), Steinmoeller et al. (2013), where it was found that a better fulfillment of the divergence-free constraint and inter-element continuity of the velocity field improves the stability of the spatially discretized pressure-projection operator in projection-type methods. However, these works do not introduce these remedies for improved mass conservation as additional terms in a variational context, but rather as postprocessing techniques applied to the velocity field. Consistent divergence and continuity penalty terms added to the variational form have first been proposed in Krank et al. (2017) in the context of a projection method. By comparing monolithic and projection-type Navier–Stokes solvers, it has been shown in Fehn et al. (2018b) that the stabilization terms are not specifically related to the use of a projection scheme, but rather to the L^2-conforming nature of the function space. As argued in Akbas et al. (2018), these stabilization terms – specifically designed for L^2-conforming discretizations – can be seen in analogy to the well-known grad-div stabilization for H^1-conforming discretizations (Franca and Hughes 1988, Olshanskii et al. 2009). The work by Schroeder and Lube (2017) used a grad-div stabilization term only, and the works by Guzmán et al. (2016), Montlaur et al. (2008) used a normal-continuity penalty term. A shortcoming of these works is that the stabilization parameters have not been defined in a way to ensure consistent physical units. It should be emphasized that the use of a divergence penalty term only is not sufficient in case of L^2-conforming dis-

cretizations, and that an additional normal-continuity penalty term is required in general (Fehn et al. 2018b).

As a motivation, consider the discretized continuity equation (2.77) with the divergence operator in weak form, equation (2.97),

$$-\left(\nabla q_h, \boldsymbol{u}_h\right)_{\Omega_e} + \left(q_h, \{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right)_{\partial\Omega_e} = 0 \ , \tag{2.154}$$

or using the equivalent strong formulation, equation (2.98),

$$-\left(q_h, \nabla \cdot \boldsymbol{u}_h\right)_{\Omega_e} + \left(q_h, \frac{1}{2}\left[\boldsymbol{u}_h\right] \cdot \boldsymbol{n}\right)_{\partial\Omega_e} = 0 \ . \tag{2.155}$$

Inserting a constant test function $q_h = 1$ into the weak form yields the conservation property

$$\left(1, \{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right)_{\partial\Omega_e} = \int_{\partial\Omega_e} \{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n} \mathrm{d}\Gamma = 0 \ , \tag{2.156}$$

i.e., the mass fluxes over all boundaries of an element equate to zero, but only in the sense of the average velocity $\{\{\boldsymbol{u}_h\}\}$ and not for the interior velocity $\boldsymbol{u}_h^-$. The strong formulation of the continuity equation highlights that the divergence-free constraint and the continuity of the normal velocity between elements are fulfilled in a weak sense for L^2-conforming discretizations and not in a pointwise exact sense.

To improve mass conservation, two additional terms can be added to the momentum equation (2.76) of the variational formulation

$$\begin{aligned} m_{h,u}^e\left(\boldsymbol{v}_h, \frac{\partial \boldsymbol{u}_h}{\partial t}\right) + c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h; \boldsymbol{g}_u\right) + v_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h; \boldsymbol{g}_u, \boldsymbol{h}_u\right) \\ + a_{\mathrm{D},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h) + a_{\mathrm{C},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h; \boldsymbol{g}_u) + g_h^e\left(\boldsymbol{v}_h, p_h; g_p\right) - b_h^e\left(\boldsymbol{v}_h, \boldsymbol{f}(t)\right) = 0 \ , \end{aligned} \tag{2.157}$$

$$-d_h^e(q_h, \boldsymbol{u}_h; \boldsymbol{g}_u) = 0 \ , \tag{2.158}$$

where $a_{\mathrm{D},h}^e$ denotes a divergence penalty term and $a_{\mathrm{C},h}^e$ a continuity penalty term. These terms are introduced below and enforce the divergence-free constraint as well as continuity of the velocity field across elements in a weak sense.

The divergence penalty term $a_{\mathrm{D},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h)$ has similarities with the grad–div stabilization term often used in continuous finite element methods, see for example Olshanskii et al. (2009), and is defined as

$$a_{\mathrm{D},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h) = \left(\nabla \cdot \boldsymbol{v}_h, \tau_{\mathrm{D}} \nabla \cdot \boldsymbol{u}_h\right)_{\Omega_e} \ . \tag{2.159}$$

The penalty parameter τ_{D} is derived by means of dimensional analysis and is expressed in terms of a characteristic velocity and an effective element length

$$\tau_{\mathrm{D},e} = \zeta_{\mathrm{D}} \, \overline{\|\boldsymbol{u}_h\|} \, \frac{h}{k_u + 1} \ , \tag{2.160}$$

where the norm of the velocity $\|\boldsymbol{u}_h\|$ acts as a characteristic velocity measure, $\overline{(\bullet)}$ denotes an elementwise volume-averaged quantity, and $h = V_e^{1/3}$ a characteristic element length where V_e

is the element volume. In the time-discrete case, the characteristic velocity is replaced by a high-order extrapolation of order J using information from previous time steps

$$\boldsymbol{u}_h^{n+1,\mathrm{ex}} = \sum_{i=0}^{J-1} \beta_i \boldsymbol{u}_h^{n-i} \ . \tag{2.161}$$

A similar expression of the stabilization parameter is obtained in Olshanskii et al. (2009) in the convection-dominated regime, where the grad–div stabilization term is motivated from the point of view of variational multiscale methods and residual-based subgrid modeling. The factor $h/(k_u+1)$ in equation (2.160) is an effective element length scale taking into account shape functions of higher polynomial degree. The continuity penalty term $a_{\mathrm{C},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h)$ is defined as

$$a_{\mathrm{C},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h) = \left(\boldsymbol{v}_h \cdot \boldsymbol{n}, \tau_{\mathrm{C},f} \left[\boldsymbol{u}_h\right] \cdot \boldsymbol{n}\right)_{\partial\Omega_e} \ . \tag{2.162}$$

While Krank et al. (2017) applied the continuity penalty term to all components of the velocity field, Fehn et al. (2018b) proposed to apply this term to the normal component of the velocity only, motivated from the analogy to exactly divergence-free H^{div}-conforming discretizations, see Section 2.4.3. As suggested in Fehn et al. (2021a), the continuity penalty term is applied to all faces of an element and not only to the interior faces as in Fehn et al. (2018b), Krank et al. (2017), where mainly turbulent flow examples with periodic boundaries have been considered. The penalty parameter $\tau_{\mathrm{C},f}$ on an interior face $f \subseteq \partial\Omega_e \setminus \Gamma_h$ is $\tau_{\mathrm{C},f} = \{\{\tau_{\mathrm{C},e}\}\} = \left(\tau_{\mathrm{C},e^-} + \tau_{\mathrm{C},e^+}\right)/2$. The elementwise continuity penalty factor is derived by means of dimensional analysis and has the physical unit of a velocity

$$\tau_{\mathrm{C},e} = \zeta_{\mathrm{C}} \, \overline{\|\boldsymbol{u}_h\|} \ . \tag{2.163}$$

In the time-discrete case, the characteristic velocity in the above equation is again replaced by a high-order extrapolation according to equation (2.161). Numerical results in Fehn et al. (2018b, 2019a) demonstrate that a penalty factor of $\mathcal{O}(1)$ leads to a discretization scheme that is robust independently of the discretization parameters h, k and the Reynolds number Re. The default value of the penalty factors is therefore $\zeta_{\mathrm{D}} = \zeta_{\mathrm{C}} = 1$, which will be used for numerical experiments unless specified otherwise.

Remark 2.9 *It is essential to formulate the penalty terms in a way that the physical units of these terms are consistent with the other terms of the Navier–Stokes equations, an aspect often not taken into account in the literature, see Akbas et al. (2018), Guzmán et al. (2016), Montlaur et al. (2008), Schroeder and Lube (2017). This is necessary in order to obtain a robust discretization that does not require a readjustment of parameters depending on the spatial resolution parameters h, k, the size of the geometry, or the Reynolds number Re. When interpreted from the point of view of implicit large-eddy simulation, this strategy allows to construct a parameter-free implicit turbulence model with suitable inbuilt numerical dissipation mechanisms (Fehn et al. 2018b, 2019a), see also the discussion in Section 2.4.5.3.*

Remark 2.10 *It is straightforward to verify consistency of the above variational formulation, equations* (2.157) *and* (2.158)*. The weak form of the individual terms is derived using integration-by-parts, defining consistent flux functions, and imposing consistent boundary conditions. Since the divergence and continuity penalty terms contain the divergence or the jump of the velocity, both vanishing when inserting the exact solution $\boldsymbol{u}$, consistency follows immediately.*

2.4.3 Analogy to exactly divergence-free H^{div}-conforming elements

This section briefly points out similarities between the stabilized L^2-conforming approach discussed here and exactly divergence-free H^{div}-conforming discretizations (realized for example by the well-known space by Raviart and Thomas (1977) in case of tensor-product elements). For a more thorough discussion, the reader is referred to Fehn et al. (2019a) and references therein. The work by Fehn et al. (2019a) also provides a one-to-one comparison of the accuracy of both approaches in the context of under-resolved turbulent flow simulations. The H^{div}-conforming space is defined as

$$\mathcal{V}_h^{u,H^{\mathrm{div}}} = \left\{ \boldsymbol{u}_h \in [L_2(\Omega_h)]^d \ : \ \nabla \cdot \boldsymbol{u}_h \in L_2(\Omega_h) \right\} \ . \tag{2.164}$$

H^{div}-conformity implies that the velocity is continuous in normal direction, i.e., $[\boldsymbol{u}_h] \cdot \boldsymbol{n} = 0$. Raviart–Thomas (RT) elements defined as

$$\mathcal{V}_h^{u,\mathrm{RT}} = \left\{ \boldsymbol{u}_h \in \mathcal{V}_h^{u,H^{\mathrm{div}}} \ : \ \nabla \cdot \boldsymbol{u}_h \in \mathcal{V}_h^p \right\} \tag{2.165}$$

are exactly divergence-free, i.e., $\nabla \cdot \boldsymbol{u}_h = 0$. This can easily be derived from the continuity equation in strong formulation

$$-\left(q_h, \nabla \cdot \boldsymbol{u}_h\right)_{\Omega_e} + \left(q_h, \frac{1}{2} \left[\boldsymbol{u}_h\right] \cdot \boldsymbol{n} \right)_{\partial\Omega_e} = 0 \xrightarrow{H^{\mathrm{div}}} -\left(q_h, \nabla \cdot \boldsymbol{u}_h\right)_{\Omega_e} = 0 \xrightarrow{\mathrm{RT}} \nabla \cdot \boldsymbol{u}_h = 0 \ . \tag{2.166}$$

An illustration of these function spaces is given in Figure 2.3. The Raviart–Thomas space leads to function spaces with different polynomial degrees in different coordinate directions. On Cartesian meshes, the condition $\nabla \cdot \boldsymbol{u}_h \in \mathcal{V}_h^p$ can be easily realized by using polynomials of degree k in direction ξ_i and polynomials of degree $k-1$ in direction $\xi_j, j \neq i$ to approximate component $u_{i,h}^e$

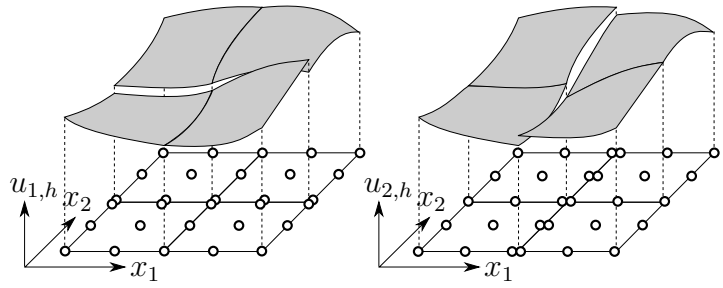
$$\boldsymbol{u}_h^e = \begin{bmatrix} u_{1,h}^e \\ \vdots \\ u_{d,h}^e \end{bmatrix} \in \begin{bmatrix} \mathbb{P}_k \otimes \mathbb{P}_{k-1} \otimes \ldots \otimes \mathbb{P}_{k-1} \\ \vdots \\ \mathbb{P}_{k-1} \otimes \ldots \otimes \mathbb{P}_{k-1} \otimes \mathbb{P}_k \end{bmatrix} \xrightarrow{\nabla \cdot} \nabla \cdot \boldsymbol{u}_h^e \in \mathcal{Q}_{k-1} = \mathcal{V}_{h,e}^p \ . \tag{2.167}$$

This leads to the following analogy: The use of consistent continuity and divergence penalty terms might therefore be interpreted as a weak enforcement of H^{div}-conformity and Raviart–Thomas elements, respectively, even though the limit $\zeta_{\mathrm{C}}, \zeta_{\mathrm{D}} \to \infty$ for the L^2-method is not equivalent to the H^{div}-conforming Raviart–Thomas method (Fehn et al. 2019a).

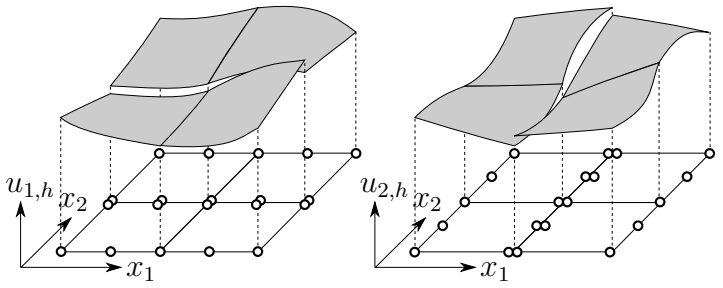
2.4.4 Pressure-robustness

The penalty terms have been motivated so far from the point of view of mass conservation. Recent works argue that the aspect of poor mass conservation is described better by a discretization property called "pressure-robustness", which is briefly discussed in this section.

Several early works on finite element discretizations for the incompressible Navier–Stokes equations discussed aspects of poor mass conservation and the occurrence of spurious velocities



(a) H^{div}-conforming space



(b) Raviart–Thomas elements

Figure 2.3: Illustration of H^{div}-conforming function space and Raviart–Thomas elements in two space dimensions ($d = 2$) for tensor-product elements of maximum polynomial degree $k = 2$.

for problems with irrotational body forces, see Dorok et al. (1994), Gerbeau et al. (1997), Pelletier et al. (1989). More recently, this topic has been explored systematically under the name "pressure-robustness", see Akbas et al. (2018), Galvin et al. (2012), Gauger et al. (2019), John et al. (2017), Lederer et al. (2017), Linke and Merdon (2016a), Linke (2014), Linke et al. (2016), Piatkowski and Bastian (2019). The term "pressure-robustness" describes the property that the velocity error is independent of the pressure and the viscosity for a discretization scheme of the incompressible Navier–Stokes equations. For non pressure-robust discretizations, however, the velocity error has a dependency on the pressure norm and also the inverse of the viscosity. Pressure-robustness is therefore particularly relevant for problems with large or complicated pressure and small viscosities. Pressure-robustness is related to the question whether an irrotational body force, i.e., a body force that can be written as the gradient of a scalar quantity, $\boldsymbol{f} = \nabla\phi$, causes spurious velocities. In the continuous/analytical case, this body force should only enter the pressure solution and not affect the velocity solution since the pressure term in the momentum equation, ∇p, has exactly the same structure as the body force, i.e., $\boldsymbol{f} \to \boldsymbol{f} + \nabla\phi \Rightarrow (\boldsymbol{u}, p) \to (\boldsymbol{u}, p + \phi)$ (which is known as fundamental invariance property of the incompressible Navier–Stokes equations). In the discrete case, however, this is fulfilled only under certain assumptions. The problematic term is the right-hand side of the momentum equation in case it is a gradient field, $\boldsymbol{f} = \nabla\phi$. Consider the right-hand side term $(\boldsymbol{v}_h, \boldsymbol{f})_{\Omega_e}$ in

equation (2.157) with $\boldsymbol{f} = \nabla\phi$, let $\phi \in H^1(\Omega)$, set $\boldsymbol{v}_h = \boldsymbol{u}_h$, and perform integration by parts

$$\begin{aligned}\sum_{e=1}^{N_{\text{el}}} (\boldsymbol{u}_h, \nabla\phi)_{\Omega_e} &= \sum_{e=1}^{N_{\text{el}}} \Big(- (\phi, \nabla\cdot\boldsymbol{u}_h)_{\Omega_e} + \big(\phi, \boldsymbol{u}_h^- \cdot \boldsymbol{n}\big)_{\partial\Omega_e} - (\phi, \{\{\boldsymbol{u}_h\}\}\cdot\boldsymbol{n})_{\partial\Omega_e} \Big) \\ &= \sum_{e=1}^{N_{\text{el}}} \left(- (\phi, \nabla\cdot\boldsymbol{u}_h)_{\Omega_e} + \left(\phi, \frac{1}{2}[\boldsymbol{u}_h]\cdot\boldsymbol{n}\right)_{\partial\Omega_e} \right) \stackrel{?}{=} 0 \,.\end{aligned} \tag{2.168}$$

The last term in the first row can be added (assuming periodic boundaries for ease of notation) since it drops out when summing over all elements, due to the fact that ϕ is continuous and $\{\{\boldsymbol{u}_h\}\}$ is single-valued. For exactly divergence-free H^{div}-conforming finite element formulations discussed in Section 2.4.3, the right-hand side becomes zero for all ϕ, i.e., gradient fields are L^2-orthogonal to discretely (in this case also exactly) divergence-free velocity fields. No spurious velocities can be amplified by gradient fields and the formulation is pressure-robust. For L^2-conforming methods discussed in the present work, the right-hand side does not become zero in general since – unlike in equation (2.155) – the potential $\phi \in H^1 \not\subseteq \mathcal{V}_h^p$ is not in the pressure space in general, i.e., the divergence-free constraint is relaxed and gradient fields are not L^2-orthogonal to discretely divergence-free velocities. As a result, the velocity solution is affected for non pressure-robust discretizations such as classical inf–sup stable mixed finite element methods (where the velocity error has a $\nu^{-1}|p|$ dependency). Projection or reconstruction techniques have been proposed to improve or ensure pressure-robustness (Gerbeau et al. 1997, Lederer et al. 2017, Linke and Merdon 2016a, Linke 2014, Linke et al. 2016), which can be interpreted as filtering out problematic contributions of the right-hand side in order to not perturb the velocity solution (the notion used in John et al. (2017) is repairing the L^2-orthogonality). Other techniques are grad–div stabilization (Akbas et al. 2018, Galvin et al. 2012), which might be described as diagnostic rather than preventive measures to deal with pressure-robustness. The discretization scheme of the incompressible Navier–Stokes equations proposed in this chapter is not pressure-robust due to its L^2-conforming nature. It is based on penalty terms for improved mass conservation and energy stability that do not result in an exactly divergence-free and mass-conserving velocity field. These penalty terms can be interpreted as the analogue of grad–div stabilization for DG discretizations and as a weak enforcement of pressure-robustness, reducing the dependency of the velocity error on the pressure from $\nu^{-1}|p|$ to only $\nu^{-1/2}|p|$ according to the analysis in Akbas et al. (2018). An important property in this context is that the stabilized DG method does not suffer from over-stabilization as reported for other stabilized continuous finite element methods, and that the solution converges to that of a pressure-robust method for increasing penalty parameter with a rate of $\zeta^{-1/2}$ (Akbas et al. 2018). Numerical investigations presented in Piatkowski and Bastian (2019) reveal that the stabilized L^2-approach performs very well compared to a pressure-robust formulation with H^{div}-reconstruction. The results shown in Fehn et al. (2019a) suggest that the present stabilized L^2-conforming approach – although not being pressure-robust – seems to be not less accurate for under-resolved turbulent flow scenarios than exactly divergence-free and pressure-robust H^{div}-conforming discretizations.

Remark 2.11 *Considering the limit $\nu \to 0$ in mathematical error estimates (see for example the error estimate (3.5) in John et al. (2017)) appears to be somewhat artificial since, from a physical perspective, the limit $\nu \to 0$ typically changes the structure of the flow for general fluid*

dynamical problems and implies turbulent flows, with discretization schemes inevitably operating in the preasymptotic regime of convergence with large velocity (interpolation) errors. This raises the question whether pressure-robustness is the effect that limits overall accuracy in such under-resolved scenarios. While classical no-flow examples with smooth analytical solution and irrotational forcing are certainly impressive in demonstrating a superior behavior of pressure-robust discretizations (the velocity term on the right-hand side of the error estimate diminishes compared to the pressure term), it still appears to be challenging to find practical flow problems highlighting the need for strict pressure-robustness on top of what stabilized approaches can provide. Studies by Linke and Merdon (2016b), Linke (2014) state that the aspect of pressure-robustness has not been sufficiently internalized by the CFD community. Gauger et al. (2019) state that "pressure-robustness appears to be a prerequisite for accurate incompressible flow solvers at high Reynolds numbers", but no turbulent flow example is studied in that work. The results shown in the present work contribute also to this discussion. According to the author's opinion, there still needs to be given motivation that pressure-robustness in the sense of truly pressure-independent velocity errors is a prerequisite for robust and accurate incompressible flow solvers. It appears as if theoretical estimates somewhat under-estimate the potential of stabilized methods (see for example the comparative studies by Fehn et al. (2019a), Piatkowski and Bastian (2019)). It could therefore be very insightful if theoretical estimates could be refined or extended in this direction.

Numerical results on this topic are shown in Section 2.6.4 for a steady Stokes problem with manufactured solution. An important class of applications for which the aspect of pressure-robustness is expected to be relevant are natural convection flows with large irrotational forces due to buoyancy forces originating from temperature and density variations, a topic discussed in more detail in Chapter 3.

2.4.5 Energy stability

This section discusses the aspect of energy stability for the continuous-in-time formulation of the L^2-conforming discretization of the incompressible Navier–Stokes equations introduced in Section 2.4.2. Moreover, this section serves as the second main motivation for the use of consistent stabilization terms introduced in Section 2.4.2.5, apart from the aspect of mass conservation. The analysis of energy stability is based on the following assumptions:

- The computational domain Ω_h is time-invariant.
- Periodic boundaries are applied on $\Gamma_h = \partial\Omega_h$.
- There are no body forces, $\boldsymbol{f} = \boldsymbol{0}$.
- Integrals are computed exactly in the discrete case.
- The solution is sufficiently smooth so that the usual integral transformations (integration-by-parts, Gauss' divergence theorem) apply.

The last assumption implies (together with the other assumptions) that, in the continuous case, energy is conserved exactly in the absence of viscosity, $\nu = 0$. In many numerical works studying the energy conservation properties of a discretization scheme, this assumption is taken as

valid without justification. However, this assumption can be rather restrictive especially for the inviscid limit (Euler equations). There is an ongoing debate in the fluid mechanics community whether the incompressible Euler equations might develop singularities in finite time. Closely related to this aspect is the occurrence of anomalous energy dissipation in the inviscid limit according to Onsager's conjecture, i.e., anomalous energy dissipation might indeed occur if the velocity field is no longer differentiable due to singularities. The reader is referred to Chapter 7 for an in-depth discussion of this aspect. With these restrictions in mind, the aim of this section is to investigate the energy stability of the discretization scheme and not a proof of exact energy conservation for vanishing viscosity. Regarding the latter aspect of energy conservation, a provably energy-conserving scheme would then in fact be unable to predict a potential energy dissipation anomaly.

The numerical scheme is called energy stable if the kinetic energy E fulfills

$$E(t) = \int_{\Omega_h} \frac{1}{2} \boldsymbol{u}_h \cdot \boldsymbol{u}_h \, \mathrm{d}\Omega \leq 0 \, . \tag{2.169}$$

The rate of change of the kinetic energy can be expressed in terms of the velocity mass matrix operator

$$\frac{\mathrm{d}E(t)}{\mathrm{d}t} = m_{h,u}\left(\boldsymbol{u}_h, \frac{\partial \boldsymbol{u}_h}{\partial t}\right) = \sum_{e=1}^{N_\mathrm{el}} m_{h,u}^e \left(\boldsymbol{u}_h, \frac{\partial \boldsymbol{u}_h}{\partial t}\right) \, , \tag{2.170}$$

Inserting the discretized momentum equation (2.157) into equation (2.170) along with the above assumptions results in

$$\begin{aligned} \frac{\mathrm{d}E(t)}{\mathrm{d}t} &= -\sum_{e=1}^{N_\mathrm{el}} \left(c_h^e\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) + v_h^e\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) + g_h^e\left(\boldsymbol{u}_h, p_h\right) + a_{\mathrm{D},h}^e\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) + a_{\mathrm{C},h}^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h\right)\right) \\ &= -\left(c_h\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) + v_h\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) + g_h\left(\boldsymbol{u}_h, p_h\right) + a_{\mathrm{D},h}\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) + a_{\mathrm{C},h}\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right)\right) \, . \end{aligned} \tag{2.171}$$

Under the above assumptions, the discontinuous Galerkin formulation is symmetric with respect to the pressure gradient term and velocity divergence term, $g_h\left(\boldsymbol{u}_h, p_h\right) = -d_h\left(p_h, \boldsymbol{u}_h\right)$. Moreover, since $-d_h\left(p_h, \boldsymbol{u}_h\right) = 0$ due to the discretized continuity equation, equation (2.77) or equation (2.158), the pressure gradient term does not contribute to the rate of change of the kinetic energy. In the presence of integration errors, this holds only for weak–strong combinations of the velocity–pressure coupling terms according to Section 2.4.2.2. The SIPG discretization of the viscous term is positive semi-definite, $v_h\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) \geq 0$, which implies an energy dissipating behavior in terms of the kinetic energy evolution, irrespective of quadrature errors. Since the viscous term becomes zero in the inviscid limit, $\nu = 0$, which is most interesting from the point of view of energy stability, other terms in the energy estimate cannot be balanced by the viscous term. The viscous term is, therefore, also dropped

$$\frac{\mathrm{d}E(t)}{\mathrm{d}t} \leq -c_h\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) - a_{\mathrm{D},h}\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) - a_{\mathrm{C},h}\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) \, . \tag{2.172}$$

The divergence and continuity penalty terms are by definition positive semi-definite. These terms are, however, kept in the energy estimate as their aim is to balance contributions from the nonlinear convective term that might otherwise lead to a blow-up of the discrete solution. To investigate

the contributions of the convective term to the kinetic energy evolution, the divergence and convective formulations of this term are considered separately in the following.

2.4.5.1 Divergence formulation of the convective term

The goal is to express the convective term as a function of terms like $\nabla \cdot \boldsymbol{u}_h$ or $[\boldsymbol{u}_h]$ that vanish in the continuous case. To this end, the convective term in divergence form, $\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) = \boldsymbol{u} \otimes \boldsymbol{u}$, is first reformulated by integrating the first term on the right-hand side of equation (2.80) by parts once again to obtain the so-called strong formulation

$$c_h^e\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) = \left(\boldsymbol{u}_h, \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h)\right)_{\Omega_e} + \left(\boldsymbol{u}_h, \left(\boldsymbol{F}_{\mathrm{c}}^*(\boldsymbol{u}_h) - \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h)\right) \cdot \boldsymbol{n}\right)_{\partial\Omega_e} . \tag{2.173}$$

Moreover, the following relation is used

$$\left(\boldsymbol{u}_h, \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h)\right)_{\Omega_e} = \frac{1}{2}\left(\nabla \cdot \boldsymbol{u}_h, \boldsymbol{u}_h \cdot \boldsymbol{u}_h\right)_{\Omega_e} + \frac{1}{2}\left(\boldsymbol{u}_h, \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h) \cdot \boldsymbol{n}\right)_{\partial\Omega_e} , \tag{2.174}$$

Equation (2.174) can be derived by using the identities

$$\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = (\nabla \boldsymbol{u}) \cdot \boldsymbol{u} + (\nabla \cdot \boldsymbol{u})\, \boldsymbol{u} , \tag{2.175}$$

$$\nabla \cdot (\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) \cdot \boldsymbol{u}) = (\nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u})) \cdot \boldsymbol{u} + \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) : \nabla \boldsymbol{u} , \tag{2.176}$$

as well as integration-by-parts, to obtain

$$\begin{aligned}
\int_\Omega \boldsymbol{u} \cdot (\nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}))\, \mathrm{d}\Omega &= \int_\Omega \boldsymbol{u} \cdot ((\nabla \boldsymbol{u}) \cdot \boldsymbol{u} + (\nabla \cdot \boldsymbol{u})\, \boldsymbol{u})\, \mathrm{d}\Omega \\
&= \int_\Omega (\nabla \cdot \boldsymbol{u})\, \boldsymbol{u} \cdot \boldsymbol{u}\, \mathrm{d}\Omega + \int_\Omega \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) : \nabla \boldsymbol{u}\, \mathrm{d}\Omega \\
&= \int_\Omega (\nabla \cdot \boldsymbol{u})\, \boldsymbol{u} \cdot \boldsymbol{u}\, \mathrm{d}\Omega - \int_\Omega \boldsymbol{u} \cdot (\nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}))\, \mathrm{d}\Omega \\
&\quad + \int_{\partial\Omega} (\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) \cdot \boldsymbol{u}) \cdot \boldsymbol{n}\, \mathrm{d}\Gamma .
\end{aligned} \tag{2.177}$$

Equation (2.174) is obtained by shifting the second term on the right-hand side of equation (2.177) to the left. Inserting equation (2.174) and equation (2.81) for the Lax–Friedrichs flux into equation (2.173), and summing over all elements yields

$$\begin{aligned}
&-c_h\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) = \\
&= -\sum_{e=1}^{N_{\mathrm{el}}} \left(\frac{1}{2}\left(\nabla \cdot \boldsymbol{u}_h, \boldsymbol{u}_h \cdot \boldsymbol{u}_h\right)_{\Omega_e} + \left(\boldsymbol{u}_h, \frac{1}{2}\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h^+) \cdot \boldsymbol{n}\right)_{\partial\Omega_e} + \left(\boldsymbol{u}_h \otimes \boldsymbol{n}, \frac{\Lambda}{2}[\![\boldsymbol{u}_h]\!]\right)_{\partial\Omega_e} \right) \\
&= -\frac{1}{2}\left(\nabla \cdot \boldsymbol{u}_h, \boldsymbol{u}_h \cdot \boldsymbol{u}_h\right)_{\Omega_h} \\
&\quad - \left(\boldsymbol{u}_h^-, \frac{1}{2}\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h^+) \cdot \boldsymbol{n}^-\right)_{\Gamma_h^{\mathrm{int}}} - \left(\boldsymbol{u}_h^+, \frac{1}{2}\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h^-) \cdot \boldsymbol{n}^+\right)_{\Gamma_h^{\mathrm{int}}} \\
&\quad - \left([\![\boldsymbol{u}_h]\!], \frac{\Lambda}{2}[\![\boldsymbol{u}_h]\!]\right)_{\Gamma_h^{\mathrm{int}}} ,
\end{aligned} \tag{2.178}$$

where Γ_h^{int} denotes the set of all interior faces (note that $\Gamma_h = \partial\Omega_h = \emptyset$ due to periodic boundary conditions). The second term and the third term on the right-hand side of equation (2.178) can be further simplified by algebraic manipulations using the oriented jump operator $[\boldsymbol{u}_h] = \boldsymbol{u}_h^- - \boldsymbol{u}_h^+$

$$\begin{aligned}
&\left(\boldsymbol{u}_h^-, \frac{1}{2}\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h^+)\cdot\boldsymbol{n}^-\right)_{\Gamma_h^{\mathrm{int}}} + \left(\boldsymbol{u}_h^+, \frac{1}{2}\boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}_h^-)\cdot\boldsymbol{n}^+\right)_{\Gamma_h^{\mathrm{int}}} \\
&= \left(\boldsymbol{u}_h^-, \left(\frac{1}{2}\boldsymbol{u}_h^-\otimes\boldsymbol{u}_h^- - \frac{1}{2}\boldsymbol{u}_h^-\otimes[\boldsymbol{u}_h] - \frac{1}{2}[\boldsymbol{u}_h]\otimes\boldsymbol{u}_h^- + \frac{1}{2}[\boldsymbol{u}_h]\otimes[\boldsymbol{u}_h]\right)\cdot\boldsymbol{n}^-\right)_{\Gamma_h^{\mathrm{int}}} \\
&\quad + \left(\boldsymbol{u}_h^-, \left(\frac{1}{2}\boldsymbol{u}_h^-\otimes\boldsymbol{u}_h^-\right)\cdot\boldsymbol{n}^+\right)_{\Gamma_h^{\mathrm{int}}} - \left([\boldsymbol{u}_h], \left(\frac{1}{2}\boldsymbol{u}_h^-\otimes\boldsymbol{u}_h^-\right)\cdot\boldsymbol{n}^+\right)_{\Gamma_h^{\mathrm{int}}} \\
&\quad = \left(\boldsymbol{u}_h^-, \left(-\frac{1}{2}\boldsymbol{u}_h^-\otimes[\boldsymbol{u}_h] + \frac{1}{2}[\boldsymbol{u}_h]\otimes[\boldsymbol{u}_h]\right)\cdot\boldsymbol{n}^-\right)_{\Gamma_h^{\mathrm{int}}} \\
&= \left(\boldsymbol{u}_h^-, \left(-\frac{1}{2}\boldsymbol{u}_h^+\otimes[\boldsymbol{u}_h]\right)\cdot\boldsymbol{n}^-\right)_{\Gamma_h^{\mathrm{int}}} = \left(-\frac{1}{2}\boldsymbol{u}_h^-\cdot\boldsymbol{u}_h^+, [\boldsymbol{u}_h]\cdot\boldsymbol{n}^-\right)_{\Gamma_h^{\mathrm{int}}}.
\end{aligned} \tag{2.179}$$

Inserting equations (2.178) and (2.179) into equation (2.172) yields the result

$$\begin{aligned}
\frac{\mathrm{d}E(t)}{\mathrm{d}t} \leq &-\frac{1}{2}\left(\nabla\cdot\boldsymbol{u}_h, \boldsymbol{u}_h\cdot\boldsymbol{u}_h\right)_{\Omega_h} - a_{\mathrm{D},h}\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) \\
&+\frac{1}{2}\left([\boldsymbol{u}_h]\cdot\boldsymbol{n}, \boldsymbol{u}_h^-\cdot\boldsymbol{u}_h^+\right)_{\Gamma_h^{\mathrm{int}}} - a_{\mathrm{C},h}\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) \\
&-\left(\llbracket\boldsymbol{u}_h\rrbracket, \frac{\Lambda}{2}\llbracket\boldsymbol{u}_h\rrbracket\right)_{\Gamma_h^{\mathrm{int}}}.
\end{aligned} \tag{2.180}$$

2.4.5.2 Convective formulation of the convective term

The above analysis is repeated for the convective formulation with upwind flux, equation (2.94). In a first step, the volume term in equation (2.94) is reformulated so that it contains the divergence of the velocity. Inserting the identity

$$\boldsymbol{u}_h\cdot\nabla\boldsymbol{u}_h\cdot\boldsymbol{u}_h = -\frac{1}{2}\nabla\cdot\boldsymbol{u}_h\left(\boldsymbol{u}_h\cdot\boldsymbol{u}_h\right) + \frac{1}{2}\nabla\cdot\left(\boldsymbol{u}_h\left(\boldsymbol{u}_h\cdot\boldsymbol{u}_h\right)\right), \tag{2.181}$$

into equation (2.94) and applying Gauss' divergence theorem yields

$$\begin{aligned}
c_h^e\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) = &-\frac{1}{2}\left(\nabla\cdot\boldsymbol{u}_h, \boldsymbol{u}_h\cdot\boldsymbol{u}_h\right)_{\Omega_e} + \frac{1}{2}\left(\boldsymbol{u}_h\cdot\boldsymbol{u}_h, \boldsymbol{u}_h\cdot\boldsymbol{n}\right)_{\partial\Omega_e} \\
&-\left(\boldsymbol{u}_h, \left(\{\{\boldsymbol{u}_h\}\}\cdot\boldsymbol{n}\right)\boldsymbol{u}_h\right)_{\partial\Omega_e} \\
&+\left(\boldsymbol{u}_h, \left(\{\{\boldsymbol{u}_h\}\}\cdot\boldsymbol{n}\right)\{\{\boldsymbol{u}_h\}\} + \frac{1}{2}|\{\{\boldsymbol{u}_h\}\}\cdot\boldsymbol{n}|\,[\boldsymbol{u}_h]\right)_{\partial\Omega_e}.
\end{aligned} \tag{2.182}$$

Next, face integrals in the second row and third row of the above equation can be combined

$$\begin{aligned} c_h^e\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) = & -\frac{1}{2}\left(\nabla \cdot \boldsymbol{u}_h, \boldsymbol{u}_h \cdot \boldsymbol{u}_h\right)_{\Omega_e} \\ & + \frac{1}{2}\left(\boldsymbol{u}_h \cdot \boldsymbol{u}_h, \boldsymbol{u}_h \cdot \boldsymbol{n}\right)_{\partial\Omega_e} - \left(\boldsymbol{u}_h, \left(\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right) \frac{1}{2}\left[\boldsymbol{u}_h\right]\right)_{\partial\Omega_e} \\ & + \left(\boldsymbol{u}_h, \frac{1}{2}|\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}|\left[\boldsymbol{u}_h\right]\right)_{\partial\Omega_e} . \end{aligned} \tag{2.183}$$

Summation over all elements yields

$$\begin{aligned} c_h\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) = & -\frac{1}{2}\left(\nabla \cdot \boldsymbol{u}_h, \boldsymbol{u}_h \cdot \boldsymbol{u}_h\right)_{\Omega_h} \\ & + \sum_{e=1}^{N_{\mathrm{el}}} \left(\frac{1}{2}\left(\boldsymbol{u}_h \cdot \boldsymbol{u}_h, \boldsymbol{u}_h \cdot \boldsymbol{n}\right)_{\partial\Omega_e} - \left(\boldsymbol{u}_h, \left(\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right) \frac{1}{2}\left[\boldsymbol{u}_h\right]\right)_{\partial\Omega_e} \right) \\ & + \left(\left[\boldsymbol{u}_h\right], \frac{1}{2}|\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}|\left[\boldsymbol{u}_h\right]\right)_{\Gamma_h^{\mathrm{int}}} . \end{aligned} \tag{2.184}$$

Finally, it remains to reformulate and simplify the second row on the right-hand side of the above equation by using the oriented jump operator $[\boldsymbol{u}_h] = \boldsymbol{u}_h^- - \boldsymbol{u}_h^+$

$$\begin{aligned} & \sum_{e=1}^{N_{\mathrm{el}}} \left(\frac{1}{2}\left(\boldsymbol{u}_h \cdot \boldsymbol{u}_h, \boldsymbol{u}_h \cdot \boldsymbol{n}\right)_{\partial\Omega_e} - \left(\frac{\boldsymbol{u}_h}{2}, \left(\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right)\left[\boldsymbol{u}_h\right]\right)_{\partial\Omega_e} \right) \\ = & + \frac{1}{2}\left(\boldsymbol{u}_h^- \cdot \boldsymbol{u}_h^-, \boldsymbol{u}_h^- \cdot \boldsymbol{n}^-\right)_{\Gamma_h^{\mathrm{int}}} + \frac{1}{2}\left(\boldsymbol{u}_h^+ \cdot \boldsymbol{u}_h^+, \boldsymbol{u}_h^+ \cdot \boldsymbol{n}^+\right)_{\Gamma_h^{\mathrm{int}}} \\ & - \left(\{\{\boldsymbol{u}_h\}\}, \left(\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right)\left[\boldsymbol{u}_h\right]\right)_{\Gamma_h^{\mathrm{int}}} \\ = & \ \ldots \\ = & + \frac{1}{2}\left(\{\{\boldsymbol{u}_h \cdot \boldsymbol{u}_h\}\}, \left[\boldsymbol{u}_h\right] \cdot \boldsymbol{n}^-\right)_{\Gamma_h^{\mathrm{int}}} . \end{aligned} \tag{2.185}$$

The details of this step are skipped here as the procedure is similar to equation (2.179). The interested reader is referred to (Fehn et al. 2021a, Appendix C) where the algebraic manipulations are listed in detail. Inserting equations (2.184) and (2.185) into equation (2.172) yields the result

$$\begin{aligned} \frac{\mathrm{d}E(t)}{\mathrm{d}t} \leq & + \frac{1}{2}\left(\nabla \cdot \boldsymbol{u}_h, \boldsymbol{u}_h \cdot \boldsymbol{u}_h\right)_{\Omega_h} - a_{\mathrm{D},h}\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) \\ & - \frac{1}{2}\left(\left[\boldsymbol{u}_h\right] \cdot \boldsymbol{n}, \{\{\boldsymbol{u}_h \cdot \boldsymbol{u}_h\}\}\right)_{\Gamma_h^{\mathrm{int}}} - a_{\mathrm{C},h}\left(\boldsymbol{u}_h, \boldsymbol{u}_h\right) \\ & - \left(\left[\boldsymbol{u}_h\right], \frac{1}{2}|\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}|\left[\boldsymbol{u}_h\right]\right)_{\Gamma_h^{\mathrm{int}}} . \end{aligned} \tag{2.186}$$

2.4.5.3 Discussion

This section summarizes and discusses the energy estimates (2.180) and (2.186). It is interesting to realize that the energy estimate contains similar terms independently of the type of formulation used for the convective term. The divergence term in the first row containing $\nabla \cdot \boldsymbol{u}_h$ and the

jump term in the second row containing $[\boldsymbol{u}_h] \cdot \boldsymbol{n}$ are sign-indefinite for both formulations of the convective term and can be identified as potential sources of instabilities if no sufficient control is provided over divergence errors and jumps of the velocity between elements in the direction normal to the face. It is a key observation that the stabilization term of the Lax–Friedrichs flux or upwind flux in the third row of the energy estimates (2.180) and (2.186), despite exhibiting an energy-dissipating behavior, does not render the convective operator energy-stable in the discrete case. The origin is the nonlinear nature of the convective term in the incompressible Navier–Stokes equations, rendering the construction of energy-stable schemes non-trivial. This is in contrast to linear advection terms for which an upwind flux results in an energy-stable scheme (Hesthaven and Warburton 2007).

Note that these energy estimates are another main motivation for the use of consistent divergence and continuity penalty terms. Their aim is to balance the sign-indefinite terms orginating from the convective term, as a means to render the overall discretization scheme stable. Apart from the aspect of mass conservation, equation (2.155), the energy estimates (2.180) and (2.186) explain why it is sufficient to penalize the jump of the normal component of the velocity, instead of all velocity components as originally proposed in Krank et al. (2017). Since the divergence-free constraint and inter-element continuity of the velocity are only enforced in a weak sense through the use of penalty terms, there is currently no proof that the overall scheme is discretely energy-stable for the chosen penalty factors of order $\mathcal{O}(1)$. The reader is referred to Fehn et al. (2019a) for a characterization of the discretization scheme for the limit case $\tau_\mathrm{C}, \tau_\mathrm{D} \to \infty$. However, the effectiveness of the chosen stabilization approach (including the derivation of penalty factors by means of dimensional analysis) has been demonstrated by numerical experiments for challenging inviscid flow simulations such as the Taylor–Green vortex on Cartesian and deformed meshes (Fehn et al. 2018b). As discussed in detail in Chapter 7, the present discretization scheme might be suitable to predict the phenomenon of anomalous energy dissipation, see also Fehn et al. (2021b).

The problematic sign-indefinite terms vanish in case of an H^div-conforming space for the velocity (for which the normal component of the velocity is continuous across element faces) along with polynomial spaces providing exactly divergence-free velocity fields in the discrete case (such as Raviart–Thomas elements for the velocity combined with a discontinuous pressure one order lower, see Section 2.4.3). The jump penalty term weakly enforces H^div-conformity so that this stabilization can also be denoted as H^div-stabilization (Akbas et al. 2018). Similarly, the divergence penalty term might be interpreted as a weak enforcement of exactly divergence-free velocity spaces and might be seen as a weak realization of Raviart–Thomas elements for the velocity. The reader is referred to Fehn et al. (2019a) for a comparative study discussing similarities and differences between the present stabilized approach and exactly divergence-free H^div-conforming methods. In this context, the use of standard L^2-conforming tensor-product elements used in the present thesis is motivated by the fact that these elements can be easily implemented, are already available in most finite element libraries, and extend naturally to deformed elements by the standard finite element procedure of using high-order mappings. For example, to apply Raviart–Thomas spaces to non-affine elements, other techniques such as a Piola transformation are needed. Another aspect concerns the computational efficiency achievable for the different approaches. In the L^2-conforming case, the mass matrix is block-diagonal and cheap to invert in a matrix-free way, while the mass matrix is no longer block-diagonal in case of Raviart–Thomas elements.

In terms of turbulence modeling, the approach proposed in this thesis is a purely numerical one with inbuilt dissipation mechanisms, commonly known as implicit large-eddy simulation as opposed to explicit large-eddy simulation. The reader is referred to Grinstein et al. (2007), Hickel et al. (2006), Margolin et al. (2006), Margolin and Rider (2002) and references therein for a rationale for implicit LES, and to the textbooks by Pope (2001), Sagaut (2006) for an overview of explicit LES. Note, however, that the present discretization scheme is neither motivated from a modified equation analysis nor is it designed to mimic an explicit sub-grid scale model. The aim of the approach proposed in this thesis is to obtain a parameter-free flow solver. As the stabilized approach is generic, the same numerical method can applied to laminar, transitional, and turbulent flows, where the goal is to achieve accurate results by making optimal use of the polynomial space independently of the type of problem. Generally speaking, it can be considered very advantageous if no calibration of turbulence model parameters (potentially depending on the flow configuration, the Reynolds number, or the polynomial degree of the shape functions used for the simulation) is required. Two requirements can be formulated for a discretization scheme to qualify for implicit LES: (i) the scheme allows robust simulations of under-resolved, high-Reynolds-number or even inviscid flows, and (ii) the scheme provides mechanisms of numerical dissipation as a means to realize the dissipation of scales not resolved by the discretization scheme.

As indicated in Section 2.1.2, the DG community studying the compressible Navier–Stokes equations is more experienced in turbulence simulations, so it might be insightful to have a look at the developments gained there. Since the late 1990s, discontinuous Galerkin methods have been proposed for the compressible Navier–Stokes equations (Bassi and Rebay 1997, 2000, Baumann and Oden 1999, Hartmann and Houston 2005, Lomtev and Karniadakis 1999). These methods have been validated for LES and DNS computations of canonical turbulent flows such as turbulent channel flow in Chapelier et al. (2014), Collis (2002), Ramakrishnan and Collis (2004), Wei and Pollard (2011), Wiart et al. (2015), the Taylor–Green vortex problem in Carton de Wiart et al. (2014), Chapelier et al. (2014), Gassner and Beck (2013), and for geometrically more complex transitional and turbulent flow problems in Beck et al. (2014), Uranga et al. (2011). A rationale for the suitability of high-order DG discretizations for under-resolved turbulent flows is provided by linear dispersion–diffusion analysis, see for example Gassner and Kopriva (2011), Moura et al. (2015), and is used as a motivation for no-model or implicit large-eddy simulation in Beck et al. (2014), Gassner and Beck (2013), Moura et al. (2017a), Wiart et al. (2015). The analysis in Moura et al. (2017a) suggests an improved resolution capability of high-order discretizations, motivating the use of large polynomial degrees also for turbulence simulations where the solution can be considered non-smooth. At the same time, the sharper dissipation behavior of higher polynomial degrees is expected to cause an energy-bump behavior in turbulent energy spectra as compared to low-order discretizations, which is why the use of moderately high polynomial degrees is recommended in Moura et al. (2019). Explicit LES subgrid-scale models such as the standard Smagorinsky model and the dynamic Smagorinsky model have been used in combination with high-order DG discretizations in several publications (Abbá et al. 2015, Chapelier et al. 2016, 2012, de la Llave Plata et al. 2017, Fernandez et al. 2018, Flad and Gassner 2017, Manzanero et al. 2020, Ramakrishnan and Collis 2004, Sengupta et al. 2007, Van Der Bos and Geurts 2010, Zhang et al. 2007). Their outcome is currently inconclusive in terms of an overall benefit achievable by the addition of an explicit model.

For example, the use of explicit models might be primarily motivated as a means to render the discretization scheme stable.

For the compressible Navier–Stokes equations, the effect of aliasing related to nonlinear terms is a central topic, where classical remedies are over-integration (also termed polynomial dealiasing) (Beck et al. 2014, Gassner and Beck 2013) and filtering (Fischer and Mullen 2001, Flad et al. 2016, Gassner and Beck 2013). Despite these remedies, instabilities might still occur especially for under-resolved, inviscid problems and high polynomial degrees (Winters et al. 2018). For this reason, a transition towards discretely energy-conserving formulations took place more recently (Flad and Gassner 2017, Winters et al. 2018), which is realized by collocation-type formulations with summation-by-parts property and suitable split-DG formulations (Gassner 2013, Gassner et al. 2016). The work by Flad and Gassner (2017) suggests to construct a dissipation-free discretization scheme, which is then combined with an explicit subgrid-scale model. However, this work discusses accuracy mainly within the metric of fitting the kinetic energy spectrum in the inertial range for homogeneous turbulence, while other works foster a more holistic view (Manzanero et al. 2020).

All in all, it appears as if evidence still needs to be provided that physically motivated LES subgrid-scale models systematically improve the accuracy of the results as compared to a no-model LES strategy. The investigations in Van Der Bos and Geurts (2010) reveal that improving the accuracy by classical LES models is complicated and that optimal values of model constants depend on several parameters. Hence, the a-priori selection of turbulence model parameters optimal for a wide range of other parameters (spatial resolution, polynomial degree, flow configuration, Reynolds number) appears to be an open issue in this context. For the present incompressible DG solver, a potential benefit by the use of classical eddy-viscosity sub-grid scale models has been assessed critically in Dockhorn (2017), Neumann (2018). These works suggest that demonstrating improved accuracy by an explicit model for a certain example or in a certain metric is not sufficient to cover the complexity of this topic. A closely related aspect deserving more attention is the question whether the improved resolution capabilities that high-order discretizations can provide in an implicit LES setting (Moura et al. 2017a) can be preserved in an explicit LES setting. It is remarkable that the more recent works by Chapelier et al. (2016), Flad and Gassner (2017), Manzanero et al. (2020) arguing for explicit LES in combination with high-order DG discretizations are not able to demonstrate an accuracy advantage of very high-order discretizations compared to low-order or moderately high-order methods, e.g. with $k = 2, 3$. Note, however, that this would be necessary in order to render high-order methods more efficient overall, since other aspects (time step restrictions, iterative solvers, implementation) tend to increase computational costs for large k, i.e., high-order methods are only more efficient than low-order methods if they are sufficiently more accurate. These aspects are discussed in detail in Chapters 4, 5, and 6.

Finally, it appears to be appropriate to dare a view beyond the DG-horizon and to raise the question which benefits an L^2-conforming DG discretization exhibits as compared to H^1-conforming continuous finite element discretizations, which have a long tradition as discretization methods for the incompressible Navier–Stokes equations. Although an investigation of this topic is beyond the scope of this work, some ideas are shared below. It is well known that continuous finite element discretizations require some form of convection-stabilization when convection becomes dominant. Different remedies have been proposed to address this problem. Important classes of methods are filtering (Fischer and Mullen 2001, Fischer et al. 2002, Tufo and Fis-

cher 1999), spectral vanishing viscosity (SVV) (Karamanos and Karniadakis 2000, Kirby and Karniadakis 2002, Kirby and Sherwin 2006), streamline upwind Petrov–Galerkin (SUPG) stabilization (Brooks and Hughes 1982) and many other stabilized methods such as residual-based and local projection stabilizations (Arndt et al. 2015). A huge body of literature is available that interprets many of these methods as variational multiscale methods, see for example the review articles by Ahmed et al. (2017), Gravemeier (2006), Rasthofer and Gravemeier (2018) and references therein. A multitude of works investigated these methods for the simulation of incompressible turbulent flows, exemplarily mentioning two elaborate approaches by Gravemeier et al. (2010), Rasthofer and Gravemeier (2013) that have been proposed after years of experience with stabilized finite element methods. Against this background, DG methods addressing the simulation of incompressible turbulent flows as proposed in this thesis are still in a very early stage. Hence, it would be too early to draw conclusions, but comparative studies need to be performed (some first comparative studies are shown in Section 2.6). Nevertheless, the fact that the problem of convection-stabilization appears to be less of a concern in the DG case clearly justifies to investigate this approach in more detail. Both approaches seem to require stabilization techniques for improved mass conservation in general (grad–div stabilization for continuous Galerkin case, div–div and normal continuity stabilization for discontinuous Galerkin case). In this context, it is interesting that stabilized L^2-conforming discretizations do not suffer from over-stabilization like H^1-conforming continuous finite element discretizations (Akbas et al. 2018). Finally, note that more recent studies highlight the parameter-dependency of the SVV approach (Ferrer et al. 2019, Manzanero et al. 2020).

Often, the development of LES methods concentrates on turbulence modeling aspects in the sense of optimizing the accuracy that can be achieved for a certain spatial resolution (unknown degrees of freedom). However, it would be unrealistic to expect ever-increasing accuracy by the development of new numerical methods for turbulent flows. Decades of LES research revealed that a multitude of methods – achieving a similar level of accuracy when the spatial resolution of the schemes is comparable – is available. Typically, common accuracy limits seem to be reached after having optimized different discretization methods, see also the numerical results shown in Section 2.6.7. In the end, the spatial resolution is the main factor that drives the accuracy of LES. This perspective considers LES as a numerical method that yields improved accuracy when investing more computational effort, and strengthens aspects of computational efficiency of a LES solver. From this perspective, an optimal LES approach appears to be one that is theoretically simple, easy to implement, computationally efficient, and that minimizes parameter-dependency (as a means to predict rather than reproduce results). It is this perspective that motivated the development of new computational methods for turbulent flow simulations in the course of this thesis.

2.5 Fully discrete formulation

This section summarizes the fully-discrete formulation for the temporal discretization schemes discussed in Section 2.3, using the discontinuous Galerkin discretization techniques discussed in Section 2.4.

2.5.1 Coupled solution approach

Combining the discrete-in-time problem (2.28) with the discrete-in-space problem in equations (2.157) and (2.158) yields: Find $\boldsymbol{u}_h^{n+1} \in \mathcal{V}_h^u$, $p_h^{n+1} \in \mathcal{V}_h^p$ such that

$$\begin{aligned} m_{h,u}^e\left(\boldsymbol{v}_h, \frac{\gamma_0^n \boldsymbol{u}_h^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}_h^{n-i}}{\Delta t_n}\right) + c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}; \boldsymbol{g}_u(t_{n+1})\right) \\ + a_{\mathrm{D},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}) + a_{\mathrm{C},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}; \boldsymbol{g}_u(t_{n+1})) \\ + v_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}; \boldsymbol{g}_u(t_{n+1}), \boldsymbol{h}_u(t_{n+1})\right) \\ + g_h^e\left(\boldsymbol{v}_h, p_h^{n+1}; g_p(t_{n+1})\right) - b_h^e\left(\boldsymbol{v}_h, \boldsymbol{f}(t_{n+1})\right) = 0 \,, \end{aligned} \tag{2.187}$$

$$-d_h^e(q_h, \boldsymbol{u}_h^{n+1}; \boldsymbol{g}_u(t_{n+1})) = 0 \,, \tag{2.188}$$

for all $(\boldsymbol{v}_h, q_h) \in \mathcal{V}_{h,e}^u \times \mathcal{V}_{h,e}^p$ and for all elements $e = 1, ..., N_{\mathrm{el}}$. As initial guesses for the iterative solution of algebraic systems of equations, $\boldsymbol{u}_h^{n+1,(0)} = \sum_{i=0}^{J-1} \beta_i^n \boldsymbol{u}_h^{n-i}$ and $p_h^{n+1,(0)} = \sum_{i=0}^{J-1} \beta_i^n p_h^{n-i}$ are used.

Motivated from the perspective of computational efficiency, another solution strategy is proposed, namely to apply the divergence and continuity penalty terms separately in a postprocessing step instead of adding these terms to the monolithic system. This procedure is similar to the idea proposed in Cockburn et al. (2005) of postprocessing the velocity field to obtain a divergence-free velocity. In a first step, an intermediate velocity $\hat{\boldsymbol{u}}_h$ is calculated by solving the coupled system of equations without additional stabilization terms

$$\begin{aligned} m_{h,u}^e\left(\boldsymbol{v}_h, \frac{\gamma_0^n \hat{\boldsymbol{u}}_h - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}_h^{n-i}}{\Delta t_n}\right) + c_h^e\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h; \boldsymbol{g}_u(t_{n+1})\right) \\ + v_h^e\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h; \boldsymbol{g}_u(t_{n+1}), \boldsymbol{h}_u(t_{n+1})\right) \\ + g_h^e\left(\boldsymbol{v}_h, p_h^{n+1}; g_p(t_{n+1})\right) - b_h^e\left(\boldsymbol{v}_h, \boldsymbol{f}(t_{n+1})\right) = 0 \,, \end{aligned} \tag{2.189}$$

$$-d_h^e(q_h, \hat{\boldsymbol{u}}_h; \boldsymbol{g}_u(t_{n+1})) = 0 \,. \tag{2.190}$$

Subsequently, the divergence and continuity penalty terms are applied in a postprocessing step to obtain the final velocity $\boldsymbol{u}_h^{n+1}$

$$\begin{aligned} m_{h,u}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}) + a_{\mathrm{D},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1})\,\Delta t_n + a_{\mathrm{C},h,\mathrm{hom}}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1})\,\Delta t_n = \\ m_{h,u}^e\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h\right) - a_{\mathrm{C},h,\mathrm{inhom}}^e(\boldsymbol{v}_h, \boldsymbol{g}_u(t_{n+1}))\,\Delta t_n \,. \end{aligned} \tag{2.191}$$

For this solution strategy, the following initial guesses are used for the iterative solution of algebraic systems of equations, $\hat{\boldsymbol{u}}_h^{(0)} = \sum_{i=0}^{J-1} \beta_i^n \boldsymbol{u}_h^{n-i}$, $p_h^{n+1,(0)} = \sum_{i=0}^{J-1} \beta_i^n p_h^{n-i}$, and $\boldsymbol{u}_h^{n+1,(0)} = \hat{\boldsymbol{u}}_h$. The motivation for this approach is that state-of-the-art preconditioning strategies developed for the saddle-point type incompressible Navier–Stokes problem can be directly applied to the coupled system of equations, while the complexity associated to the penalty terms is treated separately. The postprocessing step is comparably cheap to solve since an appropriate scaling of the divergence and continuity penalty terms by the time step size renders the inverse mass matrix operator an effective and computationally efficient preconditioner, especially in the L^2-conforming context considered here where the mass matrix is block-diagonal. As a side note,

a development of block-preconditioners for the coupled solution approach taking into account additional stabilization terms has been considered in Heister and Rapin (2013) in the context of conforming finite element discretizations with grad–div stabilization. Aspects of preconditioning are discussed in detail in Chapter 5. In terms of accuracy, the postprocessing approach was found to not perturb accuracy in a significant way and to maintain optimal rates of convergence in space and time for smooth problems (Fehn et al. 2018b).

Independently of how the penalty terms are treated, the convective term can be formulated explicitly in time by replacing

$$c_h^e\left(\boldsymbol{v}_h,\boldsymbol{u}_h^{n+1};\boldsymbol{g}_u(t_{n+1})\right)\rightarrow\sum_{i=0}^{J-1}\beta_i^n c_h^e\left(\boldsymbol{v}_h,\boldsymbol{u}_h^{n-i};\boldsymbol{g}_u(t_{n-i})\right)\ , \tag{2.192}$$

which is also motivated from the point of view of computational costs. The resulting coupled system of equations is then an unsteady Stokes problem, so that the solution of a nonlinear system of equations is avoided.

2.5.2 Dual splitting scheme

This section summarizes the fully-discrete formulation using the dual splitting scheme presented in Section 2.3.3 and the DG formulation derived in Section 2.4.2.

2.5.2.1 Convective step

The weak DG formulation of the convective step (2.33) is given as follows: Find $\hat{\boldsymbol{u}}_h\in\mathcal{V}_h^u$ such that

$$\begin{aligned}m_{h,u}^e\left(\boldsymbol{v}_h,\frac{\gamma_0^n\hat{\boldsymbol{u}}_h-\sum_{i=0}^{J-1}\alpha_i^n\boldsymbol{u}_h^{n-i}}{\Delta t_n}\right)=&-\sum_{i=0}^{J-1}\beta_i^n c_h^e\left(\boldsymbol{v}_h,\boldsymbol{u}_h^{n-i},\boldsymbol{g}_u(t_{n-i})\right)\\&+b_h^e\left(\boldsymbol{v}_h,\boldsymbol{f}(t_{n+1})\right)\ ,\end{aligned} \tag{2.193}$$

for all $\boldsymbol{v}_h\in\mathcal{V}_{h,e}^u$ and for all elements $e=1,...,N_{\text{el}}$.

2.5.2.2 Pressure step

The weak formulation of the pressure Poisson equation (2.37) reads: Find $p_h^{n+1}\in\mathcal{V}_h^p$ such that

$$l_{h,\text{hom}}^e\left(q_h,p_h^{n+1}\right)=-\frac{\gamma_0^n}{\Delta t_n}d_h^e\left(q_h,\hat{\boldsymbol{u}}_h,\boldsymbol{g}_{\hat{u}}(t_{n+1})\right)-l_{h,\text{inhom}}^e\left(q_h,g_p(t_{n+1}),h_p(t_{n+1})\right)\ , \tag{2.194}$$

for all $q_h\in\mathcal{V}_{h,e}^p$ and for all elements $e=1,...,N_{\text{el}}$. As an accurate initial guess for the iterative solver, $p_h^{n+1,(0)}=\sum_{i=0}^{J-1}\beta_i^n p_h^{n-i}$ is used. The boundary condition $\boldsymbol{g}_{\hat{u}}(t_{n+1})$ for the intermediate velocity required to evaluate the discrete divergence operator on the right-hand side of the pressure Poisson equation is defined in equation (2.42), see also Fehn et al. (2017) for the importance of the velocity–pressure coupling terms regarding small-time-steps stability. The boundary values g_p and h_p in the above pressure Poisson equation are defined in equation (2.39) and

equation (2.38), respectively. To evaluate $\boldsymbol{g}_{\hat{u}}(t_{n+1})$ and $h_p(t_{n+1})$ on the right-hand side of equation (2.194) according to the boundary conditions (2.42) and (2.40), the convective term and the viscous term have to be calculated on $\partial\Omega_e$ as a function of the approximate velocity solution $\boldsymbol{u}_h$ on element e. In the discrete case and if the divergence formulation of the convective term is used, the convective term is calculated as

$$\nabla \cdot \boldsymbol{F}_{\mathrm{c}}\left(\boldsymbol{u}_h\right) = \boldsymbol{u}_h\left(\nabla \cdot \boldsymbol{u}_h\right) + \left(\nabla \boldsymbol{u}_h\right) \cdot \boldsymbol{u}_h \ . \tag{2.195}$$

The viscous term in equation (2.40) involves second derivatives and is calculated in two steps, so that the computation of second derivatives is replaced by a sequence of first derivatives. The vorticity $\boldsymbol{\omega}_h \in \mathcal{V}_h^u$ in equation (2.40) is calculated by a local L^2-projection

$$\left(\boldsymbol{v}_h, \boldsymbol{\omega}_h\right)_{\Omega_e} = \left(\boldsymbol{v}_h, \nabla \times \boldsymbol{u}_h\right)_{\Omega_e} \ . \tag{2.196}$$

The viscous term is then evaluated by calculating the curl of the vorticity $\boldsymbol{\omega}_h$ on the respective boundary.

2.5.2.3 Projection step

In elementwise notation, the weak form of the projection step (2.43) is to find $\hat{\hat{\boldsymbol{u}}}_h \in \mathcal{V}_h^u$ such that

$$m_{h,u}^e(\boldsymbol{v}_h, \hat{\hat{\boldsymbol{u}}}_h) = m_{h,u}^e\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h\right) - \frac{\Delta t_n}{\gamma_0^n} g_h^e\left(\boldsymbol{v}_h, p_h^{n+1}, g_p\left(t_{n+1}\right)\right) \ , \tag{2.197}$$

for all $\boldsymbol{v}_h \in \mathcal{V}_{h,e}^u$ and for all elements $e = 1, ..., N_{\mathrm{el}}$. A pressure Dirichlet boundary condition prescribed on Γ_h^{N}, see equation (2.39), is required to evaluate the discrete pressure gradient. As emphasized in Fehn et al. (2017), this is in contrast to formulations that do not perform integration-by-parts for the velocity–pressure coupling terms and that suffer from instabilities for small time step sizes.

2.5.2.4 Viscous step

The viscous step computes another intermediate velocity $\hat{\hat{\hat{\boldsymbol{u}}}}_h$ by solving a Helmholtz-like equation. As for the pressure Poisson equation, inhomogeneous boundary face integrals are shifted to the right-hand side to obtain the following weak formulation of equation (2.45): Find $\hat{\hat{\hat{\boldsymbol{u}}}}_h \in \mathcal{V}_h^u$ such that

$$\begin{aligned} m_{h,u}^e\left(\boldsymbol{v}_h, \frac{\gamma_0^n}{\Delta t_n}\hat{\hat{\hat{\boldsymbol{u}}}}_h\right) + v_{h,\mathrm{hom}}^e\left(\boldsymbol{v}_h, \hat{\hat{\hat{\boldsymbol{u}}}}_h\right) = &+ m_{h,u}^e\left(\boldsymbol{v}_h, \frac{\gamma_0^n}{\Delta t_n}\hat{\hat{\boldsymbol{u}}}_h\right) \\ &- v_{h,\mathrm{inhom}}^e(\boldsymbol{v}_h, \boldsymbol{g}_u(t_{n+1}), \boldsymbol{h}_u(t_{n+1})) \ , \end{aligned} \tag{2.198}$$

for all $\boldsymbol{v}_h \in \mathcal{V}_{h,e}^u$ and for all elements $e = 1, ..., N_{\mathrm{el}}$. An extrapolated velocity serves as initial guess for the iterative solver, $\hat{\hat{\hat{\boldsymbol{u}}}}_h^{(0)} = \sum_{i=0}^{J-1} \beta_i^n \boldsymbol{u}_h^{n-i}$.

2.5.2.5 Penalty step

The divergence and continuity penalty terms are treated in the final step: Find $\boldsymbol{u}_h^{n+1} \in \mathcal{V}_h^u$ such that

$$\begin{aligned} m_{h,u}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}) + a_{\mathrm{D},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1})\,\Delta t_n + a_{\mathrm{C},h,\mathrm{hom}}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1})\,\Delta t_n = \\ m_{h,u}^e\left(\boldsymbol{v}_h, \hat{\hat{\boldsymbol{u}}}_h\right) - a_{\mathrm{C},h,\mathrm{inhom}}^e(\boldsymbol{v}_h, \boldsymbol{g}_u(t_{n+1}))\,\Delta t_n\ . \end{aligned} \tag{2.199}$$

for all $\boldsymbol{v}_h \in \mathcal{V}_{h,e}^u$ and for all elements $e = 1, ..., N_{\mathrm{el}}$. The converged solution of the viscous step is used as initial guess for the iterative solution of the penalty step, $\boldsymbol{u}_h^{n+1,(0)} = \hat{\hat{\boldsymbol{u}}}_h$. The procedure of treating the penalty terms in a final step slightly differs from the procedure originally proposed in Fehn et al. (2018b), Krank et al. (2017) where these terms have been added to the projection step, equation (2.197). As noted in Fehn et al. (2021a), a formulation that applies the continuity penalty term on all faces including domain boundaries with the imposition of boundary conditions such as $\boldsymbol{g}_u(t_{n+1})$ should be used for reasons of stability. Since this boundary condition would be inconsistent and would prevent high-order accuracy in time if imposed for the intermediate velocity $\hat{\hat{\boldsymbol{u}}}_h$ in the projection step, the penalty terms are applied in a separate postprocessing step, equation (2.199).

2.5.3 Pressure-correction scheme

This section summarizes the fully-discrete formulation using the pressure-correction scheme presented in Section 2.3.4 and the DG formulation derived in Section 2.4.2.

2.5.3.1 Momentum step

The discontinuous Galerkin discretization of the time discrete momentum equation (2.48) to be solved in the first sub-step of the pressure-correction scheme reads: Find $\hat{\boldsymbol{u}}_h \in \mathcal{V}_h^u$ such that

$$\begin{aligned} m_{h,u}^e\left(\boldsymbol{v}_h, \frac{\gamma_0^n \hat{\boldsymbol{u}}_h - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}_h^{n-i}}{\Delta t_n}\right) + c_h^e\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h, \boldsymbol{g}_u(t_{n+1})\right) \\ + v_h^e\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h, \boldsymbol{g}_u(t_{n+1}), \boldsymbol{h}_u(t_{n+1})\right) \\ + \sum_{i=0}^{J_p-1} \beta_i^n g_h^e\left(\boldsymbol{v}_h, p_h^{n-i}, g_p(t_{n-i})\right) - b_h^e\left(\boldsymbol{v}_h, \boldsymbol{f}(t_{n+1})\right) = 0\ , \end{aligned} \tag{2.200}$$

for all $\boldsymbol{v}_h \in \mathcal{V}_{h,e}^u$ and for all elements $e = 1, ..., N_{\mathrm{el}}$. An extrapolated velocity is used as initial guess for the iterative solver, $\hat{\boldsymbol{u}}_h^{(0)} = \sum_{i=0}^{J-1} \beta_i^n \boldsymbol{u}_h^{n-i}$. The boundary conditions $\boldsymbol{g}_u$ and $\boldsymbol{h}_u$ prescribed for the intermediate velocity $\hat{\boldsymbol{u}}_h$ are defined in equations (2.51) and (2.52), respectively. When solving the incremental formulation of the pressure-correction scheme, a boundary condition g_p defined in equation (2.53) has to be prescribed for the pressure in order to evaluate the discrete pressure gradient operator.

As for the coupled solution approach described in Section 2.5.1, the convective term can be formulated explicitly in time

$$c_h^e\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h; \boldsymbol{g}_u(t_{n+1})\right) \rightarrow \sum_{i=0}^{J-1} \beta_i^n c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n-i}; \boldsymbol{g}_u(t_{n-i})\right) \ , \tag{2.201}$$

resulting in a linear system of equations to be solved in the momentum step.

2.5.3.2 Pressure step

The discontinuous Galerkin formulation of the pressure Poisson equation (2.57) is given as: Find $\phi_h \in \mathcal{V}_h^p$ such that

$$l_{h,\text{hom}}^e\left(q_h, \phi_h^{n+1}\right) = -\frac{\gamma_0^n}{\Delta t_n} d_h^e\left(q_h, \hat{\boldsymbol{u}}_h, \boldsymbol{g}_u(t_{n+1})\right) - l_{h,\text{inhom}}^e\left(q_h, g_\phi(t_{n+1}), h_\phi(t_{n+1})\right) \ , \tag{2.202}$$

for all $q_h \in \mathcal{V}_{h,e}^p$ and for all elements $e = 1, ..., N_{\text{el}}$. The initial guess used for the iterative solver is $\phi_h^{n+1,(0)} = \sum_{i=0}^{J-1} \beta_i^n p_h^{n-i} - \sum_{i=0}^{J_p-1} \beta_i^n p_h^{n-i}$. The boundary values g_ϕ and h_ϕ are defined in equations (2.59) and (2.58), respectively. The approximate pressure solution p_h^{n+1} at time t_{n+1} is obtained from equation (2.60). In elementwise notation, the weak formulation of this pressure update reads: Find $p_h^{n+1} \in \mathcal{V}_h^p$ such that

$$m_{h,p}^e\left(q_h, p_h^{n+1}\right) = m_{h,p}^e\left(q_h, \phi_h^{n+1} + \sum_{i=0}^{J_p-1} \beta_i^n p_h^{n-i}\right) - \chi\nu \ d_h^e\left(q_h, \hat{\boldsymbol{u}}_h, \boldsymbol{g}_u\left(t_{n+1}\right)\right) \ , \tag{2.203}$$

for all $q_h \in \mathcal{V}_{h,e}^p$ and for all elements $e = 1, ..., N_{\text{el}}$. The pressure mass matrix operator in the above equation is $m_{h,p}^e\left(q_h, p_h\right) = \left(q_h, p_h\right)_{\Omega_e}$. In contrast to the dual splitting scheme, the intermediate velocity field $\hat{\boldsymbol{u}}$ fulfills the velocity Dirichlet boundary condition $\boldsymbol{g}_u$, which can be seen from equation (2.51). Consequently, this boundary condition is used to evaluate the discrete divergence operator applied to the intermediate velocity $\hat{\boldsymbol{u}}_h$ on the right-hand side of equations (2.202) and (2.203).

2.5.3.3 Projection step

The discontinuous Galerkin formulation of the projection step can be stated as: Find $\hat{\hat{\boldsymbol{u}}}_h \in \mathcal{V}_h^u$ such that

$$m_{h,u}^e\left(\boldsymbol{v}_h, \hat{\hat{\boldsymbol{u}}}_h\right) = m_{h,u}^e\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h\right) - \frac{\Delta t_n}{\gamma_0^n} g_h^e\left(\boldsymbol{v}_h, \phi_h^{n+1}, g_\phi\left(t_{n+1}\right)\right) \ , \tag{2.204}$$

for all $\boldsymbol{v}_h \in \mathcal{V}_{h,e}^u$ and for all elements $e = 1, ..., N_{\text{el}}$. The pressure boundary condition g_ϕ is defined in equation (2.59).

2.5.3.4 Penalty step

Finally, the divergence and continuity penalty terms are applied: Find $\boldsymbol{u}_h^{n+1} \in \mathcal{V}_h^u$ such that

$$\begin{aligned} m_{h,u}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}) + a_{\mathrm{D},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1})\,\Delta t_n + a_{\mathrm{C},h,\mathrm{hom}}^e(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1})\,\Delta t_n = \\ m_{h,u}^e\left(\boldsymbol{v}_h, \hat{\hat{\boldsymbol{u}}}_h\right) - a_{\mathrm{C},h,\mathrm{inhom}}^e(\boldsymbol{v}_h, \boldsymbol{g}_u(t_{n+1}))\,\Delta t_n\ . \end{aligned} \tag{2.205}$$

for all $\boldsymbol{v}_h \in \mathcal{V}_{h,e}^u$ and for all elements $e = 1, ..., N_{\mathrm{el}}$, using $\boldsymbol{u}_h^{n+1,(0)} = \hat{\hat{\boldsymbol{u}}}_h$ as initial guess for the iterative solver. The left-hand side of equation (2.204) appears on the right-hand side of equation (2.205). Hence, both equations can be combined to a single equation, as written in (Fehn et al. 2018b, equation (B.9)).

2.5.4 Impact of operator splitting on inf–sup stability

Several works use equal-order polynomials for the approximation of velocity and pressure unknowns for L^2-conforming discontinuous Galerkin discretizations of the incompressible Navier–Stokes equations, see for example Bassi et al. (2006), Chalmers et al. (2019), Ferrer and Willden (2011), Hesthaven and Warburton (2007), Klein et al. (2015), Krank et al. (2017), Shahbazi et al. (2007). However, one might expect that equal-order formulations suffer from inf–sup instabilities. The authors of Bassi et al. (2006) argue that their approach is a stabilized one due to the artificial compressibility flux, and an explicit pressure stabilization is used in Klein et al. (2015). For these equal-order formulations, sub-optimal rates of convergence are obtained for the pressure in Bassi et al. (2006), Chalmers et al. (2019), Klein et al. (2015), Shahbazi et al. (2007), while optimal rates of convergence are reported in Krank et al. (2017) where the dual splitting scheme is used. Mixed-order formulations typically achieve optimal convergence rates. Noticeable in this context is the use of equal-order formulations in combination with splitting-type solution techniques. Depending on the type of operator splitting, an inf–sup stabilizing term is introduced by the projection method, which could explain the use of equal-order formulations. As argued in Guermond et al. (2006), the inf–sup condition is still relevant for projection methods even if the pressure Poisson equation and Helmholtz equation for the velocity are solvable independently of the polynomial spaces used to represent the velocity and pressure solutions. Instead, the corresponding steady-state Stokes problem of projection methods is the decisive metric to evaluate the need of the inf–sup condition.

Following Ferrer et al. (2014), Guermond et al. (2006), this section briefly derives the steady-state Stokes equations for the dual splitting scheme and the pressure-correction scheme from the equations shown in Section 2.3, which serves as a basis for the interpretation and explanation of numerical results shown in Section 2.6.3. For ease of notation, the equations are considered at the level of differential operators, but similar relations can be derived for discretized operators or matrix formulations.

For the dual splitting scheme, the following system of equations can be derived for the steady Stokes problem

$$\begin{pmatrix} -\nabla\cdot\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) & +\nabla p \\ -\nabla\cdot\boldsymbol{u} & +\frac{\Delta t}{\gamma_0}\nabla^2 p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \frac{\Delta t}{\gamma_0}\nabla\cdot\boldsymbol{f} \end{pmatrix}\ . \tag{2.206}$$

The first equation is obtained by adding equations (2.33), (2.35) and (2.45), neglecting the convective term in equation (2.33), assuming that a steady state $\boldsymbol{u}^{n+1} = \boldsymbol{u}^n = ... = \boldsymbol{u}^{n-J+1} = \boldsymbol{u}$ is reached, and using the fact that the BDF time integration constants fulfill the property $\gamma_0 = \sum_{i=0}^{J-1} \alpha_i$. The second equation is obtained by taking the divergence of equation (2.35) and inserting equations (2.36) and (2.33). Equation (2.206) highlights that the dual splitting scheme introduces an inf–sup stabilizing term $\Delta t/\gamma_0 \nabla^2 p$ as compared to the steady-state Stokes problem for the coupled solution approach. According to this relation, the impact of the stabilization can be expected to diminish for small time step sizes Δt.

Similarly, the following system of equations can be derived for the pressure-correction scheme

$$\begin{pmatrix} -\nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\hat{\boldsymbol{u}}) + \nabla(\chi\nu\nabla\cdot\hat{\boldsymbol{u}}) & +\nabla p \\ -\nabla\cdot\hat{\boldsymbol{u}} + \frac{\Delta t}{\gamma_0}\nabla\cdot(\nabla(\chi\nu\nabla\cdot\hat{\boldsymbol{u}})) & +\frac{\Delta t}{\gamma_0}\left(1 - \sum_{i=0}^{J_p-1}\beta_i\right)\nabla^2 p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix} . \tag{2.207}$$

To derive these equations, one assumes that the solution reaches a steady state, $\boldsymbol{u}^{n+1} = \boldsymbol{u}^n = ... = \boldsymbol{u}^{n-J+1} = \boldsymbol{u}$ and $p^{n+1} = p^n = ... = p^{n-J_p+1} = p$, and uses the fact that the time integration constants fulfill $\gamma_0 = \sum_{i=0}^{J-1} \alpha_i$ and $\sum_{i=0}^{J_p-1} \beta_i = 1$. The first equation is obtained by adding equations (2.48) and (2.54), neglecting the convective term in equation (2.48), and replacing ϕ^{n+1} by equation (2.56). The second equation is derived by taking the divergence of equation (2.54) and inserting equations (2.55) and (2.56). Equation (2.207) highlights that the pressure-correction scheme introduces an inf–sup stabilizing term in case of the non-incremental formulation ($J_p = 0$), but not in case of the incremental formulation because of $1 - \sum_{i=0}^{J_p-1} \beta_i = 0$ for $J_p \geq 1$. The incremental pressure-correction scheme can be expected to show an inf–sup behavior similar to the coupled solution approach. In this context, the algebraic splitting scheme in Shahbazi et al. (2007) can be expected to behave like an incremental pressure-correction scheme.

2.5.5 Numerical integration

Volume and surface integrals are transformed to the reference element $\tilde{\Omega}_e$ where these integrals are computed numerically using Gaussian quadrature, the standard quadrature formula in finite element methods. In one space dimension, the integral of a polynomial $p^n(\xi) = \alpha_0\xi^0 + \ldots + \alpha_n\xi^n$ of degree n is approximated as

$$\int_{[0,1]} p^n(\xi)\mathrm{d}\xi = \sum_{i=1}^{n_q} p(\xi_q) w_q . \tag{2.208}$$

where $\{\xi_q\}_{q=1}^{n_q}$ is the set of one-dimensional quadrature points with corresponding quadrature weights w_q according to the Legendre–Gauss quadrature rule. In higher space dimensions, the quadrature rule is constructed as a tensor product of the one-dimensional quadrature rule, with quadrature points $\boldsymbol{\xi}_q = \boldsymbol{\xi}_{q_1\ldots q_d} = (\xi_{1,q_1}, \ldots, \xi_{d,q_d})^{\mathsf{T}}$ and weights $w_q = w_{q_1\ldots q_d} = w_{1,q_1} \cdot \ldots \cdot w_{d,q_d}$. The quadrature rule has $2n_q$ degrees of freedom that can be chosen such that polynomials of degree n with $n + 1 = 2n_q$ coefficients are integrated exactly. A collocation of nodes and quadrature points frequently used in the spectral element context is not pursued here, in order to take advantage of the improved accuracy of the Legendre–Gauss quadrature as compared to a collocated Legendre–Gauss–Lobatto quadrature (Durufle et al. 2009, Kronbichler 2021a).

The number of quadrature points selected for the integration of the different terms of the weak formulation is chosen by requiring that integrals are computed exactly on affine element geometries where the Jacobian is constant throughout the element. This implies that quadrature errors are introduced on deformed elements due to the transformation of the integral from physical space to reference space, which gives rise to additional metric terms. Typically, the geometry representation is smooth as opposed to the under-resolution of flow features, so that geometry-related errors can be expected to be small under these circumstances (Kirby and Karniadakis 2003, Mengaldo et al. 2015). Having said that, $n_q = k_u + 1$ quadrature points are used for the velocity mass matrix term, which contains polynomials of degree $2k_u$ on affine elements due to the multiplication of test and solution functions. The same number of quadrature points is used for the viscous term, the pressure gradient term, the velocity divergence term, the body force term, and the divergence and continuity penalty terms. The convective term includes quadratic non-linearities. For this reason, $n_q = \lceil \frac{3k_u+1}{2} \rceil$ quadrature points are used for the convective term, which is also known as the 3/2-rule. This quadrature rule is also used for boundary face integrals containing the convective term in case of splitting methods. The Laplace operator for the pressure as well as the pressure mass matrix operator occurring in projection-type solution methods are integrated with $k_p + 1$ quadrature points.

Remark 2.12 *A collocation between nodal polynomials and quadrature is widespread. A main motivation for a collocation approach is to obtain a diagonal mass matrix (also termed mass-lumping) in order to obtain a mass matrix that is cheap to invert (particularly for continuous spectral element methods). However, in the context of DG methods, the mass matrix is block-diagonal already for a non-collocation approach and can therefore be inverted easily in an element-by-element fashion. Moreover, the mass matrix can be inverted in a matrix-free way by exploiting the tensor-product structure of the elements (Kronbichler et al. 2016). When characterizing this operation from a computer science point of view, the application of the forward and inverse (block-diagonal) mass operator is a memory-bound operation for moderately high-polynomial degrees on curent hardware, i.e., the speed of this operation is limited by the speed at which the input and output vectors can be transferred from memory, see for example Fehn et al. (2018a, 2019c). Therefore, a diagonal mass matrix can not be faster.*

For operators other than the mass matrix, a collocated basis would reduce operations for volume integrals since the solution does not have to be interpolated into the quadrature points. At the same time, the evaluation of face integrals might become more expensive in case of a collocation basis, especially for operators with first derivatives that need to evaluate only the solution (but not its gradient) on a face of the element. For a collocated basis with Legendre–Gauss quadrature points, all degrees of freedom of the element contribute to the solution on a face of the element. Instead, for a collocated Gauss–Lobatto basis, only one layer of nodes needs to be touched to evaluate the solution on a face of the element. While a collocated Gauss–Lobatto basis appears advantageous from this perspective, it results in a loss of accuracy due to inexact integration on affine elements. For operators with second derivatives, the weak formulation also needs to evaluate the gradient of the solution on the faces of an element so that all degrees of freedom have to be touched when using typical Lagrange polynomials. A Hermite-like basis is proposed in Kronbichler et al. (2019) in order to evaluate face integrals efficiently for operators with second derivatives. It should be mentioned that arguments regarding computational efficiency invoked here are subject to changes and recent developments in computer hardware.

Finally, the conditioning of linear systems of equations in combination with certain multigrid smoothers contributes to the discussion of an optimal basis. The interested reader is referred to Kronbichler and Kormann (2019), Kronbichler and Wall (2018), Kronbichler et al. (2019) for an in-depth discussion of these aspects. In summary, the present work uses Gauss–Lobatto polynomials for good conditioning and fast convergence of iterative solvers, and Gauss quadrature points for an exact evaluation of integrals on affine geometries (recognizing that the costs of arithmetic operations for interpolation are moderate on current hardware, see Chapter 4).

2.5.6 CFL condition

An explicit time integration of some of the terms of the equations introduces a time step restriction that limits the range of time step sizes for which the eigenvalue spectrum of the discretization scheme lies within the stability region of the time integrator. For the convective term of transport equations such as the incompressible Navier–Stokes equations considered here, this time step restriction is known as the Courant–Friedrichs–Lewy (CFL) condition. In the course of this thesis, a distinction is made between a global CFL condition, which requires a-priori estimates of the mesh size and the maximum flow velocity, and a local CFL condition, which evaluates the critical time step size locally in an element-by-element fashion and selects the time step size as the minimum over all elements. The global CFL condition is defined as Fehn et al. (2018b), Hesthaven and Warburton (2007), Shahbazi et al. (2007)

$$\Delta t = \frac{\mathrm{Cr}}{k_u^r} \frac{h_{\min}}{\|\boldsymbol{u}_h\|_{\max}} \,, \tag{2.209}$$

where Cr is the Courant number, $h_{\min}$ the minimum element length, and $\|\boldsymbol{u}_h\|_{\max}$ an estimate of the maximum velocity occurring in the flow domain. This equation shows the well-known linear dependency of the time step size on the mesh size h. In the context of high-order discretizations considered here, the exponent r describes the dependency of the critical time step size on the polynomial degree of the shape functions, where values of $r = 1$ or $r = 2$ can typically be found in the literature (Chalmers et al. 2019, Hesthaven and Warburton 2007, Shahbazi et al. 2007). Stability is obtained for $\mathrm{Cr} < \mathrm{Cr}_{\mathrm{crit}}$, where $\mathrm{Cr}_{\mathrm{crit}}$ is a parameter related to the discretization and Cr a parameter of the solver, typically chosen close to the critical value for efficiency. For reasons of practicability and efficiency, it is desirable that $\mathrm{Cr}_{\mathrm{crit}}$ is a constant independent of k_u, which motivates to fit the exponent r in a way that a fixed value of $\mathrm{Cr}_{\mathrm{crit}}$ can be used without a need to readjust this value when considering a different polynomial degree k_u. In the work by Fehn et al. (2018a) studying the present discretization approach, it was found that a value of $r = 1.5$ models the dependency on the polynomial degree very well for a wide range of polynomial degrees $2 \leq k_u \leq 15$, and is therefore used in this thesis.

The above global CFL condition might introduce inaccuracies since the maximum velocity is difficult to estimate and since the maximum velocity does not occur in the element of minimum size, i.e., equation (2.209) leads to a conservative estimate of the time step size for more complex problems. This motivates the use of a local CFL condition applied in combination with adaptive time-stepping (Karniadakis and Sherwin 2013)

$$\Delta t = \min_{e=1,\ldots,N_{\mathrm{el}}} \Delta t_e, \; \Delta t_e = \min_{q=1,\ldots,N_{q,e}} \frac{\mathrm{Cr}}{k_u^r} \left. \frac{h}{\|\boldsymbol{u}_h\|} \right|_{q,e} \,, \tag{2.210}$$

where $\left.\frac{\|\boldsymbol{u}_h\|}{h}\right|_{q,e} = \|\boldsymbol{J}^{-1}\boldsymbol{u}_h\|_{q,e}$, $J_{ij} = \partial x_i/\partial \xi_j$, is a local velocity-to-mesh-size ratio evaluated at quadrature point q of element e. This CFL condition evaluates the local flow velocity and, therefore, allows to operate close to the true stability limit for transient problems. Note that this time step criterion is still a global one in the sense that all elements are advanced with the same time step size, as opposed to local time-stepping techniques.

2.5.7 From weak forms to algebraic systems of equations

This section introduces the matrix/vector notation corresponding to the fully discrete formulations presented in Sections 2.5.1, 2.5.2, and 2.5.3, in order to present the resulting linear and nonlinear systems of equations in a compact way and in order to introduce notation for subsequent chapters. Considering a discretized operator $a_h(\boldsymbol{v}_h, \boldsymbol{u}_h)$ that is linear in the unknowns $\boldsymbol{u}_h$ and that can be separated into a homogeneous and an inhomogeneous part, the following notation is introduced

$$\mathbf{a}\left(\mathbf{u}\right) = \mathbf{a}_{\mathsf{hom}}\left(\mathbf{u}\right) + \mathbf{a}_{\mathsf{inhom}} = \mathbf{A}\mathbf{u} + \mathbf{a}_{\mathsf{inhom}} \ , \tag{2.211}$$

where $\mathbf{a}, \mathbf{u} \in \mathbb{R}^N$ and $\mathbf{A} \in \mathbb{R}^{N\times N}$ with the problem size $N = N_{\mathrm{DoFs}}$ (total number of degrees of freedom). Note that the matrix formulation $\mathbf{A}\mathbf{u}$ is used since this is the standard notation in linear algebra, but that matrices are typically never build or assembled during the solution process since matrix-free methods are considered in the course of this thesis, see Chapter 4. Instead, the matrix-vector product should be understood as an application of the respective homogeneous operator $\mathbf{a}_{\mathsf{hom}}\left(\mathbf{u}\right)$ implemented in a matrix-free way. Nonlinear systems of equations are generally solved by Newton's method in this work. The solution of linear(ized) system of equations is based on state-of-the-art iterative solution techniques (Krylov methods) with suitable preconditioners. This topic is considered as black-box in this chapter and is postponed to Chapter 5 dealing with the efficient solution and preconditioning of linear systems of equations.

2.5.7.1 Newton's method for nonlinear systems of equations

To solve a nonlinear residual equation of the form

$$\mathbf{r}\left(\mathbf{u}\right) = \mathbf{0}$$

by Newton's method, the nonlinear problem is solved iteratively by solving a sequence of linearized systems of equations. To derive this method, consider the Taylor series expansion of the nonlinear residual $\mathbf{r}\left(\mathbf{u}\right)$ around an initial or approximate solution $\mathbf{u}_{\mathsf{lin}}$

$$\mathbf{r}\left(\mathbf{u}\right) = \mathbf{r}\left(\mathbf{u}_{\mathsf{lin}}\right) + \left.\frac{\partial \mathbf{r}\left(\mathbf{u}\right)}{\partial \mathbf{u}}\right|_{\mathbf{u}_{\mathsf{lin}}} \left(\mathbf{u} - \mathbf{u}_{\mathsf{lin}}\right) + \ldots \stackrel{!}{=} 0 \ ,$$

where only the linear term is considered and higher-order terms are neglected (linearization). This leads to the linearized problem

$$\left.\frac{\partial \mathbf{r}\left(\mathbf{u}\right)}{\partial \mathbf{u}}\right|_{\mathbf{u}_{\mathsf{lin}}} \left(\mathbf{u} - \mathbf{u}_{\mathsf{lin}}\right) = -\mathbf{r}\left(\mathbf{u}_{\mathsf{lin}}\right) \ ,$$

Algorithm 2.1 Newton solver

function NEWTON($\mathbf{u}$)
  Initialization $k = 0$, $\mathbf{u}^{(0)} = \mathbf{u}$
  Evaluate nonlinear residual $\mathbf{r}^{(0)} = \mathbf{r}(\mathbf{u}^{(0)})$
  while $\|\mathbf{r}^{(k)}\|/\|\mathbf{r}^{(0)}\| > \varepsilon_{\text{rel}}$ **and** $\|\mathbf{r}^{(k)}\| > \varepsilon_{\text{abs}}$ **do**
    Solve linearized problem

$$\left.\frac{\partial \mathbf{r}\,(\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}^{(k)}} \Delta\mathbf{u}^{(k+1)} = -\mathbf{r}^{(k)}$$

    Initialize step size $\omega = 1$, and choose parameter $\tau < 1$
    Update solution $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \omega\Delta\mathbf{u}^{(k+1)}$
    Evaluate nonlinear residual $\mathbf{r}^{(k+1)} = \mathbf{r}(\mathbf{u}^{(k+1)})$
    while $\|\mathbf{r}^{(k+1)}\|/\|\mathbf{r}^{(k)}\| > 1 - \tau\omega$ **do**
      Reduce step size $\omega = \omega/2$
      Update solution $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \omega\Delta\mathbf{u}^{(k+1)}$
      Evaluate nonlinear residual $\mathbf{r}^{(k+1)} = \mathbf{r}(\mathbf{u}^{(k+1)})$
    end while
    $k \leftarrow k + 1$
  end while
  return $\mathbf{u}^{(k)}$
end function

where $\left.\frac{\partial \mathbf{r}(\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}_{\text{lin}}}$ is the Jacobian matrix. This system of equations is solved for the solution increment $\Delta\mathbf{u} = \mathbf{u} - \mathbf{u}_{\text{lin}}$, so that the new solution at which to evaluate the residual is given as $\mathbf{u} = \mathbf{u}_{\text{lin}} + \Delta\mathbf{u}$, defining an iterative solution procedure. Such a Newton algorithm is illustrated in Algorithm 2.1, which is an iterative procedure consisting mainly of three steps: evaluation of the nonlinear residual, solution of the linearized problem, and update of the solution vector. An accurate initial guess $\mathbf{u}^{(0)}$ is for example obtained by using an extrapolation of the solution from previous instants of time. Convergence of the Newton iteration is checked by a relative and absolute tolerance, and the solution is considered as converged whenever one of these tolerances is reached. The Newton method is known to converge quadratically for initial solutions sufficiently close to the solution, but convergence is not guaranteed for general nonlinear problems and general initial solutions, and globalization techniques can be used to improve convergence. The Newton iteration shown in Algorithm 2.1 also contains a globalization in form of a simple line-search or damping technique which adjusts the step length (the search direction remains unchanged) in order to achieve progress in convergence. This technique can also be categorized as a backtracking method, as opposed to trust region methods.

2.5.7.2 Coupled solution approach

For simplicity, this section only considers the case where the divergence and continuity penalty terms are applied in a postprocessing step as shown in equation (2.191). Translating equa-

tions (2.189) and (2.190) into matrix-vector notation yields

$$\begin{bmatrix} \mathbf{r}_\mathsf{m} \\ \mathbf{r}_\mathsf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{M}\left(\gamma_0^n \hat{\mathbf{u}} - \sum_{i=0}^{J-1} \alpha_i^n \mathbf{u}^{n-i}\right)/\Delta t_n + \mathbf{c}\left(\hat{\mathbf{u}}\right) + \mathbf{v}\left(\hat{\mathbf{u}}\right) + \mathbf{g}\left(\mathbf{p}^{n+1}\right) - \mathbf{b}(t_{n+1}) \\ -\mathbf{d}\left(\hat{\mathbf{u}}\right) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} . \tag{2.212}$$

The linearized system of equations required by the Newton solver is given as

$$\begin{bmatrix} \frac{\gamma_0^n}{\Delta t_n}\mathbf{M} + \mathbf{C}_\mathsf{lin}\left(\mathbf{u}^{(k)}\right) + \mathbf{V} & \mathbf{G} \\ -\mathbf{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}^{(k+1)} \\ \Delta\mathbf{p}^{(k+1)} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_\mathsf{m}\left(\mathbf{u}^{(k)}, \mathbf{p}^{(k)}\right) \\ \mathbf{r}_\mathsf{c}\left(\mathbf{u}^{(k)}\right) \end{bmatrix} . \tag{2.213}$$

This linear system of equations is symmetric with respect to the pressure gradient term and the velocity divergence term under the assumptions discussed in Section 2.4.2.2, $-\mathbf{D} = \mathbf{G}^\mathsf{T}$.

An alternative is to formulate the convective term explicitly, equation (2.192), resulting in the linear system of equations

$$\begin{bmatrix} \frac{\gamma_0^n}{\Delta t_n}\mathbf{M} + \mathbf{V} & \mathbf{G} \\ -\mathbf{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}} \\ \mathbf{p}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\mathsf{m} \\ \mathbf{b}_\mathsf{c} \end{bmatrix} , \tag{2.214}$$

with right-hand side vectors

$$\mathbf{b}_\mathsf{m} = \mathbf{M} \sum_{i=0}^{J-1} \frac{\alpha_i^n}{\Delta t_n} \mathbf{u}^{n-i} - \sum_{i=0}^{J-1} \beta_i^n \mathbf{c}\left(\mathbf{u}^{n-i}\right) - \mathbf{v}_\mathsf{inhom} - \mathbf{g}_\mathsf{inhom} + \mathbf{b}(t_{n+1}) , \tag{2.215}$$

$$\mathbf{b}_\mathsf{c} = \mathbf{d}_\mathsf{inhom} . \tag{2.216}$$

The penalty terms are applied in a postprocessing step, equation (2.191), which reads in matrix-vector notation

$$\left(\mathbf{M} + \Delta t_n \mathbf{A}_\mathsf{D} + \Delta t_n \mathbf{A}_\mathsf{C}\right) \mathbf{u}^{n+1} = \mathbf{M}\hat{\mathbf{u}} - \Delta t_n \mathbf{a}_{\mathsf{C,inhom}} . \tag{2.217}$$

When considering the Euler or Stokes equations as special cases, the viscous and convective terms are simply dropped from the above equations. Equation (2.217) becomes superfluous if a formulation without penalty terms is considered, $\mathbf{u}^{n+1} = \hat{\mathbf{u}}$ in that case. It is straightforward to derive similar matrix-vector formulations in case that the penalty terms are added to the monolithic system of equations, equations (2.187) and (2.188), which is therefore not shown explicitly here.

2.5.7.3 Dual splitting scheme

This section summarizes the dual splitting scheme, equations (2.193), (2.194), (2.197), (2.198), and (2.199), in matrix-vector notation

$$\mathbf{M}\frac{\gamma_0^n\hat{\mathbf{u}}-\sum_{i=0}^{J-1}\alpha_i^n\mathbf{u}^{n-i}}{\Delta t_n}=-\sum_{i=0}^{J-1}\beta_i^n\mathbf{c}\left(\mathbf{u}^{n-i}\right)+\mathbf{b}(t_{n+1})\ , \tag{2.218}$$

$$\mathbf{L}\mathbf{p}^{n+1}=-\frac{\gamma_0^n}{\Delta t_n}\mathbf{d}\left(\hat{\mathbf{u}}\right)-\mathbf{l}_{\mathsf{inhom}}\ , \tag{2.219}$$

$$\mathbf{M}\hat{\hat{\mathbf{u}}}=\mathbf{M}\hat{\mathbf{u}}-\frac{\Delta t_n}{\gamma_0^n}\mathbf{g}\left(\mathbf{p}^{n+1}\right)\ , \tag{2.220}$$

$$\left(\frac{\gamma_0^n}{\Delta t_n}\mathbf{M}+\mathbf{V}\right)\hat{\hat{\mathbf{u}}}=\frac{\gamma_0^n}{\Delta t_n}\mathbf{M}\hat{\mathbf{u}}-\mathbf{v}_{\mathsf{inhom}}\ , \tag{2.221}$$

$$\left(\mathbf{M}+\Delta t_n\mathbf{A}_{\mathsf{D}}+\Delta t_n\mathbf{A}_{\mathsf{C}}\right)\mathbf{u}^{n+1}=\mathbf{M}\hat{\hat{\mathbf{u}}}-\Delta t_n\mathbf{a}_{\mathsf{C,inhom}}\ . \tag{2.222}$$

The systems of equations for the different sub-steps of the splitting scheme are all linear and inhomogeneous contributions are shifted to the right-hand side of the equations. Due to the L^2-conforming nature of the shape functions, the mass matrix is block-diagonal, where block refers to the degrees of freedom within one element. Hence, independently of whether a matrix-based or matrix-free implementation is chosen, the mass matrix can be inverted in an element-wise fashion without the need to solve a global system of equations. The convective step, equation (2.218), and the projection step, equation (2.220), can therefore be considered as explicit steps.

2.5.7.4 Pressure-correction scheme

In case of an implicit treatment of the convective term, the nonlinear momentum equation (2.200) reads in matrix-vector notation

$$\mathbf{r}\left(\hat{\mathbf{u}}\right)=\mathbf{M}\frac{\gamma_0^n\hat{\mathbf{u}}-\sum_{i=0}^{J-1}\alpha_i^n\mathbf{u}^{n-i}}{\Delta t_n}+\mathbf{c}\left(\hat{\mathbf{u}}\right)+\mathbf{v}\left(\hat{\mathbf{u}}\right)+\sum_{i=0}^{J_p-1}\beta_i^n\mathbf{g}\left(\mathbf{p}^{n-i}\right)-\mathbf{b}^{n+1}=\mathbf{0}\ , \tag{2.223}$$

where the corresponding linearized problem is

$$\left(\frac{\gamma_0^n}{\Delta t_n}\mathbf{M}+\mathbf{C}_{\mathsf{lin}}\left(\mathbf{u}^{(k)}\right)+\mathbf{V}\right)\Delta\mathbf{u}^{(k+1)}=-\mathbf{r}\left(\mathbf{u}^{(k)}\right)\ . \tag{2.224}$$

In case of an explicit treatment of the convective term, equation (2.201), the momentum equation is linear in the velocity unknowns, so that constant vectors and inhomogeneous parts of discrete operators are shifted to the right-hand side of the equations. Together with all the remaining (linear) sub-steps, equations (2.202), (2.203), (2.204), and (2.205), the pressure-correction scheme

is given as follows in matrix-vector notation

$$\begin{aligned}\left(\frac{\gamma_0^n}{\Delta t_n}\mathbf{M}+\mathbf{V}\right)\hat{\mathbf{u}} = \mathbf{M}\sum_{i=0}^{J-1}\frac{\alpha_i^n}{\Delta t_n}\mathbf{u}^{n-i} - \mathbf{v}_{\mathsf{inhom}} \\ -\sum_{i=0}^{J-1}\beta_i^n\mathbf{c}\left(\mathbf{u}^{n-i}\right) - \sum_{i=0}^{J_p-1}\beta_i^n\mathbf{g}\left(\mathbf{p}^{n-i}\right) + \mathbf{b}(t_{n+1}) \ ,\end{aligned} \tag{2.225}$$

$$\mathbf{L}\boldsymbol{\phi}^{n+1} = -\frac{\gamma_0^n}{\Delta t_n}\mathbf{d}\left(\hat{\mathbf{u}}\right) - \mathbf{l}_{\mathsf{inhom}} \ , \tag{2.226}$$

$$\mathbf{M}_p\mathbf{p}^{n+1} = \mathbf{M}_p\left(\boldsymbol{\phi}^{n+1} + \sum_{i=0}^{J_p-1}\beta_i^n\mathbf{p}^{n-i}\right) - \chi\nu\mathbf{d}\left(\hat{\mathbf{u}}\right) \ , \tag{2.227}$$

$$\mathbf{M}\hat{\mathbf{u}} = \mathbf{M}\hat{\mathbf{u}} - \frac{\Delta t_n}{\gamma_0^n}\mathbf{g}\left(\boldsymbol{\phi}^{n+1}\right) \ , \tag{2.228}$$

$$\left(\mathbf{M}+\Delta t_n\mathbf{A}_{\mathsf{D}}+\Delta t_n\mathbf{A}_{\mathsf{C}}\right)\mathbf{u}^{n+1} = \mathbf{M}\hat{\mathbf{u}} - \Delta t_n\mathbf{a}_{\mathsf{C,inhom}} \ . \tag{2.229}$$

The pressure update step with pressure mass matrix $\mathbf{M}_p$, equation (2.227), and the projection step, equation (2.228), are again explicit steps in the sense that these steps do not involve the solution of a linear system of equations for the global vector of degrees of freedom. As noted in Section 2.5.3, equation (2.228) can be inserted into equation (2.229) to obtain a single equation.

2.5.7.5 A note on the solvability of linear system of equations in case of pure Dirichlet boundary conditions

As mentioned in Section 2.2, the pressure is only defined up to an additive constant in case of pure Dirichlet boundary conditions for the velocity, $\Gamma_h = \Gamma_h^{\mathrm{D}}$. In terms of the solution of linear system of equations, equation (2.213) or equation (2.214) for the coupled solution approach, equation (2.219) for the dual splitting scheme, and equation (2.226) for the pressure-correction scheme are singular. The pressure solution $\mathbf{p}$ representing the state of constant pressure forms the one dimensional null space of the corresponding matrix. In case of nodal shape functions as considered in this work, the state of constant pressure corresponds to the 1-vector $\mathbf{1} = (1, ..., 1)^{\mathsf{T}}$. In case of equation (2.213) this means

$$\begin{bmatrix}\frac{\gamma_0}{\Delta t}\mathbf{M} + \mathbf{C}_{\mathsf{lin}}\left(\mathbf{u}^{(k)}\right) + \mathbf{V} & \mathbf{G} \\ -\mathbf{D} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{0} \\ \mathbf{1}\end{bmatrix} = \mathbf{0} \ , \tag{2.230}$$

and equivalently for equation (2.214). In case of projection-type solvers, equation (2.219) and equation (2.226), the pressure Poisson matrix satisfies

$$\mathbf{L}\,\mathbf{1} = \mathbf{0} \ . \tag{2.231}$$

Despite the present singularity, the linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ is still solvable using a Krylov subspace method if the system of equations is consistent in the sense that

$$\mathbf{n}^{\mathsf{T}}\mathbf{A}\mathbf{x} = \mathbf{n}^{\mathsf{T}}\mathbf{b} = 0 \ , \tag{2.232}$$

where $\mathbf{n}$ denotes the vector that spans the null space of $\mathbf{A}$, i.e., $\mathbf{n} = [\mathbf{0}^\mathsf{T}, \mathbf{1}^\mathsf{T}]^\mathsf{T}$ for the coupled problem and $\mathbf{n} = \mathbf{1}$ for the projection methods. If the solvability condition $\mathbf{n}^\mathsf{T}\mathbf{b} = 0$ is not fulfilled, i.e., the arithmetic mean of the right-hand side vector is unequal zero, a transformed linear system of equations

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{2.233}$$

is solved by applying a Krylov projection

$$\tilde{\mathbf{A}} = \mathbf{P}\mathbf{A}\mathbf{P} \ , \ \mathbf{x} = \mathbf{P}\tilde{\mathbf{x}} \ , \ \tilde{\mathbf{b}} = \mathbf{P}\mathbf{b} \ , \tag{2.234}$$

where the projector $\mathbf{P}$ (fulfilling $\mathbf{P} = \mathbf{P}^\mathsf{T}$ and $\mathbf{P}^2 = \mathbf{P}$) is defined as

$$\mathbf{P} = \mathbf{I} - \frac{\mathbf{n}\,\mathbf{n}^\mathsf{T}}{\mathbf{n}^\mathsf{T}\mathbf{n}} \ , \tag{2.235}$$

and where $\mathbf{I}$ denotes the identity matrix. The projector shifts all entries of a vector by a constant value so that its mean value becomes zero. It follows immediately that $\mathbf{n}^\mathsf{T}\tilde{\mathbf{b}} = 0$, so that the transformed system of equations fulfills the solvability condition (2.232).

The condition $\mathbf{n}^\mathsf{T}\mathbf{b} = 0$ can now be investigated for the singular problems mentioned above. Beginning with equation (2.213), one obtains

$$[\mathbf{0}^\mathsf{T}, \mathbf{1}^\mathsf{T}]\ \mathbf{b} = -[\mathbf{0}^\mathsf{T}, \mathbf{1}^\mathsf{T}] \begin{bmatrix} \mathbf{r}_\mathsf{m}\left(\mathbf{u}^{(k)}, \mathbf{p}^{(k)}\right) \\ \mathbf{r}_\mathsf{c}\left(\mathbf{u}^{(k)}\right) \end{bmatrix} = \mathbf{1}^\mathsf{T}\mathbf{d}\left(\mathbf{u}\right) \ . \tag{2.236}$$

Applying equation (2.99) with $\Gamma_h = \Gamma_h^\mathrm{D}$ and $\Gamma_h^\mathrm{N} = \emptyset$ yields

$$\begin{aligned} \mathbf{1}^\mathsf{T}\mathbf{d}\left(\mathbf{u}\right) &= \sum_{e=1}^{N_\mathrm{el}} \Big(-\left(\nabla 1, \boldsymbol{u}_h\right)_{\Omega_e} + \left(1, \{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right)_{\partial\Omega_e \setminus \Gamma_h} + \left(1, \boldsymbol{g}_u \cdot \boldsymbol{n}\right)_{\partial\Omega_e \cap \Gamma_h^\mathrm{D}} \Big) \\ &= \int_{\Gamma_h^\mathrm{D}} \boldsymbol{g}_u \cdot \boldsymbol{n} \ \mathrm{d}\Gamma \overset{(2.9)}{=} 0 \ . \end{aligned} \tag{2.237}$$

Hence, if the specified Dirichlet boundary condition $\boldsymbol{g}_u$ is consistent according to equation (2.9), the resulting linear system of equations is also consistent. Similarly, one obtains for the linear Stokes problem (2.214)

$$[\mathbf{0}^\mathsf{T}, \mathbf{1}^\mathsf{T}]\ \mathbf{b} = \mathbf{1}^\mathsf{T}\mathbf{d}_\mathsf{inhom} \ , \tag{2.238}$$

and using the inhomogeneous part of the divergence operator as specified in equation (2.99) yields

$$[\mathbf{0}^\mathsf{T}, \mathbf{1}^\mathsf{T}]\ \mathbf{b} = \sum_{e=1}^{N_\mathrm{el}} \left(1, \boldsymbol{g}_u \cdot \boldsymbol{n}\right)_{\partial\Omega_e \cap \Gamma_h^\mathrm{D}} = \int_{\Gamma_h^\mathrm{D}} \boldsymbol{g}_u \cdot \boldsymbol{n} \ \mathrm{d}\Gamma \overset{(2.9)}{=} 0 \ . \tag{2.239}$$

For the pressure Poisson equation (2.226) of the pressure-correction scheme, a similar result is obtained

$$\mathbf{1}^\mathsf{T}\mathbf{b} = -\frac{\gamma_0^n}{\Delta t_n}\mathbf{1}^\mathsf{T}\mathbf{d}\left(\hat{\mathbf{u}}\right) - \mathbf{1}^\mathsf{T}\mathbf{l}_\mathsf{inhom} = -\frac{\gamma_0^n}{\Delta t_n}\mathbf{1}^\mathsf{T}\mathbf{d}\left(\hat{\mathbf{u}}\right) \overset{(2.237)}{=} 0 \ , \tag{2.240}$$

where the result (2.237) has been used and the fact that the inhomgeneous part $\mathsf{I}_{\mathsf{inhom}}$ vanishes in case of pure Dirichlet boundary conditions according to equations (2.128) and (2.58). Finally, the pressure Poisson equation (2.219) of the dual splitting scheme is considered

$$\mathbf{1}^{\mathsf{T}}\mathbf{b} = -\frac{\gamma_0}{\Delta t}\mathbf{1}^{\mathsf{T}}\mathbf{d}\left(\hat{\mathbf{u}}\right) - \mathbf{1}^{\mathsf{T}}\mathsf{I}_{\mathsf{inhom}} \ , \tag{2.241}$$

where both terms on the right-hand side are in general unequal zero. On the one hand, according to equation (2.194), $\mathsf{I}_{\mathsf{inhom}}$ depends on the pressure Neumann boundary conditions (2.40) that itself depends on the numerical solution $\boldsymbol{u}_h$. On the other hand, the intermediate velocity field does not fulfill the boundary condition $\boldsymbol{g}_u$ but the consistent boundary condition $\boldsymbol{g}_{\hat{u}}$, equation (2.42), for which equation (2.9) does not hold in general

$$\begin{aligned}\mathbf{1}^{\mathsf{T}}\mathbf{d}\left(\hat{\mathbf{u}}\right) &= \sum_{e=1}^{N_{\mathrm{el}}}\left(-\left(\nabla 1, \hat{\boldsymbol{u}}_h\right)_{\Omega_e} + \left(1, \{\{\hat{\boldsymbol{u}}_h\}\}\cdot\boldsymbol{n}\right)_{\partial\Omega_e\setminus\Gamma_h} + \left(1, \boldsymbol{g}_{\hat{u}}\cdot\boldsymbol{n}\right)_{\partial\Omega_e\cap\Gamma_h^{\mathrm{D}}}\right) \\ &= \int_{\Gamma_h^{\mathrm{D}}} \boldsymbol{g}_{\hat{u}}\cdot\boldsymbol{n}\,\mathrm{d}\Gamma \neq 0 \ .\end{aligned} \tag{2.242}$$

Remark 2.13 *The theoretical considerations above are in agreement with numerical experiments highlighting the necessity to project the right-hand side vector of the pressure Poisson equation in case of the dual splitting scheme according to equation* (2.234) *in order to obtain a consistent linear system of equations. However, note that the projection of the system matrix according to equation* (2.234) *can be skipped since the system matrix inherently contains this projection due to its nullspace. This is beneficial in terms of computational costs, and in particular in a parallel setting and the strong-scaling limit, since the inner product* $\mathbf{n}^{\mathsf{T}}\mathbf{p}$ *involving global communication between all processors (which would otherwise be applied in every matrix-vector product within the iterative linear solver and its preconditioner) can be avoided. The above derivations assume exact integration, so that a projection of the right-hand side might in general be necessary also for the other solution approaches. However, this has to be done only once per solution of the linear system of equations and is, therefore, negligible in terms of computational costs.*

To allow a computation of pressure errors for test cases with analytical solution, the numerical pressure solution can be shifted by a constant value, e.g., so that its mean value is consistent with the mean value of the analytical solution.

2.5.8 Analysis of eigenvalue spectrum for unsteady Stokes equations

To investigate the stability of the discretization scheme for example as a function of the time step size, it is illustrative to study the so-called propagation matrix and its eigenvalue spectrum. This section briefly presents the theoretical background for the numerical results shown in Section 2.6.2. A similar eigenvalue analysis is performed in Leriche and Labrosse (2000), Leriche et al. (2006), where the stability of the dual splitting scheme is analyzed for different orders

of the time integration and extrapolation schemes using a Chebyshev collocation method and a Legendre spectral element method for discretization in space.

By assuming $J = 1$ (BDF1 time integration scheme), $\boldsymbol{f} = \boldsymbol{0}$, homogeneous boundary conditions, and by neglecting the convective term (Stokes equations), the velocity solution $\mathbf{u}^{n+1}$ at time t_{n+1} can be written in terms of the velocity solution $\mathbf{u}^n$ at time t_n and a velocity propagation matrix G

$$\mathbf{u}^{n+1} = \mathsf{G}\mathbf{u}^n \,, \tag{2.243}$$

where $\mathbf{u} = (u_1, ..., u_{N_{\mathrm{DoFs},u}})^\mathsf{T} \in \mathbb{R}^{N_{\mathrm{DoFs},u}}$ is a vector containing the nodal degrees of freedom of the velocity. By the example of the dual splitting scheme, this equation can be derived from equations (2.193), (2.194), (2.197), (2.198), and (2.199). For this analysis, the pressure is thought of as an auxiliary variable and is eliminated by expressing the pressure as a function of the velocity. The stability condition for this problem is given as follows

$$\max_i |\lambda_i(\mathsf{G})| \leq 1 \,. \tag{2.244}$$

Under the above condition, modes can not be amplified, the solution remains bounded, and the discrete scheme is stable.

Remark 2.14 *To compute the eigenvalue spectrum within a matrix-free implementation framework, the matrix* G *is assembled columnwise. To obtain column* j *of matrix* G*, one time step is solved with input vector* $u_i^n = \delta_{ij}, i = 1, ..., N_{\mathrm{DoFs},u}$*, using the BDF1 scheme,* $\boldsymbol{f} = \boldsymbol{0}$*, neglecting the convective term, and homogeneous boundary conditions.*

2.6 A posteriori quantification of robustness and accuracy

This section provides an in-depth numerical investigation and verification of the proposed incompressible Navier–Stokes solvers. Many of the flow problems studied here are academic in nature. Widely used benchmark problems are studied which are characterized by a simple setup in terms of geometry and boundary conditions and for which analytical or accurate reference solutions are available. Despite their simple geometry, they are nevertheless challenging in terms of the robustness of discretization schemes, in particular when studying high-Reynolds-number turbulent flows in under-resolved scenarios. The objectives are twofold: (i) Robustness tests are performed for a series of examples for which state-of-the-art discretization methods from the literature lacked robustness and for which excellent stability can be demonstrated for the proposed incompressible DG solvers. (ii) Parameter studies in the form of convergence studies in space and time are performed, on the one hand to investigate the accuracy of high-order discretizations, and on the other hand to compare the accuracy of the present approach to state-of-the-art methods from the literature.

The following topics are addressed in this section. Section 2.6.1 defines default parameters. By the example of an unsteady Stokes problem, Section 2.6.2 investigates the stability of the discretization scheme for small time step sizes and Section 2.6.3 addresses the aspect of inf–sup stability. Different formulations of the velocity–pressure coupling terms are investigated as well

as both equal-order and mixed-order polynomials for the velocity and pressure. In order to investigate the influence of different terms of the discretization schemes separately, no divergence and continuity penalty terms are applied in these sections. Section 2.6.4 studies the aspect of pressure-robustness for a steady Stokes problem, where formulations with and without divergence and continuity penalty terms are considered. The remaining examples consider the full Navier–Stokes equations using formulations that ensure stability for small time step sizes and inf–sup stability, and make use of the divergence and continuity penalty terms. Section 2.6.5 investigates the temporal and spatial convergence behavior for an analytical solution of the transient incompressible Navier–Stokes equations with non-trivial and time-dependent Dirichlet and Neumann boundary conditions. Section 2.6.6 analyzes the accuracy of the present high-order DG solvers for the benchmark problem of laminar flow around a cylinder. Section 2.6.7 addresses the topics of robustness and accuracy for transitional and turbulent flows by the examples of the Orr–Sommerfeld stability problem, the three-dimensional Taylor–Green vortex problem, and turbulent channel flow. An emphasis is put on the under-resolved regime and the importance of the additional stabilization terms is highlighted. Moreover, the accuracy of high-order discretizations is assessed for coarse spatial resolutions operating in the pre-asymptotic regime of convergence. Here, accuracy is analyzed as a function of the number of unknowns for a wide range of polynomial degrees, while an efficiency analysis in terms of error-vs-costs and time-to-solution is postponed to Chapter 6. Finally, a perspective on applications is given by considering the turbulent flow over a backward facing step, where a precursor simulation strategy is used in order to obtain physically consistent (turbulent) inflow boundary conditions and where numerical results are validated against experimental data.

2.6.1 Default parameters

Unless specified otherwise, the weak formulation of the velocity–pressure coupling terms is used, the conservative formulation of the convective term (with a default value of $\zeta_{\mathrm{LF}} \approx \frac{1}{2}$ for the Lax–Friedrichs flux), and the Laplace formulation of the viscous term. The divergence and continuity penalty terms are generally used, and are applied in a postprocessing step also for the coupled solution approach by default. Penalty factors of the interior penalty discretization and the divergence and continuity penalty terms are set to 1 by default. Mixed order polynomials of degree $(k_u, k_p) = (k, k-1)$ are used by default. Regarding the extrapolation order J_p for projection methods, the default setup is $J_p = \min(2, J), J \leq 3$ for the dual splitting scheme, and $J_p = \min(2, J) - 1, J \leq 2$ for the pressure-correction scheme, where the so-called rotational formulation is chosen for the latter scheme unless specified otherwise. Regarding the boundary conditions (2.40) and (2.42) for the dual splitting scheme, the convective formulation of the convective term is used by default. Regarding the time derivative in the pressure Neumann boundary condition, either the analytical time derivative as in equation (2.40) or the discrete BDF time derivative as in equation (2.41) can be used, which will be specified for the test cases shown in the following. Further, absolute and relative solver tolerances are specified as $\varepsilon_{\mathrm{abs}}$ and $\varepsilon_{\mathrm{rel}}$, respectively.

To perform convergence tests for problems with analytical solution, relative L^2-errors are used that are defined as

$$e_u = \frac{\|\boldsymbol{u}(\boldsymbol{x},t=T)-\boldsymbol{u}_h(\boldsymbol{x},t=T)\|_{L^2(\Omega_h)}}{\|\boldsymbol{u}(\boldsymbol{x},t=T)\|_{L^2(\Omega_h)}} \ , \ e_p = \frac{\|p(\boldsymbol{x},t=T)-p_h(\boldsymbol{x},t=T)\|_{L^2(\Omega_h)}}{\|p(\boldsymbol{x},t=T)\|_{L^2(\Omega_h)}} \ , \tag{2.245}$$

where Gaussian quadrature is used to calculate the volume integrals in the above expressions. The number of one-dimensional quadrature points is k_u+3 for the velocity error and k_p+3 for the pressure error in order to ensure that the calculation of errors is not affected by quadrature errors. Experimental rates of convergence for two meshes with characteristic element lengths h_1 and h_2 are calculated as $\log(e_{h_1}/e_{h_2})/\log(h_1/h_2)$. Other error measures are defined where necessary.

2.6.2 Stability in the limit of small time step sizes

This section investigates the stability of the proposed discretization schemes for small time step sizes, where instabilities have been reported previously in the literature, see for example Ferrer and Willden (2011), Ferrer et al. (2014) and the discussion in Sections 2.1.1 and 2.1.2.

2.6.2.1 Motivation

Although the instabilities discussed here are denoted as instabilities in the limit of small time step sizes, this does not imply that the relevance of these instabilities is of academic nature. To avoid the instabilities discussed in Ferrer and Willden (2011), Ferrer et al. (2014), the time step size has to be larger than a critical time step size, $\Delta t \geq \Delta t_{\text{crit}}$ with $\nu \Delta t_{\text{crit}} \sim \frac{h^2}{k^3}$. However, when using an explicit treatment of the convective term, the time step size is also restricted according to the CFL condition, $\Delta t \leq \Delta t_{\text{CFL}}$ with $\Delta t_{\text{CFL}} = \frac{\text{Cr}_{\text{crit}}}{U_{\max}} \frac{h}{k^{1.5}}$. Stability can be obtained only if

$$\frac{C}{\nu}\frac{h^2}{k^3} = \Delta t_{\text{crit}} \leq \Delta t \leq \Delta t_{\text{CFL}} = \frac{\text{Cr}_{\text{crit}}}{k^{1.5}} \frac{h}{U_{\max}} \ . \tag{2.246}$$

Accordingly, as discussed in detail in Ferrer et al. (2014), the characteristic element length h and polynomial degree k have to fulfill a condition such as $\frac{k^{1.5}}{h} > C_{h/k} \frac{U_{\max}}{\nu}$ in order to avoid conflicts of both time step restrictions, i.e., a high spatial resolution is required to ensure stability especially for large Reynolds numbers. In other words, the method might be unstable for all time step sizes in case of coarse spatial resolutions and high-Reynolds-number flows. The above relations point to the high practical relevance of these instabilities, i.e., resolving this stability problem is a necessary prerequisite to obtain a robust flow solver for complex engineering applications.

2.6.2.2 Problem setup

Since these instabilities already occur for the Stokes equations, a simple unsteady Stokes problem with analytical solution is studied here in favor of a detailed and careful investigation of the relevant effects. Additional results (not reproduced here for reasons of brevity) highlighting the practical relevance of resolving the problem of instabilities for small time step sizes are shown in Fehn et al. (2017) by the more practical example of laminar Navier–Stokes flow around a

cylinder. As a numerical example, an unsteady Stokes flow problem is chosen that has already been analyzed in Ferrer and Willden (2011), Ferrer et al. (2014) in the context of instabilities occurring for small time step sizes. This analytical solution of the two-dimensional unsteady Stokes equations with $\boldsymbol{f} = \boldsymbol{0}$ is defined as

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{x},t) &= \begin{pmatrix} \sin(x_1)\left(a\sin(ax_2) - \cos(a)\sinh(x_2)\right) \\ \cos(x_1)\left(\cos(ax_2) + \cos(a)\cosh(x_2)\right) \end{pmatrix} \exp\left(-\lambda t\right) \ , \\ p(\boldsymbol{x},t) &= \lambda\cos(a)\cos(x_1)\sinh(x_2)\exp\left(-\lambda t\right) \ , \end{aligned} \tag{2.247}$$

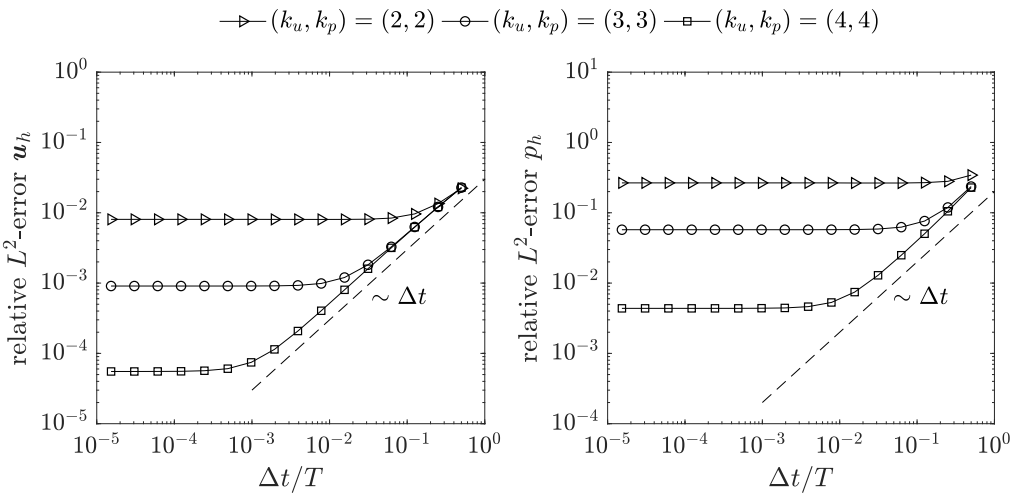
where the parameters λ, ν, a are given as $\lambda = \nu(1+a^2)$ with $\nu = 1$ and $a = 2.883356$. The domain $\Omega = [-L/2, L/2]^2$ is a square of length $L = 2$ and the time interval is $[0, T] = [0, 0.1]$. Dirichlet boundary conditions are prescribed on the whole boundary, $\Gamma = \Gamma^{\mathrm{D}}$. The Dirichlet boundary condition $\boldsymbol{g}_u$, the time derivative term $\partial \boldsymbol{g}_u / \partial t$ in equation (2.40), and initial conditions are deduced from the analytical solution. The solution at previous instants of time $t_{-J+1}, ..., t_{-1}$ required by the BDF scheme for $J > 1$ is obtained by interpolation of the analytical solution. Note that $\nabla p \cdot \boldsymbol{n} \neq 0$ on domain boundaries. Hence, this flow problem is for example also a suitable test case to assess the temporal convergence properties of the projection-type solution methods with the important topic of the imposition of pressure Neumann boundary conditions. A uniform Cartesian grid consisting of quadrilateral elements of length $h = L/2^l$ in x_1 and x_2-direction is used, where l denotes the level of refinement. To fix the pressure level, the mean value of the vector containing the pressure degrees of freedom is set to zero. This is consistent with the exact pressure solution due to the symmetry of the analytical solution and the uniformity of the mesh. Solver tolerances are $\varepsilon_{\mathrm{abs}} = 10^{-12}$ and $\varepsilon_{\mathrm{rel}} = 10^{-8}$.

2.6.2.3 Instabilities for small time step sizes

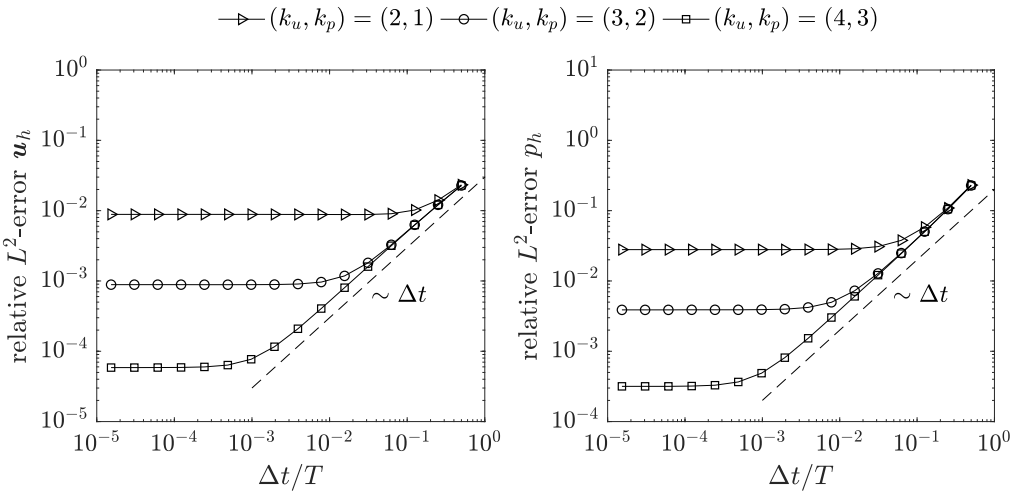
To investigate the stability of the different solution approaches for small time step sizes, temporal convergence tests are performed and the time step size $\Delta t / T$ is varied over a wide range. Since the instabilities reported in Ferrer and Willden (2011), Ferrer et al. (2014) occur primarily for coarse spatial resolutions, a coarse mesh with refine level $l = 2$ is selected. Moreover, the results are compared for both equal-order polynomials and mixed-order polynomials and varying polynomial degree. To show the impact of the temporal discretization error, this analysis is performed for first-order time integration schemes, $J = 1$. Note, however, that qualitatively similar results in terms of stability are obtained when using second-order accurate time integration schemes.

The results for the coupled solution approach presented in Figure 2.4 show the expected behavior. The error is proportional to Δt for large time steps. For small time steps, the temporal error becomes negligible as compared to the spatial discretization error and the overall error approaches a constant value. The pressure error is significantly larger for equal-order polynomials than for mixed-order polynomials while the velocity error is almost the same for both equal-order and mixed-order polynomials. This apsect is analyzed in more detail in Section 2.6.3.

Figure 2.5 shows results obtained for the dual splitting scheme. As a reference method, the DG discretization proposed in Hesthaven and Warburton (2007) without integration-by-parts of the velocity divergence term and pressure gradient term is considered, see equations (2.100) and (2.105). As in Ferrer and Willden (2011), Ferrer et al. (2014), instabilities are observed for small time step sizes and these instabilities occur similarly for equal-order and mixed-order



(a) equal-order polynomials

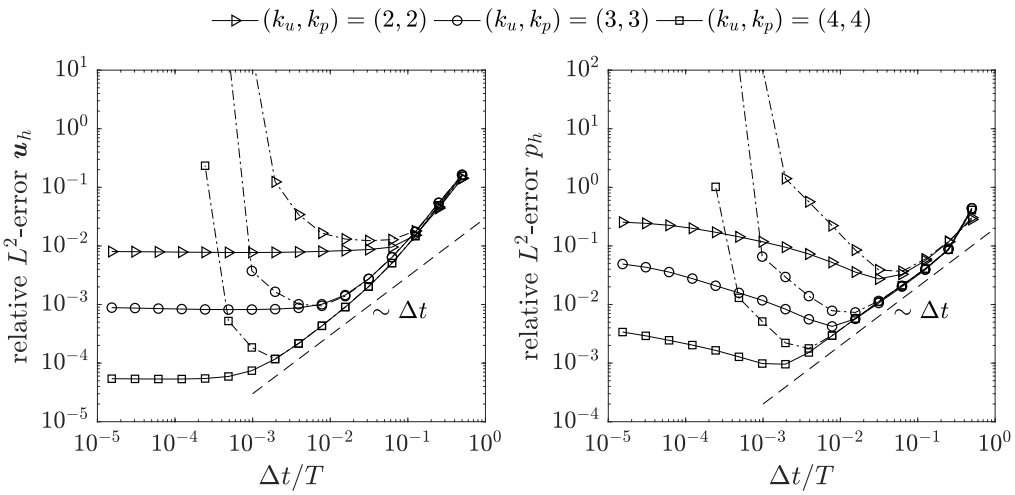


(b) mixed-order polynomials

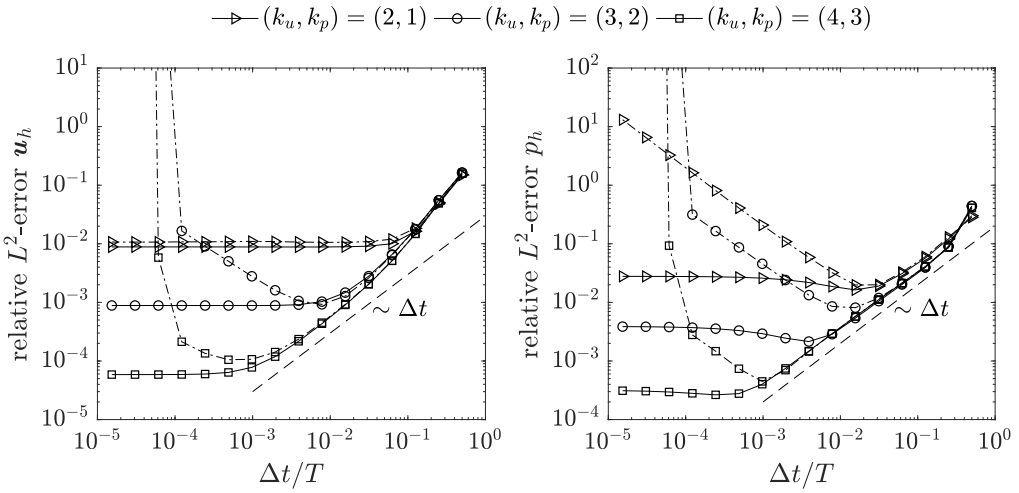
Figure 2.4: Stability analysis of coupled solution approach (BDF1) for small time step sizes.

polynomials. The situation changes, however, when applying the DG formulations d_h^e and g_h^e according to equations (2.97) and (2.102), repectively, along with consistent boundary conditions. For this formulation, stability is obtained for both equal-order and mixed-order polynomials and the behavior for small time steps is comparable to the coupled solution approach.

As a further verification of the results, the same simulations are performed for the non-incremental pressure-correction scheme in standard formulation, see Figure 2.6. Again, the present DG formulation of the velocity–pressure coupling terms in weak form is compared to the reference formulation. The stability behavior is the same as for the dual splitting scheme. Note that the reference slope representing the expected theoretical rate of convergence is $\Delta t^{1/2}$ for the pressure in case of the non-incremental pressure-correction scheme in standard form.



(a) equal-order polynomials

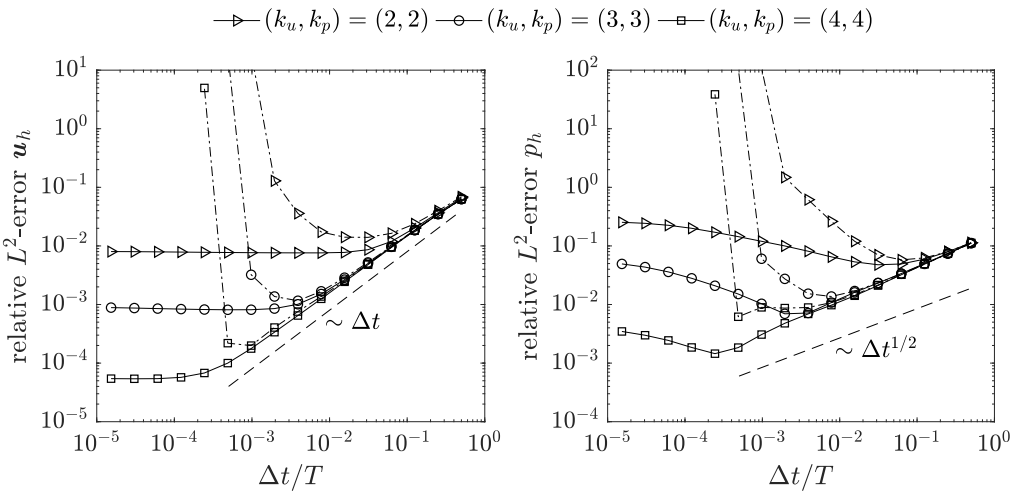


(b) mixed-order polynomials

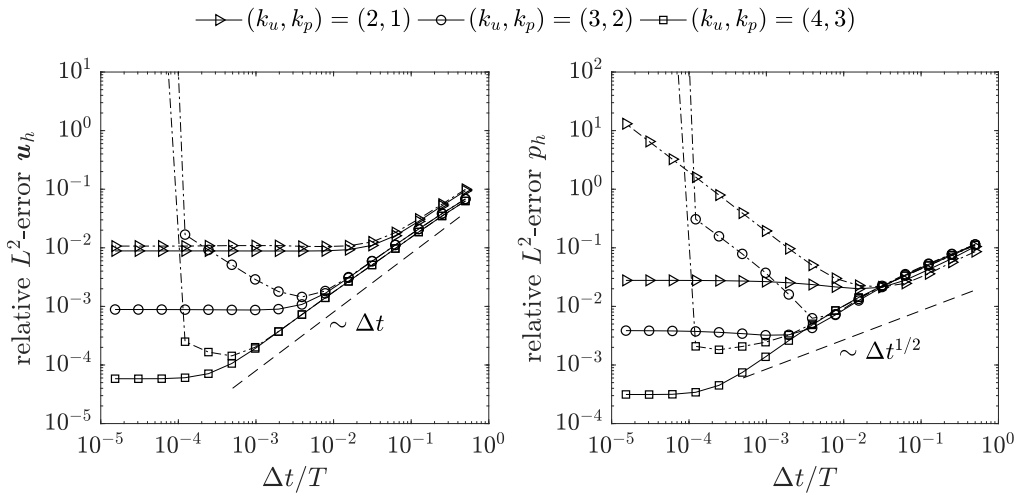
Figure 2.5: Stability analysis of dual splitting scheme (BDF1) for small time step sizes. Results obtained for the reference formulation of the velocity–pressure coupling terms without integration-by-parts are shown as dashed-dotted lines.

2.6.2.4 Eigenvalue analysis

As a further verification of the above results, the eigenvalue spectrum is investigated according to the theoretical considerations in Section 2.5.8. Eigenvalue spectra for two different time step sizes of $\Delta t/T = 10^{-1}$ and $\Delta t/T = 10^{-5}$ are visualized in Figure 2.7, where the spatial resolution is $l = 2$ with $(k_u, k_p) = (4, 4)$ in case of equal-order polynomials and $(k_u, k_p) = (4, 3)$ in case of mixed-order polynomials. For $\Delta t/T = 10^{-1}$, all eigenvalues are inside the stable regime for both the reference formulation and the present formulation. For $\Delta t/T = 10^{-5}$, some eigen-



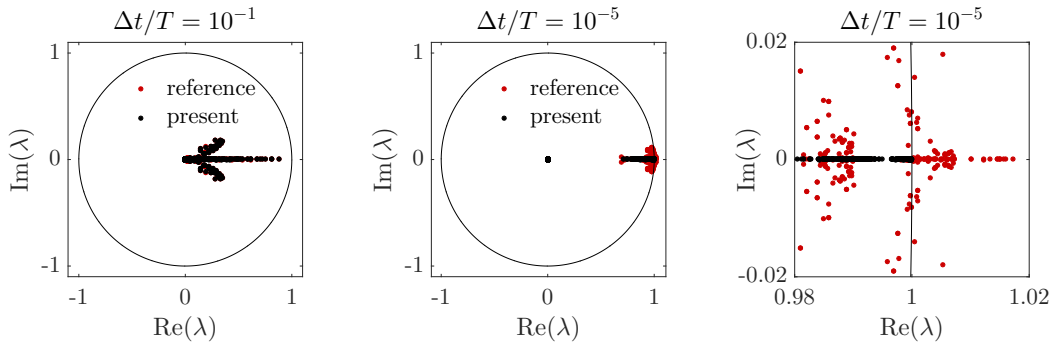
(a) equal-order polynomials



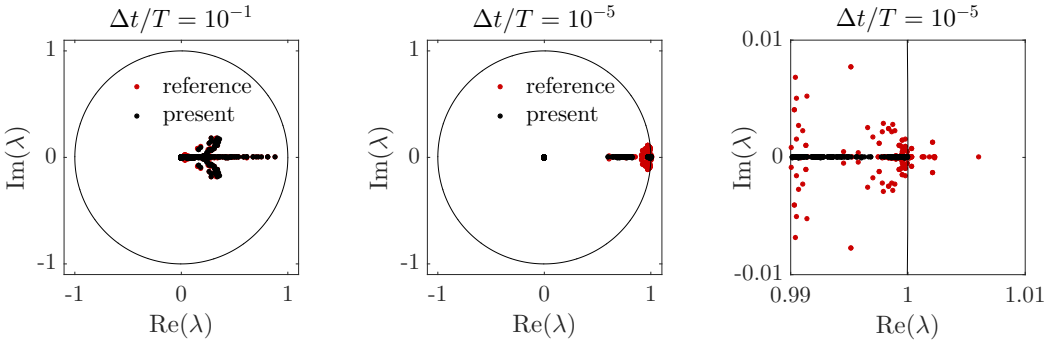
(b) mixed-order polynomials

Figure 2.6: Stability analysis of pressure-correction scheme (BDF1) for small time step sizes. Results obtained for the reference formulation of the velocity–pressure coupling terms without integration-by-parts are shown as dashed-dotted lines.

values are outside the stable regime in case of the reference formulation, while all eigenvalues are inside the stable regime in case of the present formulation. Note that the eigenvalue spectra are very similar for equal-order polynomials and mixed-order polynomials. Unstable eigenvalues clearly occur also for mixed-order polynomials when using the reference formulation of the velocity–pressure coupling terms. The reader is referred to Fehn et al. (2017) where additional results are shown that analyze the maximum eigenvalue as a function of the time step size in the limit $\Delta t \to 0$ for equal-order and mixed-order polynomials, as well as for the reference formulation and the present formulation. The eigenvalue analysis has also been performed for



(a) equal-order polynomials ($l = 2$, $(k_u, k_p) = (4, 4)$)



(b) mixed-order polynomials ($l = 2$, $(k_u, k_p) = (4, 3)$)

Figure 2.7: Eigenvalue analysis for dual splitting scheme (BDF1): eigenvalue spectra for different time step sizes.

the non-incremental pressure-correction scheme in standard formulation, yielding very similar results.

2.6.2.5 Conclusion

The above results lead to the following conclusions. The results shown here and first published in Fehn et al. (2017) are in contradiction to the conclusions drawn in Ferrer et al. (2014) where it was stated that the instabilities for small time steps are related to the temporal discretization and inf–sup instabilities but not to the spatial discretization. Instead, the results shown here suggest that these instabilities are clearly related to the discontinuous Galerkin formulation of the velocity divergence term and the pressure gradient term. Integration-by-parts of these terms along with central fluxes and consistent boundary conditions is necessary to ensure stability (and high-order accuracy) for small time step sizes. The discretization of these operators is a basic building block of any incompressible Navier–Stokes solver, independently of the temporal discretization. In fact, the increased pressure error observed for the equal-order formulation is an indication of inf–sup instabilities and is discussed in more detail below. As for the Stokes flow problem considered in this section, a qualitatively similar behavior in terms of instabilities for small time step sizes and the different formulations of the velocity divergence term and pressure gradient term is observed when considering the full Navier–Stokes equations, e.g., for the two-

dimensional Taylor–Green vortex problem considered in Section 2.6.5 or the three-dimensional Beltrami flow problem (Ethier and Steinman 1994). After the present results had been published in Fehn et al. (2017), an independent study by Xu et al. (2019) confirmed these results in the meantime.

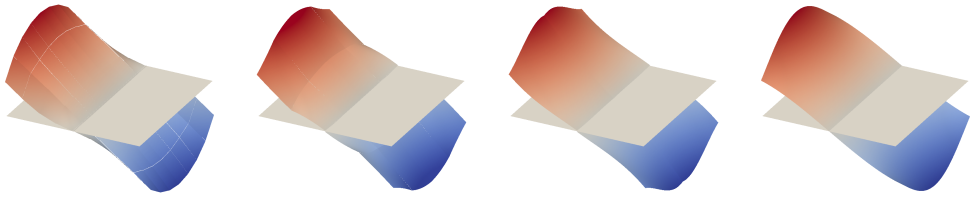
2.6.2.6 Temporal convergence test

While the above investigations concentrated on the lowest order time integration scheme with $J = 1$, additional results of a temporal convergence study for schemes of order $J = 1, 2$ (and both the standard and rotational forms in case of the pressure-correction scheme) are shown in Fehn et al. (2017) for the stable formulation that uses integration-by-parts for the velocity–pressure coupling terms. Optimal rates of convergence of order J are achieved for velocity and pressure for this Stokes flow problem for the coupled solution approach, the dual splitting scheme, and the pressure-correction scheme in rotational form.

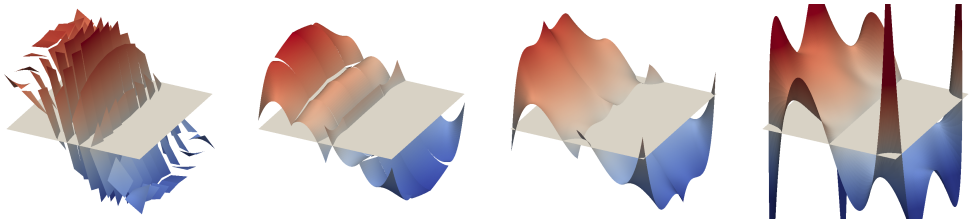
2.6.3 Inf–sup stability

While stability in the limit of small time step sizes has been obtained in the previous section for appropriate formulations of the velocity–pressure coupling terms, significantly larger pressure errors have been observed for equal-order polynomials than for mixed-order polynomials. This section investigates the aspect of inf–sup stability for different Navier–Stokes solution strategies in order to complement and verify the theoretical investigations from Section 2.5.4. The emphasis is put on the role that projection-type methods play in the context of inf–sup stability for equal-order vs. mixed-order polynomials for velocity and pressure. For example, equal-order polynomials have been used successfully for the dual splitting scheme, see for example Krank et al. (2017), and this section demonstrates that certain projection schemes indeed exhibit different characteristics in terms of inf–sup stability than a coupled solution of velocity and pressure unknowns. The Stokes flow problem investigated in this section is identical to the problem described previously in Section 2.6.2.

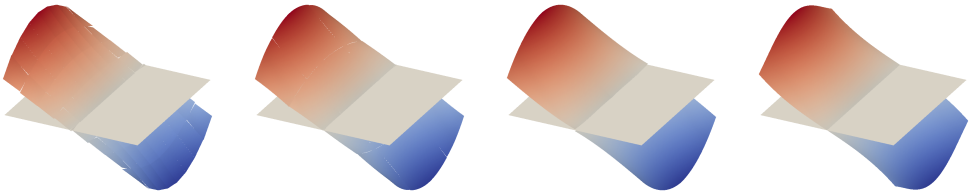
For ease of interpretation, graphical illustrations are shown here, and the reader is referred to additional results of spatial convergence tests shown in Fehn et al. (2017). Figure 2.8 displays the pressure solution for the dual splitting scheme at final time $t = T$ for equal-order and mixed-order polynomials of varying degree as well as for different time step sizes of $\Delta t/T = 10^{-1}$ and $\Delta t/T = 10^{-3}$. Since inf–sup instabilities are expected to be pronounced for coarse meshes, comparably low spatial resolutions are considered and the level of refinement is reduced simultaneously when increasing the polynomial degree. For equal-order polynomials, the time step size has a huge influence on the pressure solution and artificial pressure modes show up for small Δt. This is in agreement with equation (2.206), which predicts that the stabilizing effect is related to Δt and diminishes when decreasing the time step size. For the mixed-order formulation, the pressure solution is smooth and results for $\Delta t/T = 10^{-1}$ and $\Delta t/T = 10^{-3}$ are indistinguishable. Hence, results are only presented for the smaller time step size which is the critical one in this respect. The same simulations have been performed for the *non-incremental* pressure-correction scheme in standard formulation. For the equal-order formulation and the two time step sizes of $\Delta t/T = 10^{-1}$ and $\Delta t/T = 10^{-3}$, very similar results are obtained as for the dual-splitting scheme in terms of spurious pressure oscillations, which is in line with



(a) $\Delta t/T = 10^{-1}$, equal-order polynomials $(k_u, k_p) = (k, k)$, refine level $l = 4 - k$



(b) $\Delta t/T = 10^{-3}$, equal-order polynomials $(k_u, k_p) = (k, k)$, refine level $l = 4 - k$

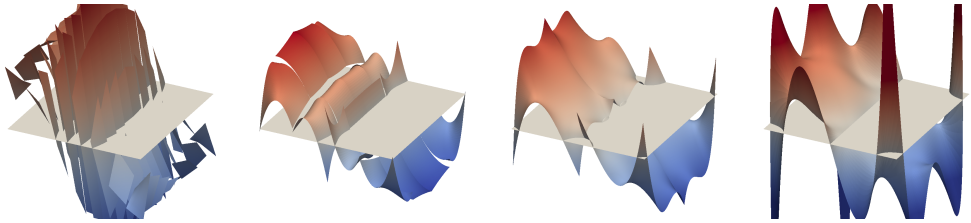


(c) $\Delta t/T = 10^{-3}$, mixed-order polynomials $(k_u, k_p) = (k + 1, k)$, refine level $l = 4 - k$

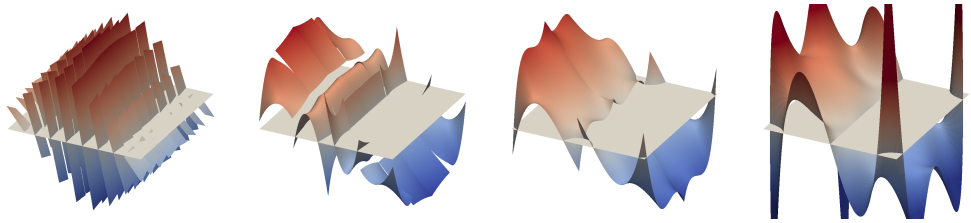
Figure 2.8: Inf–sup stability of dual splitting scheme: pressure solution at time $t = T$ for equal-order polynomials and mixed-order polynomials and different time step sizes. The parameter $k = 1, ..., 4$ varies from $k = 1$ to $k = 4$ from left to right.

equation (2.206) for the dual splitting scheme and equation (2.207) for the *non-incremental* pressure-correction scheme. Figure 2.9 shows results of the same stability experiment using the *incremental* pressure-correction scheme in rotational formulation. In contrast to the dual splitting scheme, inf–sup instabilities also occur for very large time step sizes when using equal-order polynomials. Again, no oscillations occur for the mixed-order formulation. The coupled solution approach yields results similar to those for the incremental pressure-correction scheme in rotational form. These results can be seen as a numerical verification of equation (2.207), stating that the incremental pressure-correction scheme and the coupled solution approach behave similarly in terms of inf–sup stability.

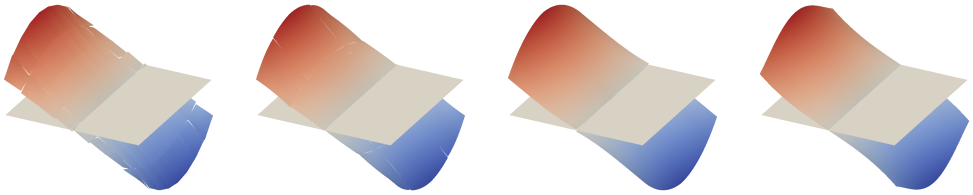
Additional spatial convergence tests shown in Fehn et al. (2017) reveal that equal-order polynomials lead to increased errors and also to sub-optimal rates of convergence (mainly for the pressure solution but also for the velocity solution). The situation is improved for the dual splitting scheme compared to the coupled solver and the pressure-correction scheme, but the effect of inf–sup instabilities is present nevertheless. For all schemes, optimal rates of convergence are obtained for mixed-order polynomials. Due to the influence of the spatial resolution and the time step size, it is therefore recommended to use an inf–sup stable formulation also for projection schemes that contain an inf–sup stabilization term. As a sidenote, the term *instability* might be



(a) $\Delta t/T = 10^{-1}$, equal-order polynomials $(k_u, k_p) = (k, k)$, refine level $l = 4 - k$



(b) $\Delta t/T = 10^{-3}$, equal-order polynomials $(k_u, k_p) = (k, k)$, refine level $l = 4 - k$



(c) $\Delta t/T = 10^{-3}$, mixed-order polynomials $(k_u, k_p) = (k + 1, k)$, refine level $l = 4 - k$

Figure 2.9: Inf–sup stability of *incremental* pressure-correction scheme in rotational formulation: pressure solution at time $t = T$ for equal-order polynomials and mixed-order polynomials and different time step sizes. The parameter $k = 1, ..., 4$ varies from $k = 1$ to $k = 4$ from left to right. Due to large pressure oscillations, the pressure solution for $k = 1$ (left picture) in subfigure 2.9(b) is scaled by a factor of 0.2 compared to all other pressure plots.

ambiguous given that polynomial spaces not satisfying the inf–sup condition result in increased errors rather than a blow-up of the simulation.

2.6.4 Pressure-robustness

This section briefly discusses the aspect of pressure-robustness (see Section 2.4.4) considering a manufactured solution of the steady Stokes equations in two space dimensions. The example considered here is taken from Lederer et al. (2017), Linke (2014) and is widely used in this context. The pressure solution is $p = x_1^5 + x_2^5 - \frac{1}{3}$, the velocity solution is $\boldsymbol{u} = (\nabla \times \boldsymbol{\psi})_{2d}$, with $\boldsymbol{\psi} = (0, 0, \psi)^{\mathrm{T}}$ and $\psi = x_1^2(x_1 - 1)^2 x_2^2(x_2 - 1)^2$. Hence, this problem has a complicated pressure solution relative to the velocity solution. The right-hand side is set to $\boldsymbol{f} = -\nu\nabla^2\boldsymbol{u} + \nabla p$ (method of manufactured solutions). The problem is solved on the domain $\Omega = [0, 1]^2$ with pure Dirichlet boundary conditions. Since a steady Stokes problem is considered, the coupled solution approach is used and the penalty terms (if considered) are added to the momentum

Table 2.4: Pressure-robustness test: relative L^2-errors for velocity and pressure for a mesh with refinement level $l = 2$ (4^2 elements) and polynomial degree $k = 4$ for different values of the viscosity ν and for different penalty factors ζ.

ν	relative L^2-error ($\zeta = 0$)		relative L^2-error ($\zeta = 1$)		relative L^2-error ($\zeta = 10$)	
	$\boldsymbol{u}_h$	p_h	$\boldsymbol{u}_h$	p_h	$\boldsymbol{u}_h$	p_h
10^0	2.30E–05	3.78E–04	2.30E–05	3.78E–04	2.29E–05	3.78E–04
10^{-1}	2.30E–04	3.78E–04	2.29E–04	3.78E–04	2.28E–05	3.77E–04
10^{-2}	2.30E–03	3.78E–04	2.28E–03	3.77E–04	2.13E–03	3.71E–04
10^{-3}	2.30E–02	3.78E–04	2.13E–02	3.71E–04	1.37E–02	3.44E–04
10^{-4}	2.30E–01	3.78E–04	1.36E–01	3.44E–04	3.48E–02	3.23E–04
10^{-5}	2.30E+00	3.78E–04	3.48E–01	3.23E–04	4.33E–02	3.20E–04
10^{-6}	2.30E+01	3.78E–04	4.36E–01	3.20E–04	5.79E–01	3.20E–04
10^{-7}	2.30E+02	3.78E–04	1.26E+01	3.20E–04	5.32E+01	3.20E–04

equation. The velocity–pressure coupling terms are used in weak formulation. Solver tolerances are $\varepsilon_{\text{abs}} = 10^{-12}$ and $\varepsilon_{\text{rel}} = 10^{-8}$. The mesh is uniform Cartesian with $(2^l)^2$ elements. Only inf–sup stable mixed-order polynomials are considered.

When selecting a degree of $k_u = 6$ (and $k_p = 5$), the discretization scheme is able to obtain the exact solution since the analytical solution is within the space of discrete solution functions for both velocity and pressure. For a degree of $k_u = 4$, the pressure solution is no longer within the function space, but the velocity space still contains the analytical velocity solution. For a pressure-robust method, one would expect that the discretization scheme finds a solution with vanishing velocity error in this case. However, the discretization scheme considered here is not pressure-robust and the velocity error depends on the pressure and the viscosity. Table 2.4 shows results for viscosities of $\nu = 10^0, \ldots, 10^{-7}$ (decreasing the viscosity in factors of 10) for mesh refinement level $l = 2$ and degree $k = 4$. Results are shown for the case without divergence and continuity penalty terms ($\zeta = 0$), for the stabilized case with the standard penalty factor of $\zeta = 1$, and for a larger penalty factor of $\zeta = 10$. As expected theoretically, the velocity error increases with ν^{-1} and the pressure error is unaffected by the viscosity for the non-stabilized case. For the stabilized case with $\zeta = 1$, the velocity error also increases with ν^{-1} for higher values of the viscosity, but a beneficial effect of the stabilization can be seen towards smaller values of the viscosity. Increasing the penalty parameter to a value of $\zeta = 10$ improves the velocity error for moderately low viscosity values, but increases the error for the smallest viscosity values studied here. However, optimizing the penalty factor is not of primary interest in the present work (given that conditioning of linear systems deteriorates for $\zeta \to \infty$). Instead, the goal is to use one set of parameters for different flow problems. Clearly, the scheme lacks pressure-robustness and errors would become arbitrarily large in the limit $\nu \to 0$. Then, only a finer spatial resolution helps to improve the velocity error (for example, when refining the mesh once, the stabilized scheme with $\zeta = 1$ results in a relative velocity error below 6% for the smallest viscosity of $\nu = 10^{-7}$ studied here). An interesting question is to which extent this sub-optimal

behavior limits applicability to practical flow problems, see also the discussion in Section 2.4.4. The reader is referred to Section 2.6.7 from which the main motivation for the use of divergence and continuity penalty terms originates by studying under-resolved transitional and turbulent flow problems.

2.6.5 Temporal and spatial convergence behavior for smooth problems

This section investigates the convergence behavior of the proposed Navier–Stokes solvers with respect to the temporal discretization and the spatial discretization. In order to verify whether optimal rates of convergence can be achieved, a problem with a smooth analytical solution is studied, the two-dimensional Taylor–Green vortex.

2.6.5.1 Problem description: Two dimensional Taylor–Green vortex

The problem setup considered here follows the test case described in Hesthaven and Warburton (2007). This two-dimensional vortex problem is an analytical solution of the unsteady incompressible Navier–Stokes equations for vanishing body forces $\boldsymbol{f} = \boldsymbol{0}$

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{x},t) &= \begin{pmatrix} -\sin(2\pi x_2) \\ +\sin(2\pi x_1) \end{pmatrix} \exp\left(-4\nu\pi^2 t\right) \ , \\ p(\boldsymbol{x},t) &= -\cos(2\pi x_1)\cos(2\pi x_2)\exp\left(-8\nu\pi^2 t\right) \ . \end{aligned} \tag{2.248}$$

The analytical solution is designed so that the time derivative term balances the viscous term, and the convective term balances the pressure gradient term. The domain $\Omega = [-L/2, L/2]^2$ is a square of length $L = 1$ and the simulations are performed for the time interval $0 \leq t \leq T = 1$. The viscosity is set to $\nu = 0.025$. On domain boundaries, Dirichlet boundary conditions are prescribed at the inflow part of the boundary and Neumann boundary conditions at the outflow part, so that the coordinate axes split each of the four sides of the rectangular domain into a Dirichlet part and a Neumann part, see also Hesthaven and Warburton (2007). An illustration of this flow problem is given in Figure 2.10. Initial conditions as well as the solution at previous instants of time $t_{n-J+1}, ..., t_{n-1}$ required by the BDF scheme and extrapolation scheme for $J > 1$ are obtained by interpolation of the analytical solution. The velocity Dirichlet boundary condition $\boldsymbol{g}_u$, the discrete time derivative term according to equation (2.41), the velocity gradient in normal direction $\boldsymbol{h}_u/\nu$, and the pressure Dirichlet boundary condition g_p are derived from the analytical solution. In case of the coupled solution approach, the Neumann boundary condition is then given as $\boldsymbol{h} = \boldsymbol{h}_u - g_p\boldsymbol{n}$. Since the velocity boundary conditions $\boldsymbol{g}_u$ and $\boldsymbol{h}_u$ and the pressure boundary condition g_p are nontrivial and time-dependent, this flow problem is an appropriate test case to verify the temporal accuracy of the different solution approaches with their respective boundary conditions. It is further an interesting test case since the CFL condition appears to be switched off for this example for moderate Reynolds numbers, which might be explained by the fact that the vortex is not convected in space but only decays over time. This characteristic eases the experimental investigation of the temporal convergence behavior of unsteady Navier–Stokes solvers with an explicit treatment of the convective term, since an "active" CFL condition

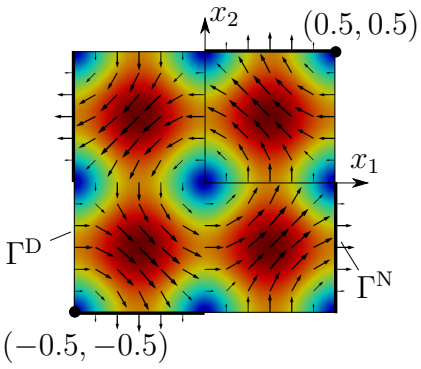
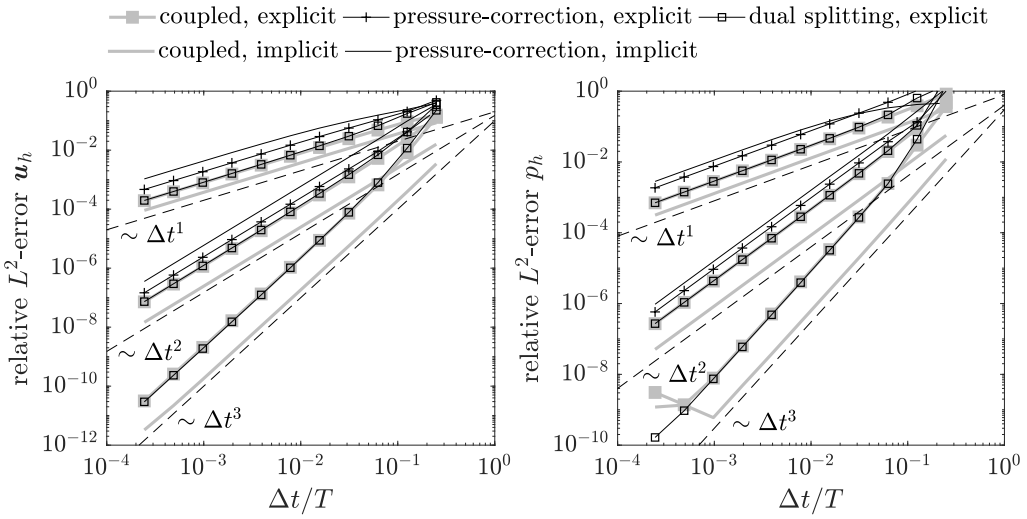


Figure 2.10: Illustration of geometry, boundary conditions, and velocity field (velocity magnitude with arrows indicating flow direction) for two-dimensional vortex problem.

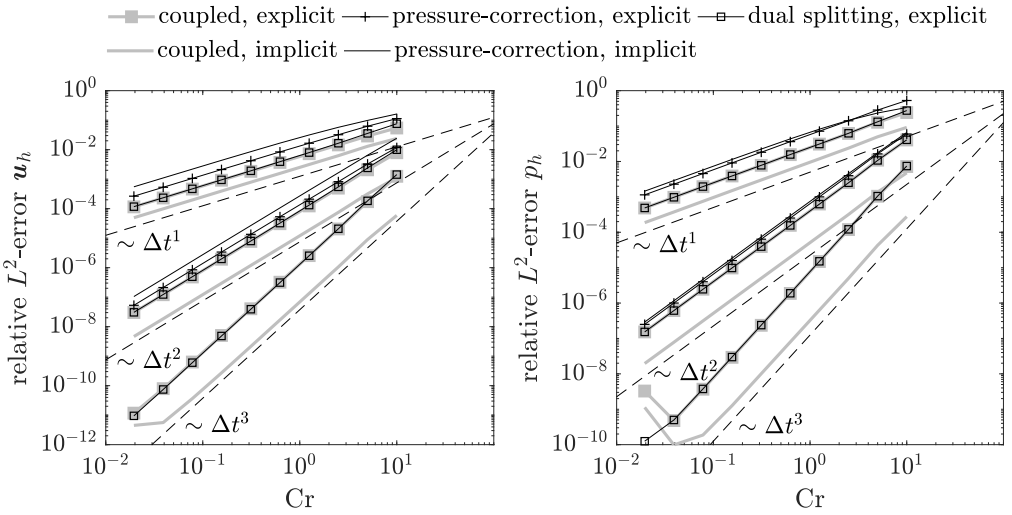
might otherwise lead to time step sizes too restrictive to measure the temporal discretization error compared to the spatial one. Note that for this type of flow problem it can be expected that the solution departs from the analytical solution for high Reynolds numbers and larger times, following two-dimensional high-Reynolds-number vortex dynamics with self-organisation into larger and larger vortices as described for example in Schroeder and Lube (2018). A uniform Cartesian grid is used for discretization in space, where the element length is $h = L/2^l$ for refinement level l. Solver tolerances are $\varepsilon_{\mathrm{abs}} = 10^{-12}$ and $\varepsilon_{\mathrm{rel}} = 10^{-6}$. For nonlinear system of equations, the absolute solver tolerance of the Newton solver is set to $\varepsilon_{\mathrm{abs}} = 10^{-10}$ to ensure convergence for all parameter combinations. Additional results shown in Fehn et al. (2018b) reveal that the divergence and continuity penalty terms do not impact accuracy for this laminar problem with smooth solution. Hence, only the case with consistent penalty terms is shown in the following. The default setup described in Section 2.6.1 is used, apart from the fact that the divergence and continuity penalty terms are added to the momentum equation in case of the coupled solution approach instead of applying these terms in a postprocessing step. The following two subsections show temporal and spatial convergence tests, and the reader is referred to Section 8.5.2 where the same convergence tests are performed for the case of moving meshes.

2.6.5.2 Temporal convergence

To investigate the temporal convergence behavior, a high spatial resolution, refine level $l = 3$ and polynomial degree $k = 8$, is used in order to make sure that the spatial discretization error is negligible. Time integration schemes of order $J = 1, 2, 3$ are investigated for the coupled solution approach and the dual splitting scheme. For the pressure-correction scheme, instabilities have been observed for $J = 3$ in agreement with theory, so that results are only shown for $J = 1, 2$. Moreover, both implicit and explicit formulations of the convective term are investigated. Figure 2.11 shows temporal convergence studies for constant time step sizes and for adaptive time stepping. In all cases, theoretically optimal rates of convergence of order Δt^J are observed. For the coupled solution approach and $J = 3$, the curves level off at some point for very small time step sizes, which might be due to the spatial discretization error or the chosen solver tolerances. In terms of absolute errors, the coupled solution approach with implicit convective term is most accurate. The error is larger for the coupled solution approach with explicit convective term and



(a) constant Δt



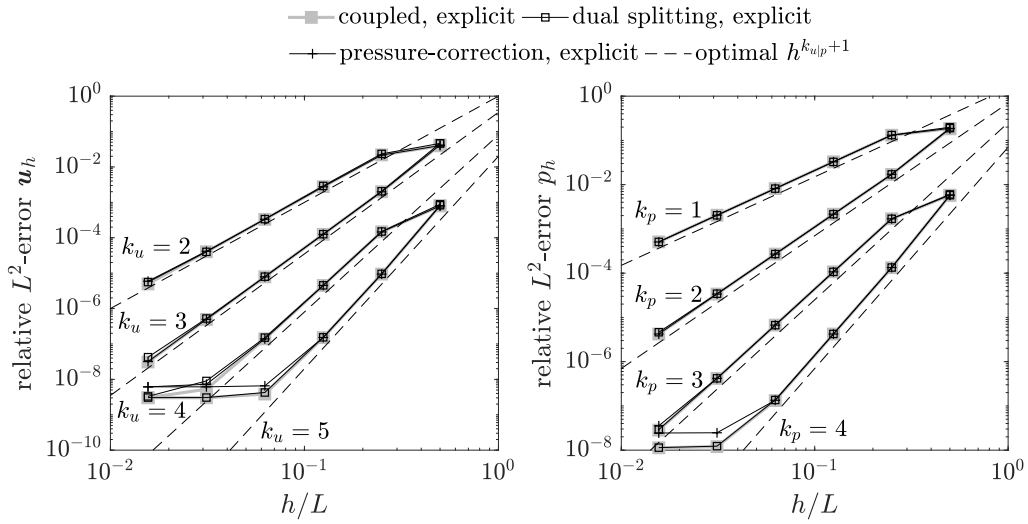
(b) adaptive Δt_n

Figure 2.11: Vortex problem: temporal convergence tests for BDF schemes of order $J = 1, 2, 3$ with $J_p = \min(2, J)$ for the dual splitting scheme and $J_p = \min(2, J) - 1, J \leq 2$ for the pressure-correction scheme. The spatial resolution is $l = 3, k = 8$.

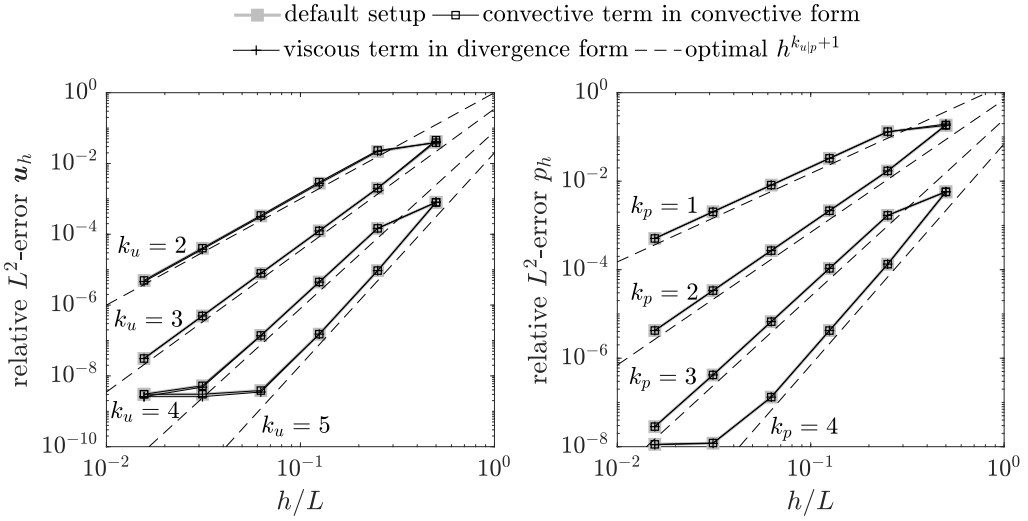
the dual splitting scheme, which yield almost the same temporal accuracy. The error is largest for the pressure-correction scheme, where an explicit treatment of the convective term is more accurate than an implicit one in this case.

2.6.5.3 Spatial convergence

Results of spatial convergence tests are shown in Figure 2.12 for polynomial degrees $k = 2, 3, 4, 5$ and refinement levels $l = 1, \ldots, 6$. The BDF2 time integration scheme is used with



(a) spatial convergence for different solution strategies



(b) spatial convergence for alternative formulations of convective and viscous terms

Figure 2.12: Vortex problem: spatial convergence tests for refinement levels $l = 1, \ldots, 6$ and polynomial degrees $k = 2, 3, 4, 5$. The BDF2 scheme is used with a fix time step size of $\Delta t/T = 5 \cdot 10^{-5}$.

a fix time step size of $\Delta t/T = 5 \cdot 10^{-5}$ so that spatial discretization errors are dominant (for all but the finest spatial resolutions). On the one hand, different solution strategies (coupled approach, dual splitting scheme, pressure-correction scheme) are analyzed for the default setup in terms of the DG discretization, i.e., the convective term is written in divergence formulation and the viscous term in Laplace formulation. All schemes use an explicit treatment of the convective term, and no differences are expected in case of an implicit formulation of the convective term (apart from the temporal error level approached for the finest spatial resolutions). On the other hand, alternative formulations of the convective term and viscous term are investigated,

i.e., the convective formulation of the convective term and the divergence formulation of the viscous term are studied as variants of the default setup. Here, the coupled solution approach with an explicit treatment of the convective term is used for discretization in time. In all cases, optimal rates of convergence of order $h^{k_u|k_p+1}$ are obtained and the errors are virtually the same for all solution techniques and the different formulations of the convective term and viscous term. Additional results are shown in Fehn et al. (2018b), where the penalty terms are applied in a postprocessing step and where optimal rates of convergence are achieved also for this formulation. When interpreting the stabilized DG discretization with penalty terms from the point of view of turbulence modeling (Fehn et al. 2018b), such an implicit turbulence modeling approach ensures that the same discretization can be applied to both laminar and turbulent flow problems, and that the "exact" solution is reproduced in the laminar case.

2.6.6 Laminar flow example – flow around cylinder

In order to demonstrate the geometric flexibility of the present Navier–Stokes solvers and to analyze the efficiency of high-order polynomial spaces for the approximation of velocity and pressure on complex domains with curved boundaries, laminar flow around a cylinder with unsteady vortex shedding is considered. This section studies the benchmark problem proposed in the 1990s by Schäfer et al. (1996), which has found widespread use in terms of the verification of incompressible Navier–Stokes solvers. In the present work, the focus is on the two-dimensional, unsteady test case named 2D-3 for which accurate reference solutions are available (Fehn et al. 2017, John 2004). In the context of high-order DG discretizations, this problem has been investigated for example in Fehn et al. (2017), Lehrenfeld and Schöberl (2016).

2.6.6.1 Problem description

The geometry is displayed in Figure 2.13. The cylinder with center $(x_{1,\mathrm{c}}, x_{2,\mathrm{c}}) = (0.2, 0.2)$ and diameter $D = 0.1$ is located slightly asymmetrically in a rectangular channel of length $L = 2.2$ and height $H = 0.41$. The inflow boundary (left boundary), the channel walls (upper and lower boundary), and the cylinder surface are treated as Dirichlet boundaries. At the inflow boundary, a parabolic velocity profile is prescribed (Schäfer et al. 1996)

$$g_{u_1}(x_1 = 0, x_2, t) = U_{\mathrm{m}} \frac{4 x_2 (H - x_2)}{H^2} \sin(\pi t / T) \ , \ g_{u_2}(x_1 = 0, x_2, t) = 0 \ , \tag{2.249}$$

where the time interval is $0 \leq t \leq T = 8$. The Reynolds number $\mathrm{Re} = \bar{U} D / \nu$ is defined using the mean inflow velocity $\bar{U} = 2 U_{\mathrm{m}} / 3$ and the cylinder diameter D. The maximum inflow velocity is given as $U_{\mathrm{m}} = 1.5$ and the viscosity is $\nu = 10^{-3}$, so that the Reynolds number reaches a maximum value of $\mathrm{Re}_{\mathrm{max}} = 100$ at time $t = T/2$. On the cylinder surface and the channel walls, no slip boundary conditions are prescribed for the velocity, $\boldsymbol{g}_u = \boldsymbol{0}$. The outflow boundary (right boundary) is treated as a Neumann boundary where the benchmark itself does not define a specific boundary condition on Γ^{N}. For the splitting approaches, $\boldsymbol{h}_u = \boldsymbol{0}$ and $g_p = 0$ is prescribed, and $\boldsymbol{h} = \boldsymbol{0}$ in case of the coupled solution approach. The discrete time derivative term according to equation (2.41) is used in the pressure Neumann boundary condition for the dual splitting scheme.

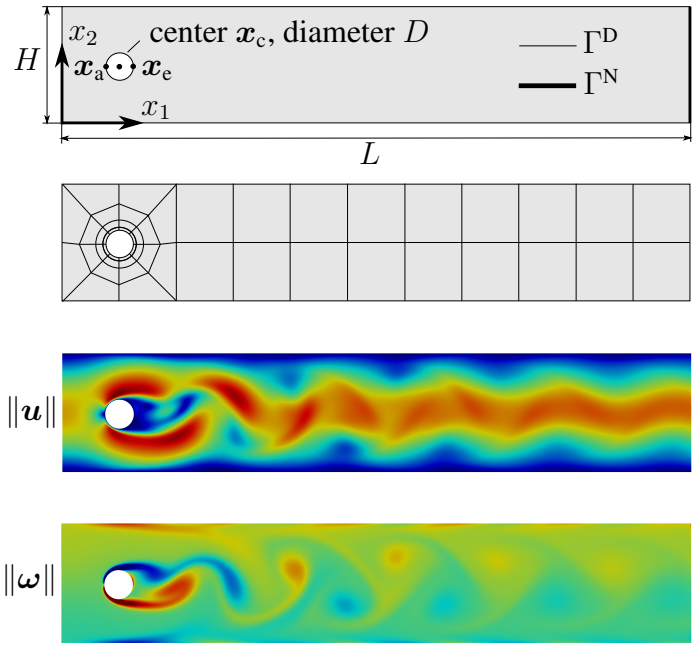


Figure 2.13: Laminar flow around cylinder: geometry and boundary conditions according to the benchmark by Schäfer et al. (1996) as well as coarsest mesh corresponding to refine level $l = 0$. Also shown is a visualization of results for test case 2D-3.

Table 2.5: Laminar flow around cylinder: reference results for test case 2D-3.

reference	$c_{\mathrm{D,max}}$	$c_{\mathrm{L,max}}$	$\Delta p(t = T)$
Schäfer et al. (1996)	$2.95 \pm 2 \cdot 10^{-2}$	$0.48 \pm 1 \cdot 10^{-2}$	$-0.11 \pm 5 \cdot 10^{-3}$
John (2004)	$2.950921575 \pm 5 \cdot 10^{-7}$	$0.47795 \pm 1 \cdot 10^{-4}$	$-0.1116 \pm 1 \cdot 10^{-4}$
Fehn et al. (2017)	2.95091839	0.47788776	−0.11161590

The accuracy of the numerical solution is evaluated by calculating the maximum drag coefficient $c_{\mathrm{D,max}}$, the maximum lift coefficient $c_{\mathrm{L,max}}$, and the pressure difference $\Delta p(t = T) = p(\boldsymbol{x}_\mathrm{a}, t = T) - p(\boldsymbol{x}_\mathrm{e}, t = T)$ between the front and the back of the cylinder at final time $t = T$, where $\boldsymbol{x}_\mathrm{a} = (x_{1,\mathrm{c}} - D/2, x_{2,\mathrm{c}})^\mathsf{T}$ and $\boldsymbol{x}_\mathrm{e} = (x_{1,\mathrm{c}} + D/2, x_{2,\mathrm{c}})^\mathsf{T}$. The drag coefficient $c_\mathrm{D} = F_1/(\rho \bar{U}^2 D/2)$ and the lift coefficient $c_\mathrm{L} = F_2/(\rho \bar{U}^2 D/2)$ are obtained by calculating the force vector $\boldsymbol{F} = (F_1, F_2)^\mathsf{T} = -\rho \int_A \left(-p\boldsymbol{I} + \nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\mathsf{T}\right)\right) \cdot \boldsymbol{n} \mathrm{d}A$ acting on the cylinder, where A denotes the cylinder surface and $\boldsymbol{n}$ the outward pointing normal vector of the computational domain. Reference solutions of these quantities are listed in Table 2.5.

The mesh is visualized in Figure 2.13 for the coarsest refine level $l = 0$ which consists of $N_{\mathrm{el},l=0} = 50$ quadrilateral elements. Finer meshes are obtained by uniform refinement so that the number of elements on level l is $N_{\mathrm{el},l} = N_{\mathrm{el},l=0}(2^d)^l$. The total number of degrees of freedom is $N_{\mathrm{DoFs}} = N_{\mathrm{el},l} N_{\mathrm{DoFs,el}}$, where the number of unknowns per element is $N_{\mathrm{DoFs,el}} = d(k_u + 1)^d + (k_p + 1)^d$. In order to accurately resolve the flow near the cylinder, the mesh is refined towards the cylinder and an isoparametric mapping is used for an improved approxima-

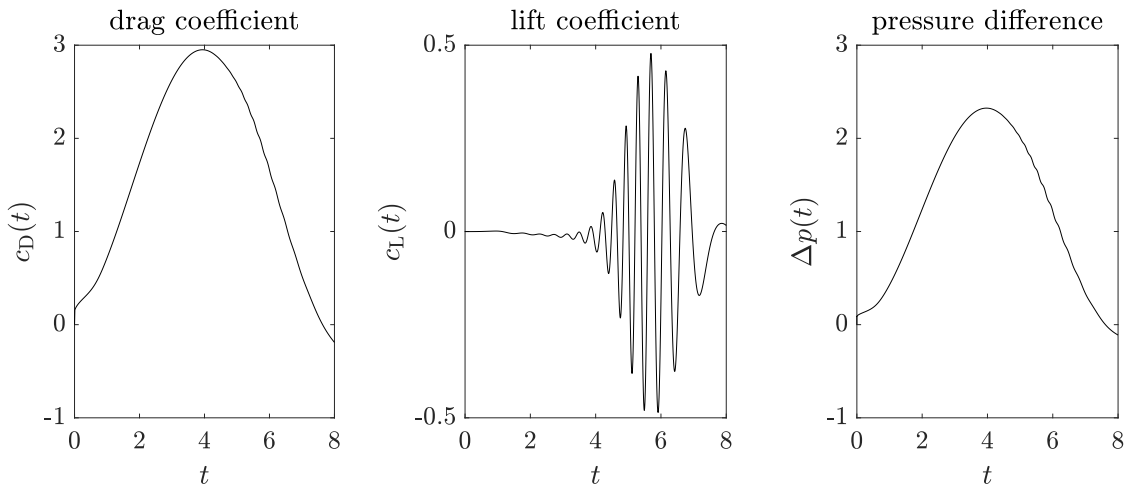


Figure 2.14: Laminar flow around cylinder: drag coefficient $c_{\mathrm{D,max}}$, lift coefficient $c_{\mathrm{L,max}}$, and pressure difference $\Delta p(t = T)$ as a function of time for cylinder test case 2D-3.

tion of the curved cylinder boundary. In this respect, the first two layers of elements around the cylinder are subject to a cylindrical manifold description to enable high-order accuracy. For the third layer of cells, a volume manifold description has been implemented allowing to prescribe a cylindrical manifold for one of the four faces of the quadrilateral element with straight edges on the other faces by transfinite interpolation techniques. Solver tolerances are $\varepsilon_{\mathrm{abs}} = 10^{-12}$ and $\varepsilon_{\mathrm{rel}} = 10^{-6}$.

2.6.6.2 Results

Apart from the results shown here, the work by Fehn et al. (2017) shows additional results demonstrating that integration-by-parts of the velocity–pressure coupling terms (using central fluxes and consistent boundary conditions) is necessary to obtain a robust flow solver for the flow past cylinder test case. For reasons of brevity, the focus is here on the accuracy of high-order discretizations and their suitability for problems with non-trivial geometry. This section addresses the question whether high-order methods require less degrees of freedom to reach the same level of accuracy, and also aims to quantify this aspect. Spatial convergence tests are run for mixed order polynomials $(k_u, k_p) = (k, k-1)$ with polynomial degrees in the range $k = 2, 4, 6, 8, 10$. The high-order dual splitting scheme with $J = 3$ is used, using adaptive time stepping according to equation (2.210) with a Courant number of $\mathrm{Cr} = 0.35$ (so that the overall error is dominated by the spatial discretization error, which has been verified by repeating the simulations for a smaller Courant number of $\mathrm{Cr} = 0.175$). Since there is no analytical solution for this test case, the time integration scheme is started with a low-order scheme in the first time steps. To give an illustration of the flow behavior, Figure 2.14 shows the temporal evolution of the lift and drag coefficients as well as the pressure difference, and Figure 2.13 visualizes the velocity field for an instant of time around which periodic vortex shedding occurs. Accuracy of

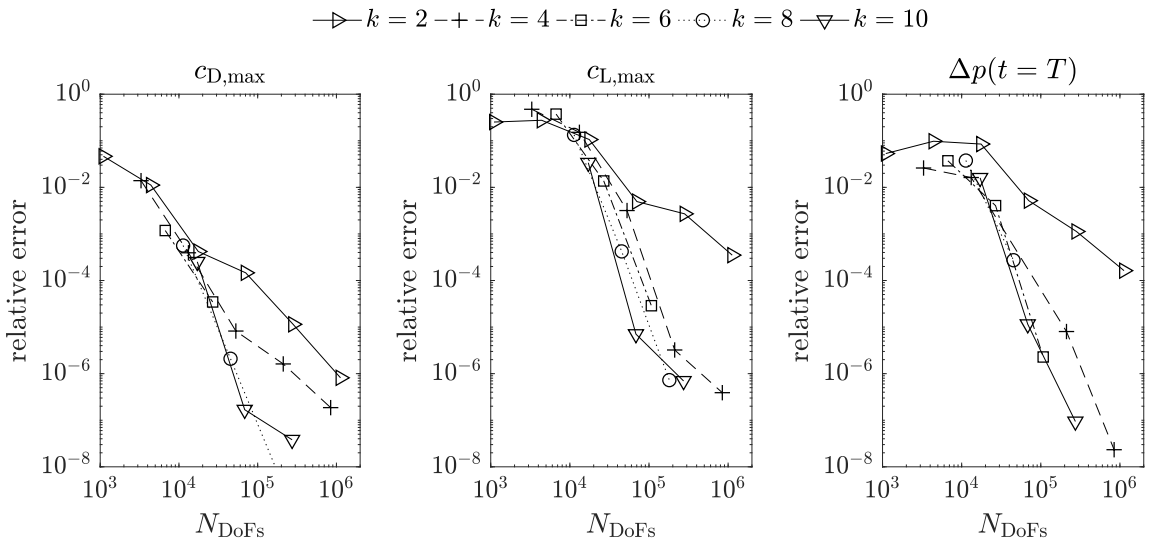


Figure 2.15: Laminar flow around cylinder: h-refinement study for various polynomial degrees $k = 2, 4, 6, 8, 10$. Relative errors of $c_{\mathrm{D,max}}$, $c_{\mathrm{L,max}}$, and $\Delta p(t = T)$ are shown as a function of the number of unknowns N_{DoFs} in order to assess the accuracy of high polynomial degrees.

the simulations is quantified in terms of the maximum lift and drag coefficients and the pressure difference at final time, where the accurate reference solution published in Fehn et al. (2017) and listed in Table 2.5 is used to calculate relative errors for these three quantities. Since this reference solution has been obtained for two different spatial resolutions (refine level $l = 3$ and polynomial degrees $k = 8$ and $k = 10$) as well as different time step sizes, it is assumed to be accurate up to all decimal places.

Figure 2.15 compares the accuracy of different polynomial degrees in terms of error vs. number of unknowns. In contrast to the previous example, the convergence does not follow the theoretically optimal behavior expected for smooth solutions. Nevertheless, the results in Figure 2.15 demonstrate that high-order polynomial degrees can significantly reduce the number of unknowns required to obtain a certain level of accuracy, and are essential to obtain solutions of highest accuracy for a given (maximum) number of unknowns. This can be seen by comparing the results for $k = 6$ to those for $k = 2$. At the same time, one observes that the increase in accuracy saturates for polynomial degrees $k = 4$ or $k = 6$ and higher (depending on the quantity of interest and the accuracy of interest). From an engineering perspective where errors around 10^{-2} might be acceptable, high-order methods of degree $k = 4, 6, 8, 10$ have difficulties in outperforming the lower-order variant $k = 2$.

2.6.7 Robustness and accuracy for transitional and turbulent flows

2.6.7.1 Orr–Sommerfeld problem

In a first example towards transitional and turbulent flows, the stability of the proposed incompressible Navier–Stokes solvers is analyzed for the Orr–Sommerfeld stability problem applied to the two-dimensional Poiseuille flow problem, which has been analyzed, e.g., in Fischer (1997), Shahbazi et al. (2007).

2.6.7.1.1 Problem description The computational domain is a rectangular channel with dimensions $[0, L] \times [-H, H]$. No-slip boundary conditions are prescribed at $x_2 = \pm H$ and periodic boundary conditions in streamwise direction. Due to the periodic boundary conditions, a constant body force, $f_1 = 2\nu U_{\text{max}}/H^2$, has to be prescribed to sustain the mean flow, which is given as $U_1(x_2)/U_{\text{max}} = 1 - (x_2/H)^2$. To obtain the initial solution for this problem, one has to solve the Orr–Sommerfeld equation. The Orr–Sommerfeld equation is derived by superimposing a small disturbance upon the mean flow $\boldsymbol{U}(\boldsymbol{x}) = (U_1(x_2), 0)^{\mathsf{T}}$ that fulfills the incompressible Navier–Stokes equations. The Navier–Stokes equations are subsequently linearized by assuming that the perturbation is small. To obtain the Orr–Sommerfeld equation, the following ansatz is used

$$u_1(\boldsymbol{x}, t) = U_1(x_2) + \varepsilon \,\mathrm{Re}\left\{ \frac{\mathrm{d}\psi(x_2)}{\mathrm{d}x_2} \exp\left(i\left(\alpha x_1 - \omega t\right)\right) \right\} , \tag{2.250}$$

$$u_2(\boldsymbol{x}, t) = -\varepsilon \,\mathrm{Re}\left\{ i\alpha\psi(x_2) \exp\left(i\left(\alpha x_1 - \omega t\right)\right) \right\} . \tag{2.251}$$

The perturbation velocity is based on the streamfunction $\Psi(\boldsymbol{x}, t) = \varepsilon\, \psi(x_2) \exp\left(i\left(\alpha x_1 - \omega t\right)\right)$ with wavenumber α, complex frequency ω, and perturbation amplitude $\varepsilon \ll 1$. The Orr–Sommerfeld equation then reads

$$i\alpha \left[\left(U_1 - \frac{\omega}{\alpha} \right) \left(\psi'' - \alpha^2 \psi \right) - U_1'' \psi \right] = \nu \left(\psi'''' - 2\alpha^2 \psi'' + \alpha^4 \psi \right) , \tag{2.252}$$

where $(\cdot)' = \frac{\mathrm{d}(\cdot)}{\mathrm{d}x_2}$. The Orr–Sommerfeld equation is a fourth-order homogeneous ordinary differential equation with variable coefficients. The boundary conditions are $\psi(-H) = \psi(H) = 0$ and $\psi'(-H) = \psi'(H) = 0$. This equation is typically solved for the complex frequency $\omega = \omega_{\text{r}} + i\omega_{\text{i}}$ and for $\psi(x_2)$, by prescribing a wavenumber α, a Reynolds number $\mathrm{Re} = U_{\text{max}} H/\nu$ or viscosity ν, and the mean flow $U_1(x_2)$. Following Fischer (1997), the parameters are $H = 1, U_{\text{max}} = 1$, $\mathrm{Re} = 7500$, $\alpha = 1$, and $\varepsilon = 10^{-5}$. The Orr–Sommerfeld equation is discretized using a spectral Galerkin ansatz with one finite element of high polynomial degree k (e.g., $k = 200$), resulting in a generalized eigenvalue problem $\boldsymbol{A}\boldsymbol{\Psi} = \lambda \boldsymbol{B}\boldsymbol{\Psi}$ for the eigenvalue $\lambda = -i\omega$ and the eigenvector $\boldsymbol{\Psi} = (\Psi_1, ..., \Psi_{k+1})^{\mathsf{T}}$. The initial solution prescribed for the Orr–Sommerfeld simulations performed below is given by equations (2.250) and (2.251), where ω is the complex frequency corresponding to the only unstable eigensolution of the Orr–Sommerfeld equation, $\omega_{\text{i}} > 0$, and where $\psi(x_2) = \sum_{i=1}^{k+1} \ell_i^k(x_2)\Psi_i$ is the interpolation of the corresponding eigenvector $\boldsymbol{\Psi}$, which is normalized to a maximum value of 1, $\max_i |\Psi_i| = 1$. The length L of the computational domain equals the wavelength of the Tollmien–Schlichting (TS) waves, $L = 2\pi/\alpha$. The simulations are performed for the time interval $0 \leq t \leq T = 2T_0$, where $T_0 = \alpha L/\omega_{\text{r}}$ is the time the TS waves need to travel through the computational domain.

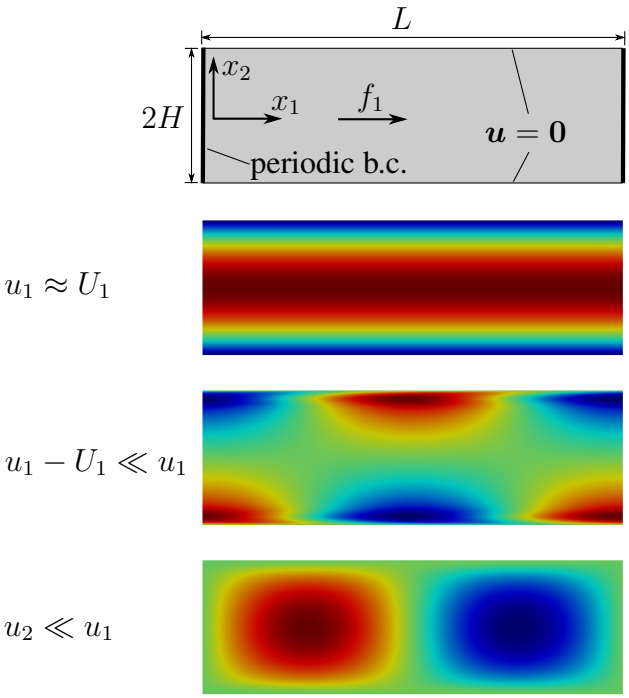


Figure 2.16: Orr–Sommerfeld problem: geometry and boundary conditions as well as visualization of velocity field.

The perturbation energy $E = \int_\Omega \|\boldsymbol{u} - \boldsymbol{U}\|^2 \mathrm{d}\boldsymbol{x}$ grows exponentially in time according to linear stability theory, $E(t)/E(t=0) = \exp(2\omega_\mathrm{i} t)$. Similar to Fischer (1997), the quantity $e(t) = |\exp(2\omega_\mathrm{i} t) - E_h(t)/E_h(t=0)|$ is used as an error measure, where E_h is the perturbation energy calculated using the numerical solution $\boldsymbol{u}_h$.

The computational domain is discretized using a uniform Cartesian grid where l denotes the number of refinements with $N_\mathrm{el}(l=0) = 1$. The time step size is calculated according to the CFL condition (2.209) with $\mathrm{Cr} = 0.2$ and $\|\boldsymbol{u}\|_\mathrm{max} = U_\mathrm{max}$. The resulting time step sizes are small enough so that the error is dominated by the spatial discretization error, which has been verified by comparing the results for $\mathrm{Cr} = 0.2$ with those for a smaller Courant number of $\mathrm{Cr} = 0.1$. The coupled solution approach is applied (BDF2 scheme) using small absolute and relative solver tolerances of 10^{-14} (which turned out to be important for this test case to allow low error levels), and penalty terms (if activated) are applied in a postprocessing step.

2.6.7.1.2 Results To analyze the robustness and stability properties of the stabilized approach with divergence and continuity penalty terms, simulations are performed for various polynomial degrees $k = 2, 4, 6, 8, 10$ and refinement levels $l = 0, 1, 2, ...$ beginning with very coarse spatial resolutions consisting of only one element. For $k = 2$, refinement levels of $l = 1, 2, ...$ are investigated since the perturbation energy would be zero for refinement level $l = 0$ at initial time $t = 0$. A comparison of the stability of the standard DG discretization without penalty terms and the stabilized approach including both the divergence and continuity penalty terms is shown in Figure 2.17 for polynomial degree $k = 2$ and refinement levels $l = 4, 5, 6$. For the formula-

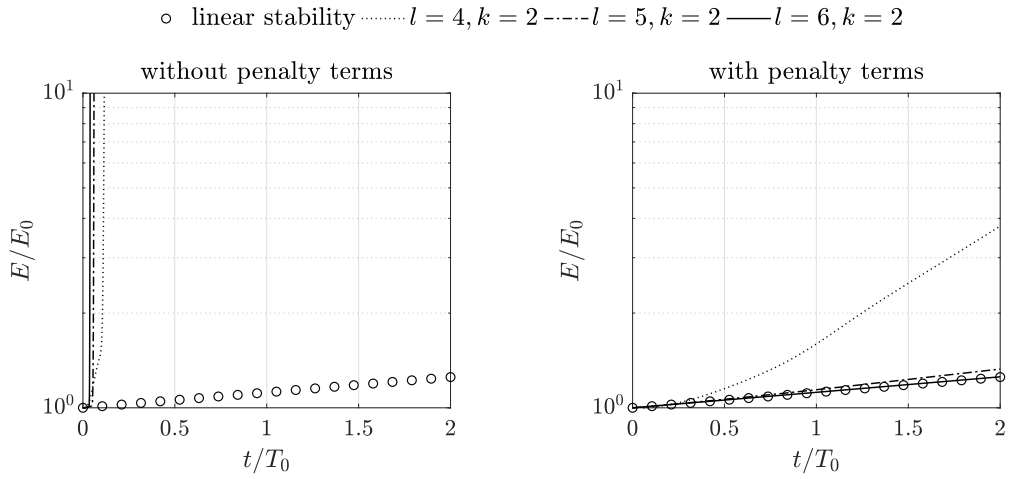


Figure 2.17: Orr–Sommerfeld stability problem: stability for polynomial degree $k = 2$ and refine levels $l = 4, 5, 6$ using the coupled solution approach.

tion without penalty terms, an unphysical growth of the perturbation energy occurs as previously observed in Shahbazi et al. (2007). Using both divergence and continuity penalty terms, the method is stable for all refinement levels and all polynomial degrees. The results shown in Figure 2.17 demonstrate that the solution tends towards the theoretical value for increasing spatial resolution. Additional results shown in Fehn et al. (2018b) show that the divergence penalty term only is not able to stabilize the basic DG formulation and leads to an unphysical growth of the perturbation energy as well. The stability has also been analyzed for the dual splitting scheme and pressure-correction scheme, and the results are in qualitative agreement with those for the coupled solution approach. By the present stabilized DG approach, the instabilities reported in Shahbazi et al. (2007) can be overcome.

Remark 2.15 *While mixed-order polynomials are considered here, an unphysical growth of the perturbation energy has also been observed for equal-order polynomials on coarse meshes when using the basic DG discretization without stabilization terms. While this behavior appears plausible, stability has been reported in Shahbazi et al. (2007) for equal-order polynomials. This might be explained by the fact that the spatial resolutions considered there are significantly finer than the ones analyzed here (Shahbazi et al. (2007) consider a mesh consisting of* 128 *triangles for polynomial degrees* $k = 6, 7, 8$*).*

To analyze the convergence properties for the Orr–Sommerfeld stability problem in more detail, Figure 2.18 shows the error as a function of the number of unknowns for various polynomial degrees and refinement levels, where the formulation with divergence and continuity penalty terms is used. Dashed lines indicate optimal rates of convergence, where the slope of $N_{\mathrm{DoFs}}^{-(k+1)/d}$ is due to the error decreasing with h^{k+1} and unknowns increasing with h^{-d}. For low refinement levels, the discretization operates in the pre-asymptotic regime of convergence. For higher refinement levels entering the asymptotic regime, optimal rates of convergence are obtained, which is to be expected given that the solution is relatively smooth for the Orr–Sommerfeld problem. For $N_{\mathrm{DoFs}} > 10^3$, high-order methods are systematically more accurate than low-order methods

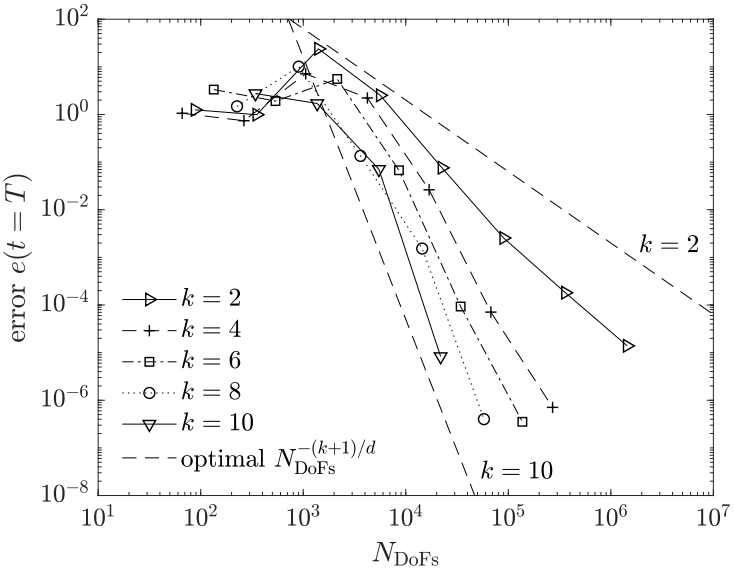


Figure 2.18: Orr–Sommerfeld stability problem: convergence tests and accuracy of high-order methods for polynomial degrees $k = 2, 4, 6, 8, 10$ using the coupled solution approach.

for the same number of unknowns, and the number of unknowns required to reach a certain level of accuracy can be reduced by a factor of 10 to 100 for high polynomial degrees as compared to the low-order method with $k = 2$.

2.6.7.2 3D Taylor–Green vortex problem

The three-dimensional Taylor–Green vortex problem (Taylor and Green 1937) is a widely used benchmark problem to assess the accuracy as well as computational efficiency of numerical methods for the simulation of transitional and turbulent flows. It is characterized by a trivial cube geometry and a simple laminar initial flow field, where vortices break down into complex turbulent flow structures at later times. This problem has already been studied in Brachet et al. (1983), Brachet (1991) using direct spectral numerical simulation, and is a standard benchmark in the high-order CFD community (Wang et al. 2013). In the context of high-order DG discretizations for the incompressible Navier–Stokes equations, the Taylor–Green vortex problem has been analyzed for example in Fehn et al. (2018a,b, 2019a), Piatkowski et al. (2018), Schroeder (2019).

2.6.7.2.1 Problem description The problem setup for the Taylor–Green vortex follows the benchmark definition from Wang et al. (2013). The computational domain Ω_h is a box $\Omega_h = [-\pi L, \pi L]^3$ where L is a characteristic length scale. Periodic boundary conditions are prescribed in all coordinate directions and the body force vector is zero, $\boldsymbol{f} = \boldsymbol{0}$. The initial solution for the

three-dimensional Taylor–Green vortex problem is given as

$$u_1(\boldsymbol{x}, t=0) = U_0 \sin\left(\frac{x_1}{L}\right) \cos\left(\frac{x_2}{L}\right) \cos\left(\frac{x_3}{L}\right) \,, \tag{2.253}$$

$$u_2(\boldsymbol{x}, t=0) = -U_0 \cos\left(\frac{x_1}{L}\right) \sin\left(\frac{x_2}{L}\right) \cos\left(\frac{x_3}{L}\right) \,, \tag{2.254}$$

$$u_3(\boldsymbol{x}, t=0) = 0 \,, \tag{2.255}$$

$$p(\boldsymbol{x}, t=0) = p_0 + \frac{U_0^2}{16}\left(\cos\left(\frac{2x_1}{L}\right) + \cos\left(\frac{2x_2}{L}\right)\right)\left(\cos\left(\frac{2x_3}{L}\right) + 2\right) \,. \tag{2.256}$$

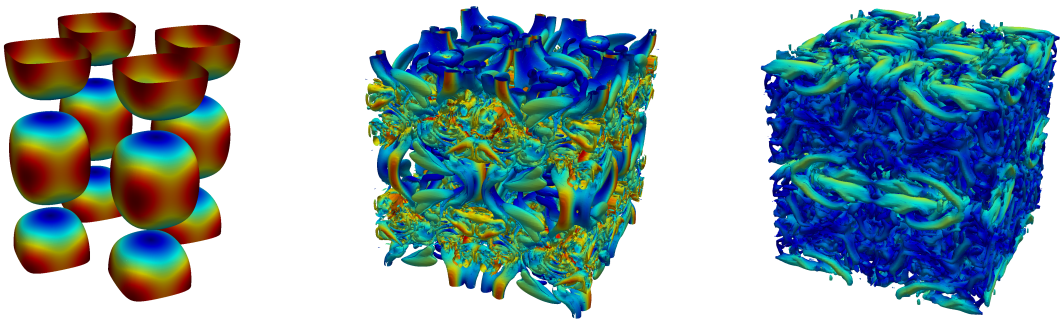
The Reynolds number is defined as $\mathrm{Re} = \frac{U_0 L}{\nu}$, where a Reynolds number of $\mathrm{Re} = 1600$ is considered in the following. The parameters are set to $p_0 = 0$ and $U_0 = 1$, $L = 1$, so that the viscosity is given as $\nu = \frac{1}{\mathrm{Re}}$. The simulations are performed for the time interval $0 \leq t \leq T$ where an end time of $T = 20T_0$ is used with $T_0 = \frac{L}{U_0}$. The mesh is discretized using a uniform Cartesian grid consisting of $(2^l)^d$ elements where l denotes the level of refinement. Using mixed-order polynomials for velocity and pressure, the total number of unknowns is given as $N_{\mathrm{DoFs}} = (2^l)^d(d(k_u+1)^d + (k_p+1)^d) = (2^l)^d(d(k+1)^d + k^d)$. Moreover, the effective spatial resolution is defined as $(2^l(k+1))^d$, i.e., the effective resolution is the number of unknowns of one component of the d-dimensional velocity vector. To give an example, the effective resolution is 64^3 for refine level $l = 3$ (with 8 elements per direction) and polynomial degree $k = 7$. A visualization of the solution at times $t = 0, 10T_0, 20T_0$ is shown in Figure 2.19.

Solver tolerances are $\varepsilon_{\mathrm{abs}} = 10^{-12}$ and $\varepsilon_{\mathrm{rel}} = 10^{-3}$. This choice is motivated by detailed investigations on the sensitivity of solver tolerances on the accuracy of results shown in Fehn et al. (2019c), where it was found that accurate results are obtained for the (comparably large) relative solver tolerance used here. Unless specified otherwise, results shown in the following have been obtained with the dual splitting scheme, and differences between the different solution strategies can be expected to be negligible. The BDF2 time integration scheme is used along with adaptive time stepping and a constant Courant number of $\mathrm{Cr} = 0.4$ relatively close to the stability limit. This choice is based on an observation made in Fehn et al. (2018a) for this TGV problem, namely that temporal discretization errors are negligible for under-resolved simulations characterized by comparably large spatial discretization errors if the time step size is restricted according to the CFL condition.

The accuracy of numerical results is evaluated by calculating the kinetic energy and its dissipation rate. The total kinetic energy is defined as

$$E = \frac{1}{V_{\Omega_h}} \int_{\Omega_h} \frac{1}{2} \boldsymbol{u}_h \cdot \boldsymbol{u}_h \, \mathrm{d}\Omega \,, \tag{2.257}$$

where the volume of the computational domain is $V_{\Omega_h} = \int_{\Omega_h} 1 \, \mathrm{d}\Omega$. These integrals are calculated numerically using Gaussian quadrature with k_u+1 quadrature points. The kinetic energy dissipation rate $-\frac{\mathrm{d}E}{\mathrm{d}t}$ is calculated using a second-order accurate central difference scheme (for variable time step sizes if adaptive time stepping is used) for interior time points $i = 1, ..., N_{\Delta t} - 1$ and a first-order accurate, one-sided finite difference scheme for the end points $i = 0$ and $i = N_{\Delta t}$.



(a) $t = 0$ (b) $t = 10\,T_0$ (c) $t = 20\,T_0$

Figure 2.19: Visualization of Taylor–Green vortex problem at $\mathrm{Re} = 1600$: iso-surfaces of Q-criterion (value of $0.1U_0/L$) colored by velocity magnitude (same color scale for all time instants, red indicates high velocity and blue low velocity). The stabilized approach including divergence and continuity penalty terms is used and the spatial resolution is $l = 6$ and $k = 3$ (effective resolution of 256^3).

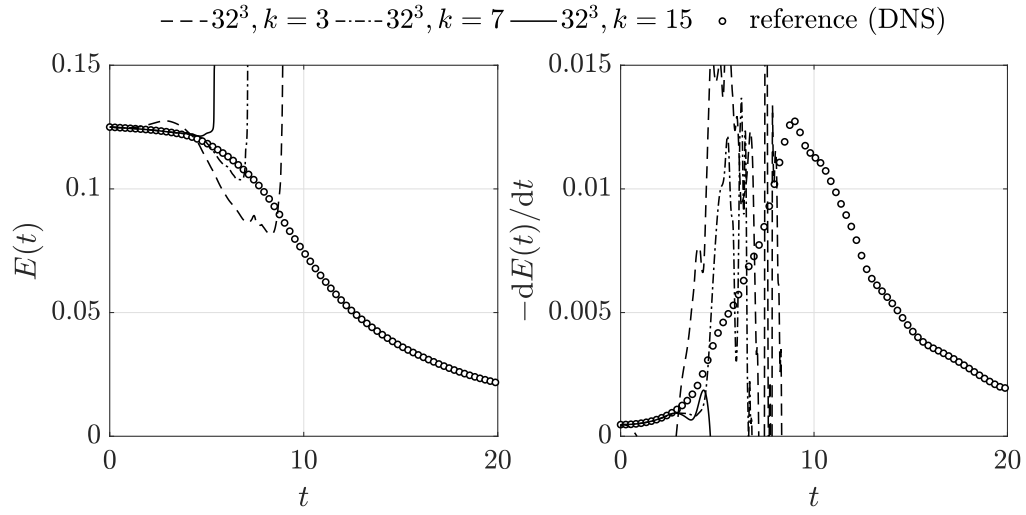
Remark 2.16 *The molecular energy dissipation rate ε (also denoted as physical dissipation rate or the dissipation rate of the resolved scales) is typically calculated as (Gassner and Beck 2013)*

$$\varepsilon = \frac{\nu}{V_{\Omega_h}} \int_{\Omega_h} \nabla \boldsymbol{u}_h : \nabla \boldsymbol{u}_h \, \mathrm{d}\Omega \,, \tag{2.258}$$

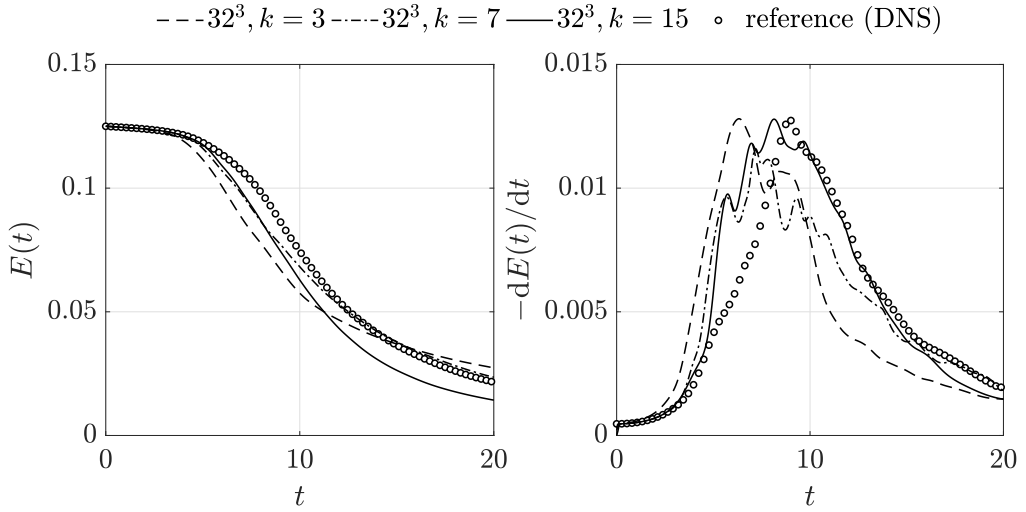
and is also frequently used to assess the accuracy of high-order methods. Using the kinetic energy dissipation rate $-\frac{\mathrm{d}E}{\mathrm{d}t}$ and the molecular dissipation rate ε, a numerical dissipation of the discretization scheme can be obtained as $-\frac{\mathrm{d}E}{\mathrm{d}t} - \varepsilon$. The present work mainly focuses on the temporal evolution of the kinetic energy and its dissipation rate, as this is considered the more relevant metric in practice. From the point of view of implicit large eddy simulation, one might argue that it is not of primary importance by which mechanism the dissipation is realized exactly, as long as the overall scheme is able to predict the dissipation rate accurately, i.e., the present work understands LES as a numerical approach rather than a modeling approach. Results on physical and numerical dissipation rates for the present DG scheme are given in Fehn et al. (2018a,b). The contribution of different terms of the Navier–Stokes equations or stabilization terms of the discretiztion scheme to the overall dissipation rate is analyzed in detail in Fehn et al. (2019a). The reader is referred to Lehrenfeld et al. (2018) for some interesting subtleties related to splitting dissipation rates into physical and numerical contributions in the context of DG discretizations.

In order to evaluate accuracy quantitatively, a relative L^2-error norm $e_{\mathrm{d}E/\mathrm{d}t}$ is defined for the kinetic energy dissipation rate $\frac{\mathrm{d}E}{\mathrm{d}t}$

$$e^2_{\mathrm{d}E/\mathrm{d}t} = \frac{\int_{t=0}^{T} \left(\frac{\mathrm{d}E(t)}{\mathrm{d}t} - \frac{\mathrm{d}E_{\mathrm{ref}}(t)}{\mathrm{d}t} \right)^2 \mathrm{d}t}{\int_{t=0}^{T} \left(\frac{\mathrm{d}E_{\mathrm{ref}}(t)}{\mathrm{d}t} \right)^2 \mathrm{d}t} \,. \tag{2.259}$$



(a) standard DG discretization without penalty terms



(b) stabilized DG discretization with divergence and continuity penalty terms

Figure 2.20: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: temporal evolution of kinetic energy and kinetic energy dissipation rate for an effective resolution of 32^3 and polynomial degrees of $k = 3, 7, 15$.

The above integrals are calculated numerically using the trapezoidal rule. For coarse spatial resolutions, the temporal resolution is also lower compared to the reference solution due to the CFL condition. Hence, results written after each time step are first interpolated to the discrete time instants of the reference solution with fine temporal resolution. Then, the trapezoidal rule is evaluated using the time step size of the reference simulation as step size.

2.6.7.2.2 Results Figure 2.20 shows results for the temporal evolution of the kinetic energy and the kinetic energy dissipation rate and compares the standard discretization approach with-

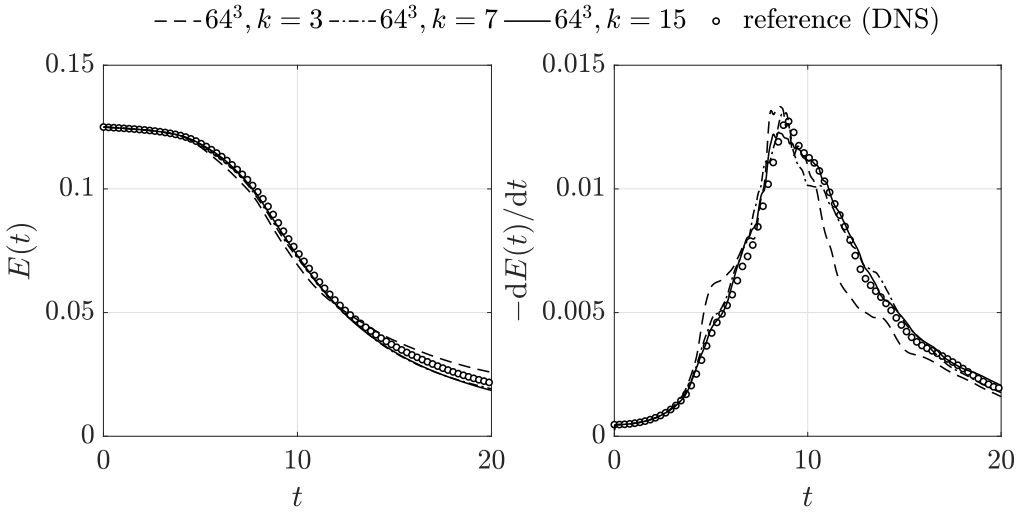


Figure 2.21: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: temporal evolution of kinetic energy and kinetic energy dissipation rate for an effective resolution of 64^3 and polynomial degrees of $k = 3, 7, 15$ using the stabilized formulation including penalty terms.

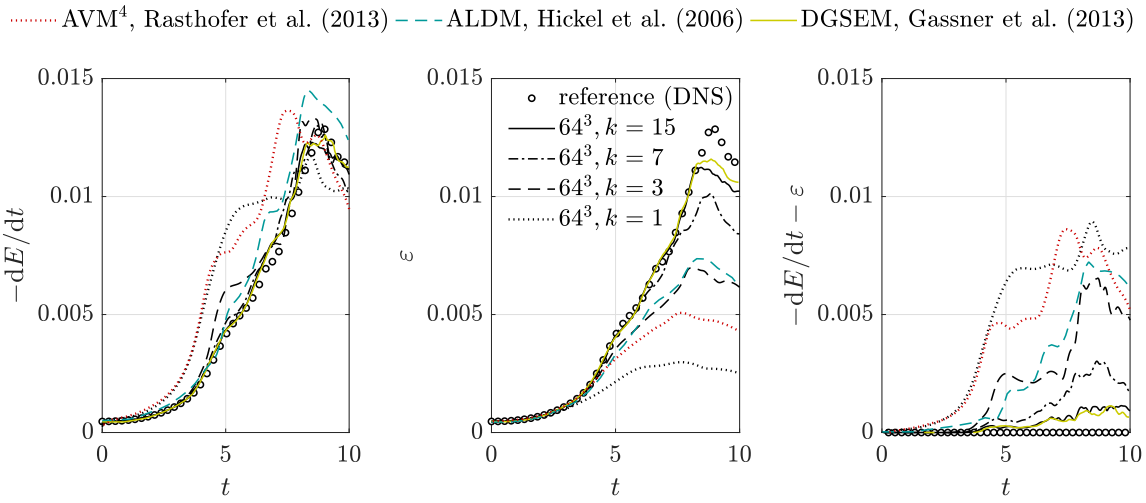


Figure 2.22: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: comparison of kinetic energy dissipation, physical dissipation, and numerical dissipation to state-of-the-art implicit LES approaches from the literature. The effective resolution is 64^3 for all approaches.

out additional stabilization terms to the stabilized formulation with divergence and continuity penalty terms. The spatial resolution is 32^3 for all polynomial degrees $k = 3, 7, 15$ by using mesh refinement levels of $l = 3, 2, 1$, respectively. The formulation without stabilization terms lacks robustness and the kinetic energy blows up for all polynomial degrees. To make sure that these instabilities are not related to the CFL condition, a smaller Courant number of $\mathrm{Cr} = 0.1$ has been used for the simulations without stabilization terms. In contrast, stability is retained by using additional divergence and continuity penalty terms, and the kinetic energy dissipation rate

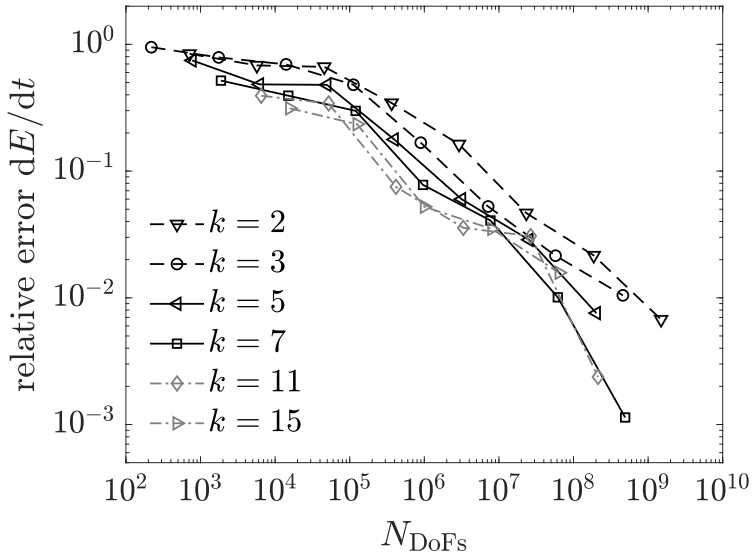


Figure 2.23: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: convergence tests and accuracy of high-order methods for polynomial degrees $k = 2, 3, 5, 7, 11, 15$. An accurate reference solution published in Fehn et al. (2018a) with effective resolution of 1024^3 for polynomial degree $k = 7$ is used to calculate the errors.

is positive throughout the simulations as expected physically. In Fehn et al. (2018b), the stability of the present discretization approach has been investigated in great detail for both the viscous case at $\mathrm{Re} = 1600$ and the inviscid limit $\mathrm{Re} \to \infty$, where the latter is particularly challenging in terms of robustness due to the absence of viscous dissipation.[2] Robustness has been tested for both Cartesian meshes and deformed, curvilinear meshes. Robustness has been achieved for the stabilized approach including divergence and continuity penalty terms for all spatial resolutions covering a wide range of polynomial degrees $k = 2, \ldots, 15$ and mesh refinement levels (starting from coarse meshes consisting of only one element), while the standard formulation without stabilization terms lacks robustness for several spatial resolution parameters, in particular in the inviscid limit.

Figure 2.21 shows results for a finer resolution of 64^3, where only the stabilized approach is considered. Compared to the 32^3 resolution, convergence to the DNS reference solution can clearly be observed. Moreover, higher polynomial degrees tend to predict the dissipation rate more accurately than lower degrees for the same effective resolution. A comparison to state-of-the-art discretization schemes from the literature such as the adaptive local deconvolution method (ALDM) by Hickel et al. (2006) based on a finite volume discretization, the algebraic variational multiscale multigrid multifractal method (AVM^4) proposed in Rasthofer and Gravemeier (2013), and the implicit LES computations published in Gassner and Beck (2013) for a compressible high-order DG solver at degree $k = 15$ is shown in Figure 2.22. This figure shows the kinetic energy dissipation (left), the phyiscal dissipation (middle), and the numerical dissipa-

[2]For example, the approach proposed in Ferrer (2017) to stabilize a DG scheme for under-resolved turbulent flow simulations by scaling the interior penalty parameter of the viscous term can not have an effect if the viscosity is zero. It is therefore imperative to study the robustness of a turbulent flow solver in the inviscid limit.

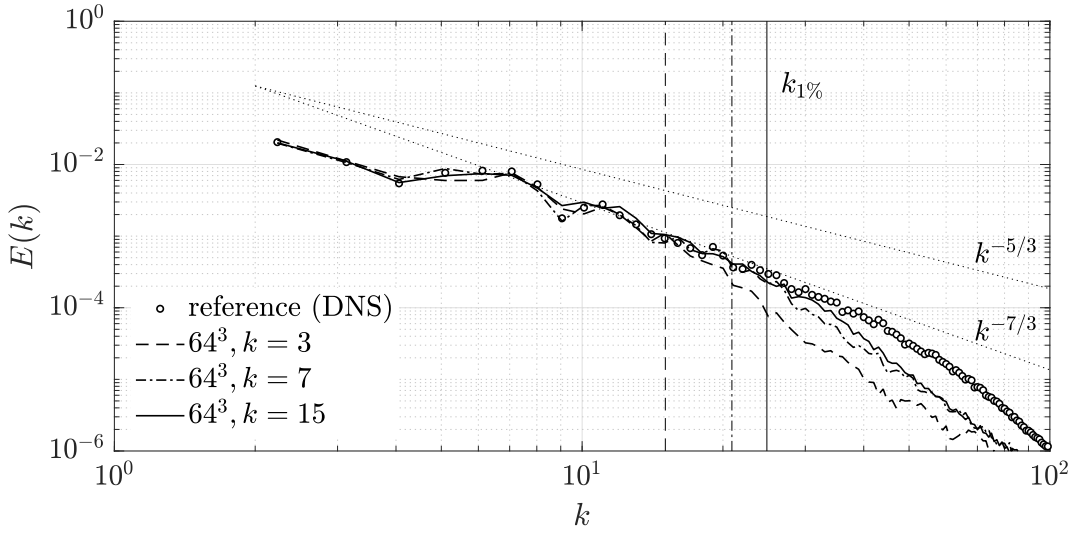


Figure 2.24: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: kinetic energy spectra $E(k)$ at time $t = 9$ for an effective resolution of 64^3 for polynomial degrees $k = 3, 7, 15$. Note the clash of notation in terms of polynomial degree k and wavenumber k.

tion (right). The results for the present DG discretization for polynomial degrees $k = 1, 3, 7, 15$ reveal the well-known behavior that low polynomial degrees are more dissipative, while the dissipation of the resolved scales is higher for large polynomial degrees. As expected, the present results at $k = 15$ agree very well with those shown in Gassner and Beck (2013) for a compressible DG solver at low Mach number for the same degree. The ALDM model exhibits dissipation characteristics comparable to the present $k = 3$ scheme, and the results obtained for AVM^4 are comparable to the present $k = 1$ scheme (which uses an equal-order formulation $k_u = k_p = 1$ compared to the mixed-order formulation used for $k = 3, 7, 15$, in order to make sure that the pressure shape functions are at least linear). This is plausible given that the variational multiscale approach from Rasthofer and Gravemeier (2013) is based on a low-order continuous finite element discretization of degree $k = 1$, where the number of elements is 64^3 for the simulation shown here. Note, however, that the number of elements is only 32^3 for the present DG discretization to reach the same effective resolution due to the duplication of degrees of freedom between elements in the DG case (a DG simulation with 64^3 elements turned out to be significantly more accurate). All in all, these results further verify the present implementation and demonstrate that the stabilized DG approach is competitive to the most accurate methods available in the literature. A detailed comparison of the present L^2-conforming discretization to an exactly divergence-free, H^{div}-conforming discretization is shown in Fehn et al. (2019a), revealing that both approaches exhibit very similar discretization properties in the regime of under-resolved turbulent flows. The study by Fehn et al. (2019a) also compares different formulations of the nonlinear convective term. These comparisons are omitted here for brevity. The influence of algebraic sub-grid scale models on dissipation rates for the TGV problem in the context of the present DG discretization is investigated in Neumann (2018).

To investigate the accuracy of high-order discretizations more quantitatively for this turbulent flow problem, Figure 2.23 shows results of spatial convergence tests for various polynomial de-

grees $k = 2, 3, 5, 7, 11, 15$ in terms of error versus number of unknowns, considering the relative error (2.259) of the kinetic energy dissipation rate. Clearly, the discretization operates in the pre-asymptotic regime and rates of convergence appear to depend mainly on the effective spatial resolution rather than on the polynomial degree. However, high-order methods are found to be systematically more accurate than lower order methods even though the convergence behavior does not follow theoretically optimal convergence trends. Furthermore, accuracy seems to saturate for very high polynomial degrees. Overall, these results are encouraging in the sense that an improvement in accuracy can principally be expected for high-order methods also for scenarios where optimal rates of convergence are not observed due to under-resolution. The results shown here complement previous works dealing with DG solvers for compressible flows (Carton de Wiart et al. 2014, Chapelier et al. 2014, Gassner and Beck 2013), and allow to obtain a more systematic and complete picture regarding the accuracy of high-order methods by spanning a wide range of the parameter space in terms of the polynomial degree k and the mesh size h.

Finally, Figure 2.24 shows kinetic energy spectra $E(k)$ at time $t = 9$ around the time of maximum dissipation, where k denotes the wavenumber. The spatial resolution is 64^3 and polynomial degrees of $k = 3, 7, 15$ are considered as in Figure 2.21 for the temporal evolution of the kinetic energy and its dissipation rate. Also shown is the estimated resolution limit $k_{1\%}$ for each polynomial degree according to the 1%-rule by Moura et al. (2017a). The results are further compared to an accurate DNS reference solution corresponding to an effective resolution of 2048^3 for polynomial degree $k = 3$. The results agree very well with the DNS data up to the resolution limit of the discretization. For higher polynomial degrees, an improved resolution capability of the discretization scheme can be observed for the same number of unknowns, and the resolution limit agrees well with the 1%-rule. In the inertial range, the energy spectrum is described more accurately by a $k^{-7/3}$ decay than the usual $k^{-5/3}$ decay according to Kolmogorov, which is in agreement with the results shown in Piatkowski (2019) and might be explained by the fact that the turbulent flow is not fully homogeneous isotropic at that time. The reader is referred to Chapter 7 for a more detailed discussion of energy spectra with a focus on the inviscid limit $\nu = 0$, and to Neumann (2018) for additional investigations on explicit sub-grid scale models.

2.6.7.3 Turbulent channel flow

The turbulent channel flow problem is a widely used benchmark problem to validate LES models for turbulent, wall-bounded flows. In the context of high-order discontinuous Galerkin discretizations for the incompressible Navier–Stokes equations, this test case has been analyzed in Fehn et al. (2018b, 2019a), Krank et al. (2017), Piatkowski (2019), Schroeder (2019).

2.6.7.3.1 Problem description The computational domain Ω_h is a rectangular box with physical dimensions $(L_1, L_2, L_3) = (2\pi\delta, 2\delta, \pi\delta)$ where δ denotes the channel half-width. On the walls located at $x_2 = \pm\delta$, no-slip Dirichlet boundary conditions are prescribed, $\boldsymbol{g}_u = \boldsymbol{0}$, while periodic boundaries are used in the x_1 (streamwise) and x_3 (spanwise) directions. The friction Reynolds number is $\mathrm{Re}_\tau = u_\tau \delta/\nu$, where $u_\tau = \sqrt{\tau_\mathrm{w}/\rho}$ denotes the wall friction velocity defined as a function of the wall shear stress τ_w and the density ρ. The body force vector $\boldsymbol{f}$ driving the flow acts in x_1-direction, $\boldsymbol{f} = (f_1, 0, 0)$. Defining $\rho = 1$, $\delta = 1$, and $f_1 = 1$, a balance of forces in x_1-direction implies $\tau_\mathrm{w} = 1$ and $u_\tau = 1$, so that the viscosity ν is given as $\nu = 1/\mathrm{Re}_\tau$.

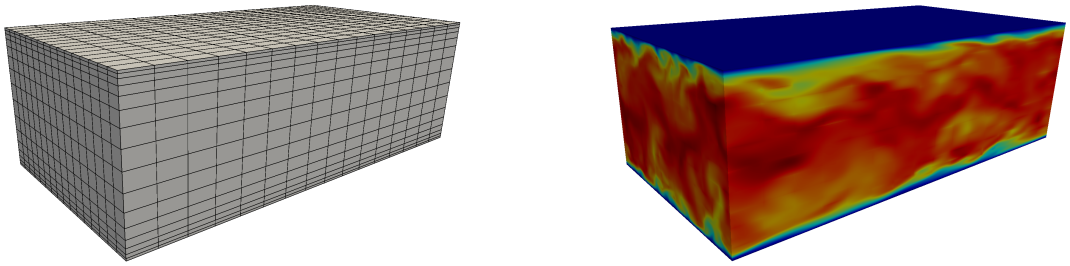


Figure 2.25: Turbulent channel flow problem: geometry/mesh for grid stretch factor $\gamma = 1.8$ and refine level $l = 4$ (left) and velocity magnitude for $\mathrm{Re}_\tau = 180$ (right).

To evaluate the accuracy of the results, the profiles in wall normal direction of the mean velocity $\langle u_1 \rangle$, the root–mean–square values $\mathrm{rms}(u_i) = \left\langle u_i'^2 \right\rangle^{\frac{1}{2}}$, $i = 1, .., d$, and the Reynolds shear stress $\langle u_1' u_2' \rangle$ are considered, where statistical averages are denoted as $\langle \cdot \rangle$ and fluctuations as $(\cdot)' = (\cdot) - \langle \cdot \rangle$. Velocities and components of the Reynolds stress tensor are normalized using the numerically calculated friction velocity u_τ, leading to $u_1^+ = \langle u_1 \rangle / u_\tau$, $(u_i')^+ = \mathrm{rms}(u_i)/u_\tau$, and $(u_1' u_2')^+ = \langle u_1' u_2' \rangle / u_\tau^2$. The dimensionless wall normal coordinate is $x_2^+ = (x_2 + 1)/l^+$ with $l^+ = \nu / u_\tau$. Note that the friction velocity u_τ obtained as a result of a numerical simulation is used to normalize profiles for the results shown in the following, which deviates from the theoretical value $u_\tau = 1$ due to discretization errors. The simulated time interval is $0 \leq t \leq 200 T_0$, where the characteristic time scale $T_0 = L_1 / \|\boldsymbol{u}\|_{\max}$ is defined as the flow-through time based on the mean centerline velocity $\|\boldsymbol{u}\|_{\max} = \langle u_1 \rangle (x_2 = 0)$, which is known a priori from available DNS reference data. The sampling of statistical data is performed over the time interval $100 T_0 \leq t \leq 200 T_0$ and results are sampled every 10^{th} time step. Accordingly, the statistics are sampled over 100 flow-through times based on the mean centerline velocity. It has been verified experimentally that the statistics can be considered as converged and that statistical errors have a negligible effect on the overall accuracy of the results for the chosen parameters.

For an improved resolution of large velocity gradients close to the no-slip boundaries, a mesh stretching in x_2-direction is applied according to the hyperbolic mesh stretching function $f : [0, 1] \to [-\delta, \delta]$ defined in Krank et al. (2017)

$$x_2 \mapsto f(x_2) = \delta \frac{\tanh(C(2x_2 - 1))}{\tanh(C)} \,. \tag{2.260}$$

The parameter C defines the mesh stretching and a value of $C = 1.8$ is used for all turbulent channel flow simulations in the following. Due to the homogeneity of the flow in streamwise and spanwise directions, elements are distributed equidistantly in x_1- and x_3-directions. A high-order isoparametric mapping is used so that each individual element is stretched in x_2-direction. As discussed and observed in Fehn et al. (2019a), this improves the accuracy of results compared to a linear mapping. An illustration of the mesh and the flow field is shown in Figure 2.25. Unless specified otherwise, the BDF2 time integration scheme is used along with adaptive time stepping and a constant Courant number of $\mathrm{Cr} = 0.4$ relatively close to the stability limit. Temporal discretization errors can be expected to be negligible for such a setup. Unless specified otherwise, the dual splitting scheme is used and the default setup of the DG discretization as described in Section 2.6.1. Note that the same numerical setup in terms of grid stretch factor and penalty

Table 2.6: Turbulent channel flow at $\mathrm{Re}_\tau = 180$: analysis of divergence error ε_D and continuity error ε_C (mean values) for different solution strategies. The spatial resolution is $l = 2$ and $k = 3$ corresponding to an effective resolution of 16^3.

	no penalty terms		div penalty term		div + conti penalty	
	ε_D	ε_C	ε_D	ε_C	ε_D	ε_C
coupled solution approach	1.382	0.432	0.062	0.452	0.016	0.014
dual splitting scheme	0.249	0.134	0.028	0.078	0.016	0.013
pressure-correction scheme	1.374	0.430	0.062	0.455	0.015	0.012

factors of the divergence and continuity penalty terms is used for all mesh refinement levels and for all Reynolds numbers in order to assess the predictive capabilities of the solver. Absolute solver tolerances of $\varepsilon_\mathrm{abs} = 10^{-12}$ as well as relative solver tolerances of $\varepsilon_\mathrm{rel} = 10^{-6}$ are used for all linear solvers.

2.6.7.3.2 Stability In a first step, a stability experiment is conducted in order to highlight the importance of the divergence and continuity penalty terms. Instead of considering the kinetic energy, the divergence and continuity error measures defined in Krank et al. (2017)

$$\varepsilon_\mathrm{D}(\boldsymbol{u}_h) = \frac{L \int_{\Omega_h} |\nabla \cdot \boldsymbol{u}_h| \, \mathrm{d}\Omega}{\int_{\Omega_h} \|\boldsymbol{u}_h\| \, \mathrm{d}\Omega} \quad , \quad \varepsilon_\mathrm{C}(\boldsymbol{u}_h) = \frac{\int_{\partial\Omega_h \setminus \Gamma_h} |\, [\boldsymbol{u}_h] \cdot \boldsymbol{n}| \, \mathrm{d}\Gamma}{\int_{\partial\Omega_h \setminus \Gamma_h} |\{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}| \, \mathrm{d}\Gamma} \tag{2.261}$$

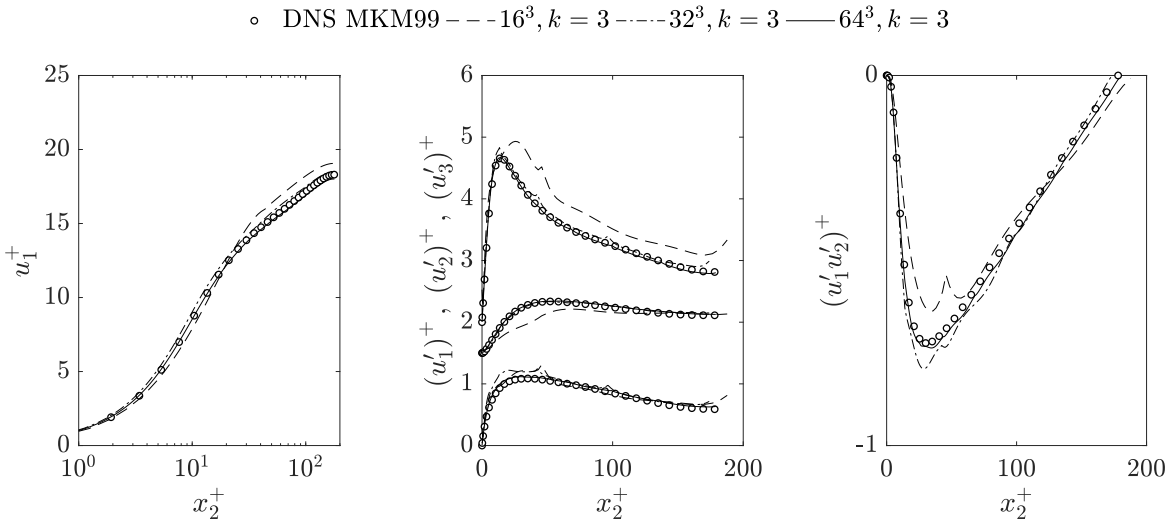
have proven effective in analyzing the stability properties for turbulent flows in the context of high-order discontinuous Galerkin methods. In the above equations, L is a characteristic length scale (given as $L = \delta$ for the turbulent channel flow problem) and $\partial\Omega_h \setminus \Gamma_h$ denotes all interior element faces. Since the relevant effects and instabilities already occur for small Reynolds numbers, a friction Reynolds number of $\mathrm{Re}_\tau = 180$ is selected for this analysis. A very coarse spatial resolution with refinement level $l = 2$ and a polynomial degree of $k = 3$ (resulting in an effective resolution of 16^3) is chosen, since instabilities are expected to arise particularly in the under-resolved regime. Principally, a Courant number of $\mathrm{Cr} = 0.3$ is used for this stability experiment, but smaller Courant numbers ($\mathrm{Cr} = 0.1$) have to be used in case of insufficient discretization schemes to avoid a blow-up due to a violation of the CFL condition.

Time averaged quantities of the divergence and continuity errors are listed in Table 2.6 for different formulations and for different solution strategies such as the coupled solution approach, the dual splitting scheme, and the pressure-correction scheme. The formulation without additional penalty terms serves as a reference, while stabilized formulations with an additional divergence penalty term or both divergence and continuity penalty terms are considerd as well. The reference formulation shows large divergence and continuity errors of order $\mathcal{O}(1)$ indicating that this formulation does not lead to a stable and accurate discretization scheme for turbulent flow problems. Including the divergence penalty term reduces the divergence error significantly, but the continuity error remains high for the coupled solution approach and the pressure-correction scheme. Interestingly, including the divergence penalty term does not only reduce the divergence

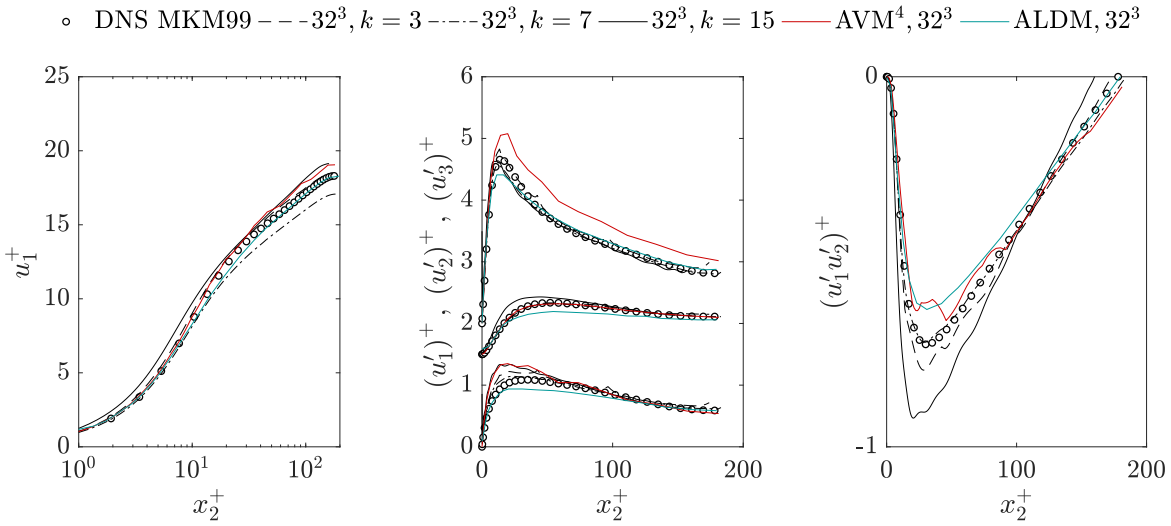
error but also the continuity error in case of the dual splitting scheme. However, to obtain small continuity errors for all solution strategies, the continuity penalty term has to be used as well. Using both penalty terms leads to a robust discretization scheme where the divergence and continuity errors are small for all solution techniques. The results for the dual splitting scheme also explain why the purely divergence penalty based approach has been preferred over the variant including both divergence and continuity penalty terms in Krank et al. (2017), where only the dual splitting scheme has been considered for discretization in time. In general, however, both the divergence penalty term and the continuity penalty term are identified as necessary ingredients to obtain a stable and accurate discontinuous Galerkin discretization for turbulent flow problems. The large errors in Table 2.6 result in large oscillations and jumps in the velocity field that prevent the practical use of the reference formulation without penalty terms as demonstrated in Fehn et al. (2018b), where additional results (omitted here for reasons of brevity) are shown for the statistical quantities of the flow for the different stabilization variants. Hence, both penalty terms are used for all simulation results presented below.

2.6.7.3.3 Results The results shown in the following are compared to accurate DNS reference data from Moser et al. (1999) for $\mathrm{Re}_\tau = 180$ denoted as DNS MKM99, and to reference data from Del Alamo et al. (2004), Hoyas and Jiménez (2008) for $\mathrm{Re}_\tau = 950$ denoted as DNS AJZM04. Moreover, the results are compared to the AVM^4 turbulence model (Rasthofer and Gravemeier 2013) and the ALDM model (Hickel and Adams 2007) in order to evaluate the accuracy of the present approach. The AVM^4 model is a sophisticated turbulence model in the context of variational multiscale methods using a multifractal approach, where the spatial discretization is based on continuous, stabilized, low-order finite element methods. The ALDM model is a state-of-the-art implicit LES approach in the context of finite volume schemes.

Figure 2.26 shows results for a Reynolds number of $\mathrm{Re}_\tau = 180$. An h-convergence test is performed for polynomial degree $k = 3$ by increasing the refinement level from $l = 2$ to $l = 4$ so that the effective resolution increases from 16^3 to 64^3. Intentionally, also a setup with a very coarse spatial resolution is studied in order to explicitly show when and how the results degenerate. The numerical solution tends to the DNS reference solution for increasing spatial resolution. For an effective resolution of 64^3 velocity degrees of freedom, the results agree very well with the DNS data. The results for the $32^3, 64^3$ resolutions appear to be comparable in terms of accuracy as those presented in Ramakrishnan and Collis (2004), where an implicit LES approach is used for DG discretizations of the compressible Navier–Stokes equations and where comparable spatial resolutions are considered for $\mathrm{Re}_\tau = 180$. To investigate the question whether the accuracy of the results obtained for a given number of unknowns can be improved by increasing the polynomial degree of the shape functions, an effective spatial resolution of 32^3 is considered for three different polynomial degrees $k = 3, 7, 15$. Since the flow is under-resolved for such coarse discretizations, it is unclear whether high-order methods are superior in terms of accuracy. Numerical results are shown in Figure 2.26. For all polynomial degrees, the prediction of statistical quantities is acceptable and the results achieve a similar level of accuracy as those for the AVM^4 and ALDM turbulence models. However, taking both mean velocity profiles and the fluctuations into account, no clear advantage in terms of accuracy can be observed for very high-order methods with polynomial degree $k = 7, 15$ as compared to the lower polynomial degree $k = 3$. The results for $k = 15$ appear to be somewhat less accurate, and it was found



(a) h-convergence test for polynomial degree $k = 3$ considering effective resolutions of $16^3, 32^3, 64^3$.

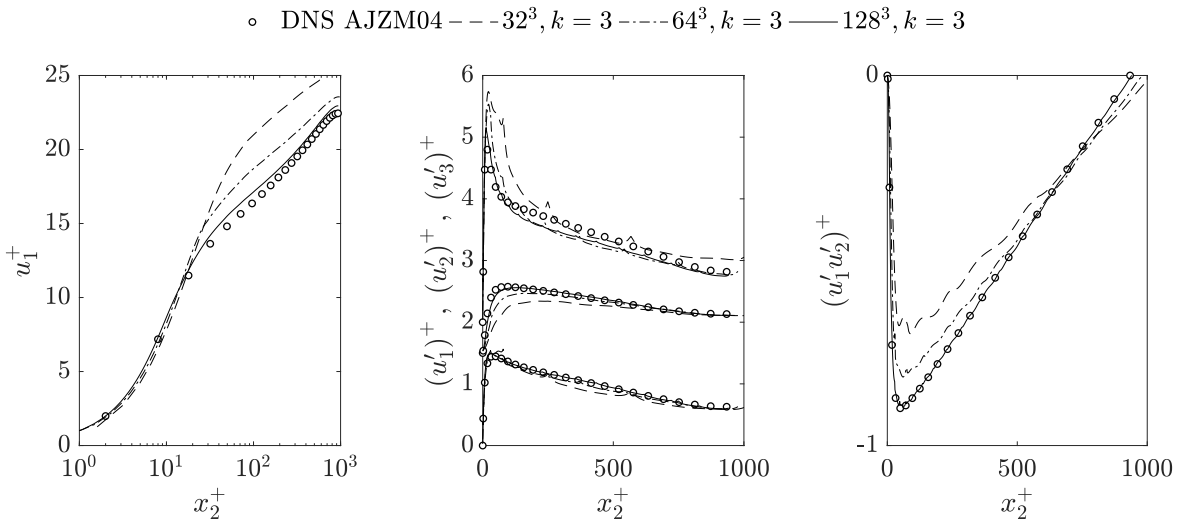


(b) comparison to the AVM^4 and ALDM turbulence models for the same effective resolution of 32^3.

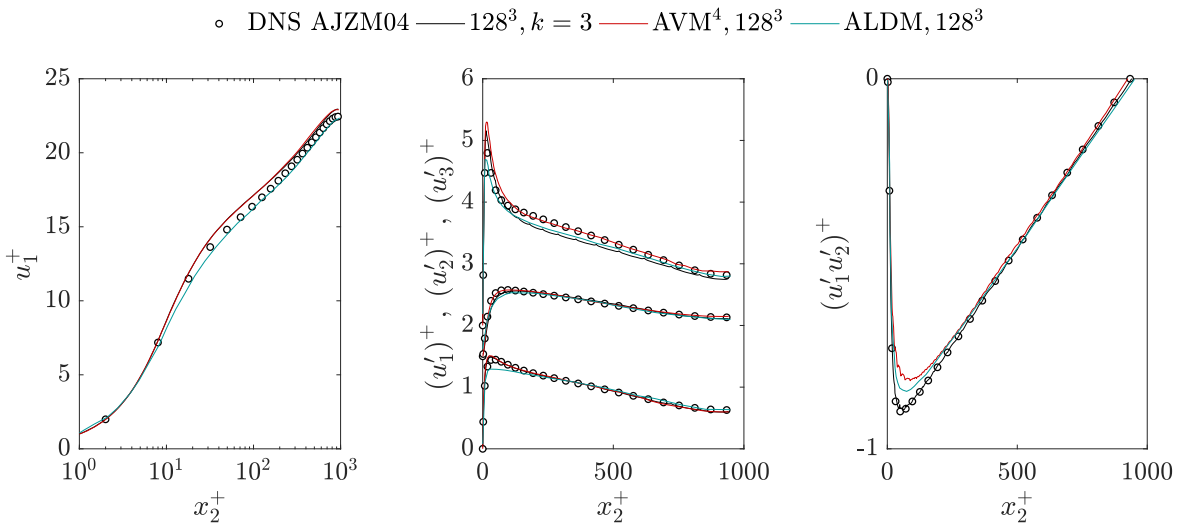
Figure 2.26: Turbulent channel flow at $\mathrm{Re}_\tau = 180$: results obtained for stabilized DG discretization with divergence and continuity penalty terms using the dual splitting scheme.

that apparently more accurate results are obtained for $k = 15$ when using an affine mapping as in Fehn et al. (2018b). Nevertheless, only results for the isoparametric mapping are shown here, given that such a behavior can not be known a priori and that the study by Fehn et al. (2019a) reported generally improved accuracy for isoparametric mappings. Due to the behavior observed for high polynomial degrees for this example, only $k = 3$ will be considered in the following.[3]

[3] A possible explanation for this behavior could be that small flow structures are in fact better resolved for high polynomial degrees, but that this effect is counter-balanced by the increased numerical dissipation of lower



(a) h-convergence test for polynomial degree $k = 3$ considering effective resolutions of $32^3, 64^3, 128^3$.



(b) comparison to the AVM^4 and ALDM turbulence models for the same effective resolution of 128^3.

Figure 2.27: Turbulent channel flow at $\mathrm{Re}_\tau = 950$: results obtained for stabilized DG discretization with divergence and continuity penalty terms using the dual splitting scheme.

Figure 2.27 shows results for the turbulent channel flow problem at $\mathrm{Re}_\tau = 950$. An h-convergence test using polynomial degree $k = 3$ and refinement levels $l = 3$ to $l = 5$ corresponding to effective resolutions of 32^3 to 128^3 is performed. The results are compared to DNS reference data as well as to the AVM^4 and ALDM turbulence models for an effective resolution of 128^3. As for the lower Reynolds number case, results are shown for a very coarse spatial res-

order methods, such that the macroscopic behavior in terms of statistical quantities of the flow is comparable for polynomial degrees ranging from $k = 3$ to $k = 15$.

olution in order to demonstrate the accuracy of the scheme in the highly under-resolved regime. For increasing spatial resolution, the accuracy of the results improves continuously and the results converge towards the DNS reference solution. Note that the results shown here differ from those shown in the original publication (Fehn et al. 2018b) for the coarser spatial resolutions. This is due to a different setup of the discretization scheme. In particular, the continuity penalty term is here applied on both interior and boundary faces for improved robustness (Fehn et al. 2021a), while it has been applied only on interior faces in Fehn et al. (2018b). In this context, a general observation made during extensive numerical studies of this flow problem is that a seemingly perfect prediction of the mean velocity profile for very coarse resolutions might be coincidence. Compared to the AVM^4 and ALDM turbulence models with an effective resolution of 128^3, the present approach achieves a similar level of accuracy or is slightly more accurate than the reference results for AVM^4 and ALDM, e.g., the prediction of the Reynolds shear stress is clearly more accurate for the present discretization scheme. Based on these results, it can be concluded that the high-order discontinuous Galerkin discretization approach with consistent stabilization terms is very promising for large-eddy simulation of incompressible flows and appears to be highly competitive to state-of-the-art LES approaches. The fact that the present approach is a purely numerical approach can be seen as an advantage over physically motivated LES models for which model parameters often have to be calibrated to a specific flow problem. Note also that – unlike the AVM^4 and ALDM models – the present approach does not include a correction term for wall-bounded flows, rendering the present DG discretization a generic numerical approach that aims at obtaining a parameter-free turbulent flow solver.

Apart from the results shown here, a detailed comparison of low-order versus high-order methods for the turbulent channel flow problem has also been performed in Fehn et al. (2019a), with an additional one-to-one comparison of the present L^2-conforming discretization to an exactly divergence-free H^{div}-conforming discretization. The work by Fehn et al. (2019a) has also investigated the impact of affine versus isoparametric element mappings, as well as the influence of discretization parameters on the accuracy and predictive properties of the results by the example of turbulent channel flow at $\text{Re}_\tau = 395$. DNS and LES computations for turbulent channel flow at $\text{Re}_\tau = 180$ and $\text{Re}_\tau = 590$ have been shown in Krank et al. (2017) using an early version of the present DG discretization approach. In the present work, an emphasis is put onto the robustness and accuracy properties of the discretization scheme in the highly under-resolved regime. Results of p-refinement studies on a mesh with a fixed number of elements considering Reynolds numbers of $\text{Re}_\tau = 180$ and $\text{Re}_\tau = 950$ have been shown in Fehn et al. (2019c) using the present incompressible DG solver, along with one-to-one comparisons to a compressible DG solver operating at low Mach number in order to mimic the incompressible case. For reasons of brevity, these results are omitted here. The influence of algebraic sub-grid scale models for the turbulent channel flow problem has been investigated in Dockhorn (2017) in the context of the present DG discretization.

2.6.7.4 Backward facing step

Often, neither an analytical solution nor DNS reference data is available for verification. This section validates the present turbulent flow solver by comparison to experimental data for the well-known backward facing step problem, see Figure 2.28 for a visualization of the geometry and boundary conditions, as well as an instantaneous velocity field. In terms of turbulence, com-

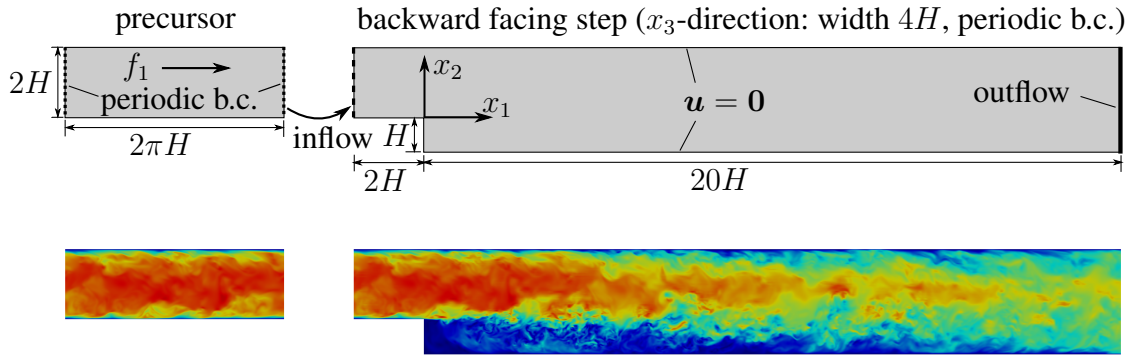


Figure 2.28: Backward facing step: geometry and boundary conditions for precursor simulation strategy as well as a visualization of the flow field.

plexity is increased compared to previous examples since the backward facing step problem is a wall-bounded flow with a free shear layer. A so-called precursor simulation strategy is used to generate turbulent inflow data, meaning that a turbulent channel flow problem is solved on a separate domain and the results of this simulation are mapped to the actual backward facing step domain to prescribe the inflow boundary condition. Both simulations are performed simultaneously solving one time step for the precursor domain and afterwards for the actual domain. The setup chosen here is taken from Rasthofer and Gravemeier (2013) and the reader is referred to this work for detailed information. The friction Reynolds number is $\mathrm{Re}_\tau = u_\tau H/\nu = 290$ ($\hat{=}\mathrm{Re}_H = U_\mathrm{c} H/\nu = 5540$). Using the parameters $H = 0.041$, $\nu = 1.5268 \cdot 10^{-5}$ yields a friction velocity of $u_\tau = \mathrm{Re}_\tau \nu/H = 0.107993$ and a force of $f_1 = \tau_\mathrm{w}/H = 0.2844518$ with $\tau_\mathrm{w} = u_\tau^2$.

The default setup of parameters is used, considering the dual splitting scheme with BDF2 and adaptive time stepping ($\mathrm{Cr} = 0.3$ is used for $k = 2$ and $\mathrm{Cr} = 0.4$ for $k = 5$). The discrete time derivative in the pressure Neumann boundary condition according to equation (2.41) is used. An outflow boundary condition according to Gravemeier et al. (2012) is used in order to ensure stability in case backflow occurs at the outflow boundary. Solver tolerances are $\varepsilon_\mathrm{abs} = 10^{-12}$ and $\varepsilon_\mathrm{rel} = 10^{-3}$.

Results are compared to experimental data from Kasagi and Matsunaga (1995) and denoted as KM95 in the following, as well as to LES simulations from Rasthofer and Gravemeier (2013) with the AVM^4 turbulence model. To allow a fair comparison, spatial resolutions with a comparable number of unknowns are considered. A coarse resolution with approximately 1 million unknowns is studied (the AVM^4 results are obtained on a mesh with $233,472$ elements resulting in $0.93 \cdot 10^6$ DoFs), and a fine resolution with approximately 8 million unknowns (the AVM^4 results are obtained on a mesh with $1,867,776$ elements resulting in $7.5 \cdot 10^6$ DoFs). For the present solver, a coarse mesh with $N_\mathrm{el}(l = 0) = 21$ elements is used. Choosing polynomial degrees of $k = 2$ and $k = 5$, the coarse resolutions with $0.96 \cdot 10^6$ and $1.0 \cdot 10^6$ are obtained for mesh refinement levels of $l = 3$ and $l = 2$, respectively. Increasing the refinement level to $l = 4$ and $l = 3$ for the two polynomial degrees $k = 2$ and $k = 5$ results in the fine resolutions with $7.7 \cdot 10^6$ and $8.3 \cdot 10^6$ unknowns, respectively. In addition, results are shown for a very fine resolution with approximately $61 \cdot 10^6$ and $66 \cdot 10^6$ unknowns for $l = 5, k = 2$ and $l = 4, k = 5$,

respectively. A mesh stretching function similar to Rasthofer and Gravemeier (2013) is used in order to refine the mesh near the wall boundaries and to improve the resolution in the boundary and shear layers, see Dockhorn (2017) for details.

The quality of the results is assessed along vertical lines located at $x_1/H = 0, 1, \ldots, 10$ in terms of the mean streamwise velocity, root-mean-square values for the three velocity components, and the Reynolds shear stress $\langle u_1' u_2' \rangle$, where the mean centerline velocity U_c is used to normalize all profiles. The a-posteriori result U_c of each simulation is used, implying that $\langle u_1(x_1 = -2H, x_2 = H) \rangle / U_\mathrm{c} = 1$ for all runs. The simulated time interval is $0 \leq t \leq 300 T_0$, where the characteristic time scale $T_0 = H/U_\mathrm{c}$ is defined as the flow-through time based on the mean centerline velocity U_c (where U_c is obtained a-priori from the value of Re_H specified above) and the step height H. The precursor starts earlier at time $t = -300 T_0$ in order to obtain a developed turbulent flow in the precursor domain. The sampling of statistical data is performed over the time interval $100 T_0 \leq t \leq 300 T_0$ and results are sampled every 10^{th} time step.

Results are shown in Figure 2.29 for the coarse resolution and in Figure 2.30 for the fine resolution. For the coarse mesh, the mean velocity in streamwise direction is already predicted quite accurately, while larger deviations from the experimental results can be observed for the Reynolds stresses. With increasing spatial resolution, convergence towards the experimental results can clearly be observed. Without going into details, the results for the high polynomial degree $k = 5$ appear to be more accurate than those for the lower degree $k = 2$ overall. When compared to the results for the AVM^4 model from Rasthofer and Gravemeier (2013), a similar level of accuracy can be observed for a similar number of unknowns. The results for AVM^4 are often between those for $k = 2$ and those for $k = 5$. Taking the Reynolds shear stress as an example, the results for $k = 2$ are comparable to AVM^4 on the coarse mesh, and the results for $k = 5$ are comparable to AVM^4 on the fine mesh. To evaluate the accuracy of the different LES computations, additional results are shown for the very fine resolution in Figure 2.31. The results for $k = 2$ and $k = 5$ are close to each other for this resolution, indicating that the solution becomes more and more resolved. The agreement with experimental results is very good and deviations can mainly be observed in regions where the experimental results show a large scatter. All in all, the present implicit DG solver gives results in agreement with experimental data and consistent with state-of-the-art LES approaches. As a sidenote, the predictive capabilities of the present implicit LES approach become evident when realizing that classical LES sub-grid scale models may deteriorate the results significantly for this problem, see Dockhorn (2017), Robalo Rei (2017). The close agreement with the AVM^4 results for comparable problem size is remarkable given that the underlying numerical methods differ significantly and the meshes are also different.

One conclusion that one might draw from these results is that the impact of turbulence modeling with respect to accuracy is clearly limited for a given spatial resolution, but that the most reliably way to improve the accuracy in LES is instead increasing the spatial resolution (which is the big promise of LES compared to RANS). Against this background, a successful LES approach is one that – apart from good robustness and accuracy properties – is computationally efficient, which bridges the gap to upcoming chapters.

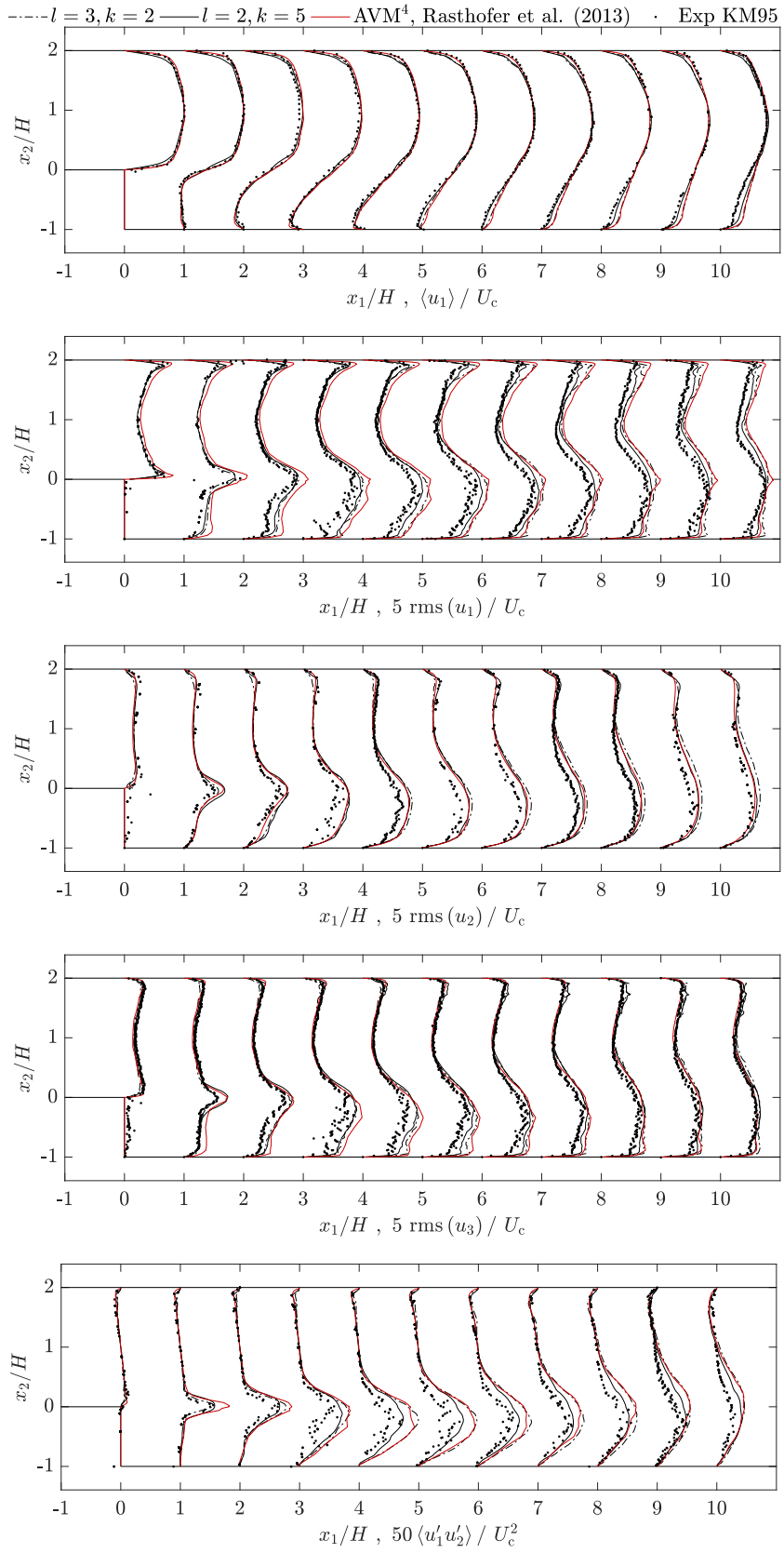


Figure 2.29: Backward facing step: results obtained for *coarse* resolution.

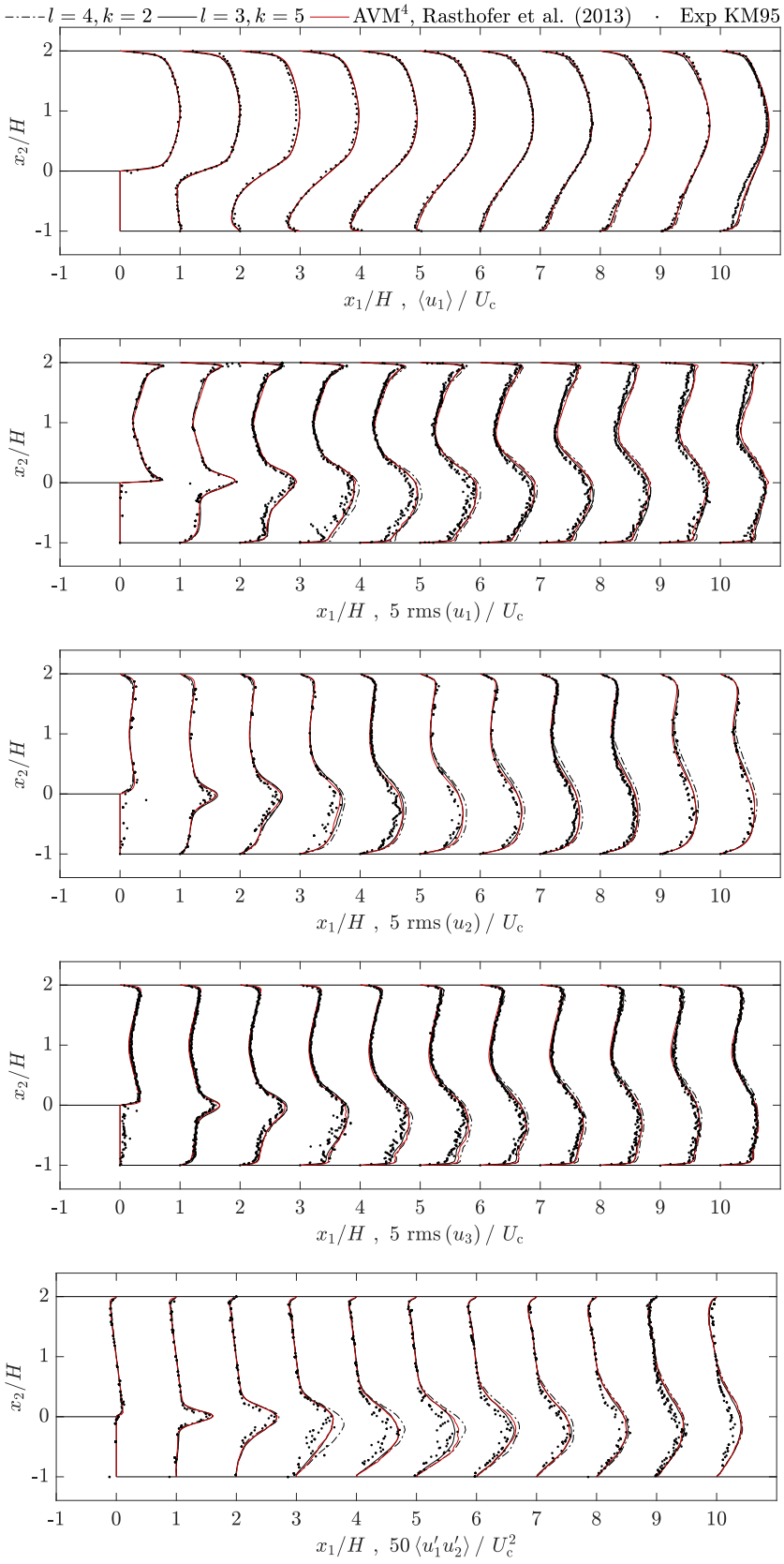


Figure 2.30: Backward facing step: results obtained for *fine* resolution.

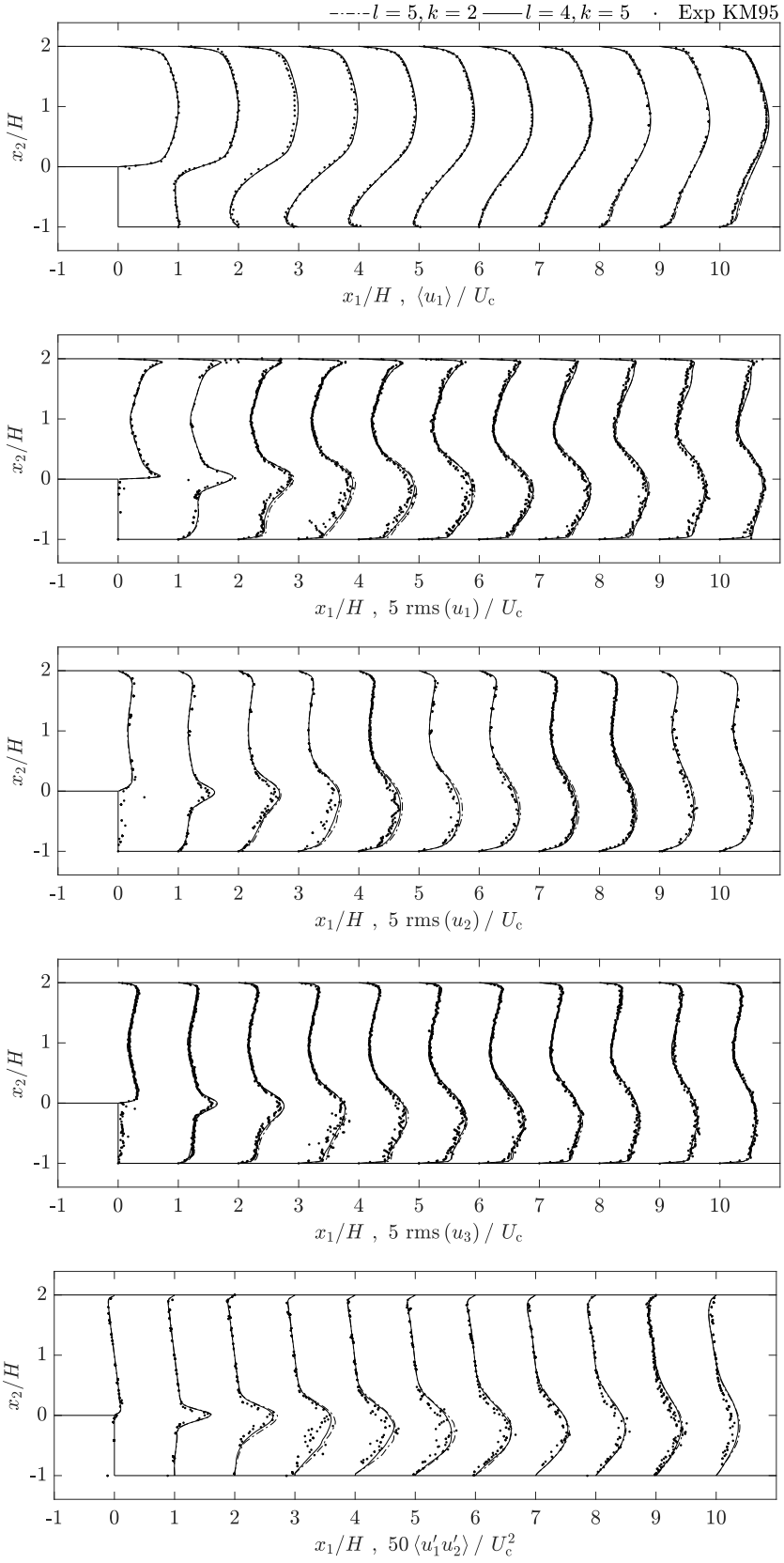


Figure 2.31: Backward facing step: results obtained for *very fine* resolution.

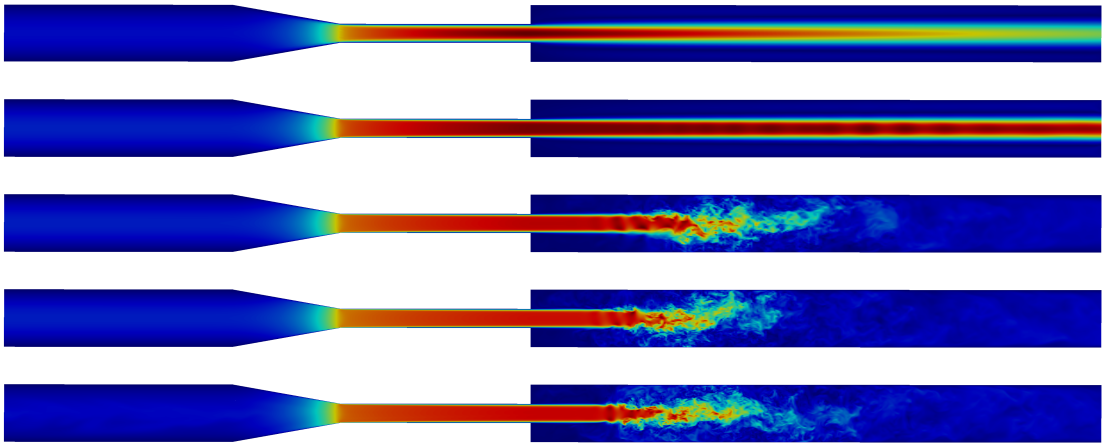


Figure 2.32: Visualization of numerical results for FDA benchmark nozzle model for Reynolds numbers of $\mathrm{Re}_{\mathrm{th}} = 500, 2000, 3500, 5000, 6500$ (from top to bottom), see Fehn et al. (2019b).

2.6.7.5 More turbulent flow examples

Apart from the results shown here, the present DG discretization approach has been extensively validated for the FDA benchmark nozzle model in Fehn et al. (2019b). This benchmark problem has been designed to obtain a benchmark setup in computational fluid dynamics representative of flow configurations in biomedical devices. An illustration of the flow problem is shown in Figure 2.32 for Reynolds numbers of $\mathrm{Re}_{\mathrm{th}} = 500, 2000, 3500, 5000, 6500$, covering the laminar, transitional, and turbulent regimes characteristic of flows in biomedical engineering. In the study by Fehn et al. (2019b), a precursor simulation approach has been suggested as for the backward facing step problem shown above. While the original goal of this study has been the validation of the DG discretization scheme, it turned out that the FDA benchmark problem complicates the investigation of the discretization properties of a numerical method due to a large sensitivity of results (in particular the location of the jet breakdown for transitional and turbulent cases) with respect to certain parameters. The reader is referred to Fehn et al. (2019b) for an in-depth discussion of these aspects.

Moreover, the present DG discretization framework has also been validated by the example of the well-known periodic hill flow problem in Krank et al. (2018a), Kronbichler et al. (2018b) in terms of DNS and LES computations. Extensions to wall-modeling as well as RANS and hybrid RANS/LES modeling have been developed by Krank et al. (2018b, 2019a,b).

2.7 Conclusion and outlook

This chapter has discussed modern discretization methods for the incompressible Navier–Stokes equations. With respect to discretization in time, different solution approaches have been discussed with an emphasis on projection-type methods as computationally efficient alternatives to a coupled solution approach. Principally, projection methods are state-of-the-art techniques and

attention is paid here to the imposition of boundary conditions and other pecularities in the context of DG discretizations. The main emphasis has been on obtaining a robust DG discretization approach as a generic and accurate flow solver for laminar, transitional, and turbulent flow problems. The problem of instabilities observed for small time step sizes is solved by a consistent discretization of the velocity–pressure coupling terms based on integration-by-parts with consistent numerical fluxes and boundary conditions. It has been shown that mixed-order polynomials should be used in order to ensure inf–sup stability, also for projection-type methods that might inherently contain inf–sup stabilizing terms.

The main novelity of this chapter is a stabilization approach for L^2-conforming discretizations based on divergence and continuity penalty terms in order to ensure robustness and accuracy in under-resolved scenarios. This approach has been motivated theoretically as a technique to weakly enforce mass conversation and energy stability, and has been demonstrated to perform very well in practice by studying a sequence of increasingly complex benchmark problems. Optimal convergence behavior has been demonstrated for problems with smooth solutions, and the accuracy of high-order methods has been assessed critically when applied to more realistic flow problems which either exhibit geometric complexities or operate in the under-resolved regime. The examples considered here are still academic, but might nevertheless be informative in terms of which level of accuracy can be expected from high-order discretizations (at best) when applied in an industrial context. On the one hand, it might be disappointing that optimal convergence behavior of high-order methods is lost for these turbulent flow problems. On the other hand, the present results are nevertheless encouraging given that the DG discretization proposed here is highly competitive to the most accurate LES approaches currently available and, essentially, does not contain any turbulence model parameters. This makes one confident that it might be possible to address the long-standing problem of LES modeling by robust and accurate discretization schemes, with the ultimate goal to remove the burden of physical sub-grid modeling. This conclusion might be tentative in light of the fact that the examples studied here are moderate-Reynolds-number flows. In this context, however, it should be noted that a new class of problems is opened in this thesis by the present DG discretization approach, namely the goal to obtain grid-converged results for the challenging inviscid Taylor–Green problem with infinite Reynolds number studied in detail in Chapter 7.

Taking the DG formulation from Hesthaven and Warburton (2007) as a reference (from which this PhD project has originally started), the state-of-the-art could be improved significantly, which can be seen as a necessary step towards preparing this type of methods for industrial LES. In retrospect, parts of the DG community have been over-optimistic in expecting that an upwind flux will do the job in case of the incompressible Navier–Stokes equations and convection-dominated flows without further stabilization techniques in an L^2-conforming setting. The present thesis identified severe robustness issues for state-of-the-art approaches and could significantly contribute to the question of which ingredients are actually necessary to obtain desired robustness properties in the incompressible case. Comparisons to H^{div}-conforming methods (Fehn et al. 2019a) underlined these conclusions, highlighting the importance of mass conservation and energy stability in addition to well-known techniques such as upwinding for convective terms and consistent integration of nonlinear terms. An advantage of L^2-conforming discretizations over H^1-conforming discretizations might be that convection-stabilization appears to be less of a concern and that stabilized formulations for improved mass conservation do

not suffer from over-stabilization. A lack of pressure-robustness for L^2-conforming discretizations is a disadvantage compared to exactly divergence-free H^{div}-conforming discretizations.

The L^2-conforming approach is straightforward to implement since it only requires "standard" finite element ingredients. It is attractive since the inversion of the mass matrix is very cheap and since fast preconditioning techniques are available. In favor of these attractive properties, a compromise made is that pressure-robustness is formally given up compared to exactly divergence-free H^{div}-conforming methods. In the context of method development, an extension of the present incompressible Navier–Stokes solvers in the `ExaDG` software project (built upon the `deal.II` finite element library) towards H^{div}-conforming elements could therefore be a very interesting direction for the future. This would enable a comparison of both approaches within the same implementation framework and allow to address the question of overall computational efficiency. Application to industrial flows would be one of the next logical step. This naturally calls for meshes composed of simplicial elements, the support of which is currently being initiated in the `deal.II` library. Finally, the implementation should be extended towards enabling adaptively refined hexahedral meshes w.r.t. matrix-free implementations and multigrid solvers in a high-order DG context.

3 Extension to natural convection flows

This chapter presents an extension of the incompresible flow solvers from Chapter 2 to coupled flow–transport problems. In addition to the incompressible Navier–Stokes equations, a transport equation is solved for a scalar quantity, where the transport velocity is the solution of the incompressible flow problem. This leads to active or passive scalar transport, depending on whether the additional transport equation couples back to the incompressible Navier–Stokes equations or not. An important representative of transport problems with an active scalar are buoyancy-driven flows based on the Boussinesq approximation, termed natural convection flows in the following. In the presence of gravitational forces, temperature differences imply differences in the density, and the related buoyancy force enters the momentum equation as a source term. Variations of the density are neglected in all other terms of the equations. The set of governing equations is

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}) - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) + \nabla p = \left(1 - \beta\left(\theta - \theta_{\mathrm{ref}}\right)\right) \boldsymbol{g} \text{ in } \Omega \times [0,T] \ , \tag{3.1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \text{ in } \Omega \times [0,T] \ , \tag{3.2}$$

$$\frac{\partial \theta}{\partial t} + \nabla \cdot (\boldsymbol{u}\, \theta) - a \nabla^2 \theta = 0 \text{ in } \Omega \times [0,T] \ , \tag{3.3}$$

where β is the thermal expansion coefficient, θ the temperature, θ_{ref} a reference temperature, $\boldsymbol{g}$ the gravitational force, and a the thermal diffusivity. The remaining quantities have been described in Section 2.2 for the incompressible Navier–Stokes equations. Note that this class of problems then also contains the more simple case of passive scalar transport as a special case. Exploiting $\nabla \cdot \boldsymbol{u} = 0$, the transport equation can be alternatively written in convective form

$$\frac{\partial \theta}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \theta - a \nabla^2 \theta = 0 \text{ in } \Omega \times [0,T] \ . \tag{3.4}$$

The following non-dimensional quantities can be introduced

$$\mathrm{Pr} = \frac{\nu}{a} \ , \ \mathrm{Gr} = \frac{g \beta \left(\theta - \theta_{\mathrm{ref}}\right) L_{\mathrm{ref}}^3}{\nu^2} \ , \ \mathrm{Ra} = \mathrm{GrPr} \ . \tag{3.5}$$

where L_{ref} is a characteristic length scale, Pr the Prandtl number, Gr the Grashof number, and Ra the Rayleigh number. The Prandtl number describes the ratio of molecular transfer of momentum to molecular transfer of heat, the Grashof number the ratio of buoyancy forces to viscosity, and the Rayleigh number the ratio of buoyancy forces to combined heat and momentum diffusivities. Defining the characteristic velocity and the corresponding Reynolds number as

$$U_{\mathrm{ref}} = \sqrt{g \beta \left(\theta - \theta_{\mathrm{ref}}\right) L_{\mathrm{ref}}} \ , \ \mathrm{Re} = \frac{U_{\mathrm{ref}} L_{\mathrm{ref}}}{\nu} \ , \tag{3.6}$$

the Grashof number and Rayleigh number can be alternatively written as $\mathrm{Gr} = \mathrm{Re}^2$ and $\mathrm{Ra} = \mathrm{Re}^2\mathrm{Pr}$, respectively.

This chapter proposes a novel L^2-conforming DG discretization for natural convection flows using a weakly-coupled solution strategy without sub-iterations in order to efficiently solve the coupled flow–transport problem. A key ingredient towards robustness in the L^2-conforming setting with weakly divergence-free velocity is to formulate the transport term in the temperature equation in convective form instead of the conservative form. Numerical results investigate and demonstrate the robustness of the proposed formulation.

3.1 Motivation

It is well-known that natural convection flows pose special requirements in terms of mass conservation in order to obtain stable and accurate solutions and to avoid spurios velocities, see for example Dawson et al. (2004), Dorok et al. (1994), Galvin et al. (2012), Gerbeau et al. (1997), Pelletier et al. (1989), Waluga et al. (2016). Two aspects appear to be important in terms of the accuracy and stability of discretization schemes for natural convection problems:

- With respect to the incompressible flow problem, the topic is also closely related to the aspect of pressure-robustness as discussed in Section 2.4.4. In case of natural convection flows with large irrotational forces, large velocity errors (spurious velocities) might occur in particular if the viscosity becomes small.

- With respect to the coupled flow–transport problem, mass-conservation errors of the discrete velocity solution might adversely impact the temperature solution. Since the temperature enters the incompressible Navier–Stokes equations via the body force term according to the Boussinesq approximation, this feedback might even lead to instabilities.

Discretization schemes such as H^{div}-conforming formulations along with a velocity space whose divergence is in the pressure space yield a velocity that is normal-continuous and exactly divergence-free, see also Section 2.4.3. Such schemes are pressure-robust (see equation (2.168)), and the discrete velocity is exactly mass-conserving, so that continuity errors can not be amplified by the temperature equation for coupled flow–transport problems (see Section 3.4 below). The discretization scheme proposed in Chapter 2 for the incompressible Navier–Stokes equations is not strictly pressure-robust and is based on penalty terms for improved mass conservation. It is therefore interesting to analyze how stable and accurate discretization schemes can be devised for the class of natural convection problems. The analysis in Dawson et al. (2004) suggests that the discretization scheme for the incompressible Navier–Stokes equations does not have to be exactly mass-conserving, but that a compatibility condition should be fulfilled between the discretization schemes for the flow and transport equations.

3.1.1 State-of-the-art

Table 3.1 summarizes previous contributions in the field of discontinuous Galerkin methods for natural convection problems. The formulations are categorized with respect to the function spaces for velocity and pressure, the aspect of stabilization techniques for the incompressible

Table 3.1: Publications from the literature addressing the solution of natural convection problems by discontinuous Galerkin discretizations. Legend: The symbol ✓means that this option is considered in a given study.

Study	Function space	Stabilization	$\nabla \cdot (\boldsymbol{u}\, \theta)$	$(\boldsymbol{u} \cdot \nabla)\theta$
Bassi et al. (2007), Franciolini et al. (2017)	L^2 (equal-order)	AC inviscid flux	✓	
Schroeder and Lube (2017)	L^2 (mixed-order)	$\nabla \cdot \boldsymbol{u}_h, [p_h]$		✓
	L^2 (equal-order)	$\nabla \cdot \boldsymbol{u}_h, [p_h]$		✓
	H^{div} (mixed-order)	$\nabla \cdot \boldsymbol{u}_h = 0, [\boldsymbol{u}_h] \cdot \boldsymbol{n} = 0$		✓
Piatkowski (2019)	L^2 (mixed-order)	H^{div}-reconstr.	✓	
Busto et al. (2020)	L^2 (equal-order)	staggered DG	✓	
Present work	L^2 (mixed-order)	$\nabla \cdot \boldsymbol{u}_h, [\boldsymbol{u}_h] \cdot \boldsymbol{n}$	✓	✓

flow problem as well as exact fulfillment of the divergence-free constraint, and the type of formulation (convective, conservative) chosen for the transport term in the temperature equation. While various formulations have been presented in the literature with application to natural convection problems, the following discussion reveals that there appears to be no clear understanding of which ingredients are necessary to ensure a robust simulation of this type of problems in an L^2-conforming setting:

The method proposed in Bassi et al. (2007) and also used in Franciolini et al. (2017) is an L^2-conforming DG discretization where the computation of the inviscid fluxes is based on an artificial compressibility approach that acts as a stabilization allowing the use of equal-order polynomials (inf–sup stability) according to Bassi et al. (2007). It is unclear whether this stabilization also helps regarding the aspects of mass conservation and energy stability highly relevant for L^2-conforming discretizations as discussed in Chapter 2. The convective term in the temperature equation is used in conservative formulation.

Schroeder and Lube (2017) consider grad-div stabilization for both equal-order and mixed-order polynomials. Additionally, a pressure-jump penalization is considered in both cases, even though this stabilization is only necessary for inf–sup stability in case of an equal-order formulation (Cockburn et al. 2009). The used penalty terms have not been designed with consistent physical units in mind, and heavy grad-div stabilization is considered with penalty factors varying from order unity up to 10^9, which might pose difficulties when using iterative solvers. Additionally, an exactly divergence-free H^{div}-conforming discretization is also considered, for which additional stabilization would be superfluous. The transport term in the temperature equation is formulated in convective form without explanation.

Piatkowski (2019) uses an L^2-conforming DG discretization with mixed-order polynomials and an H^{div}-reconstruction as a stabilization technique in the projection step of the pressure-correction scheme for the incompressible Navier–Stokes equations. The resulting velocity field

is pointwise divergence-free but not normal-continuous between elements. The convective term in the temperature equation is written in conservative form.

Busto et al. (2020) present a staggered DG approach for natural convection problems using equal-order polynomials for velocity and pressure. The convective term of the temperature equation is written in conservative formulation and no additional stabilization techniques are used for the incompressible flow problem. However, the authors mention that artificial viscosity has been used to avoid instabilities. Results are shown mainly for fine spatial resolutions, so that energy stability and robustness for highly under-resolved scenarios is not entirely clear for this approach. Note that the present work only refers to the Eulerian approach for the incompressible model under the Boussinesq assumption presented in Busto et al. (2020), where also a semi-Lagrangian approach and a fully compressible model are presented that are not discussed here.

3.1.2 Novel contributions of the present work

Numerical results in Chapter 2 provide evidence that the use of both divergence and normal-continuity penalty terms are necessary ingredients to obtain a robust and accurate flow solver, see also Akbas et al. (2018), Fehn et al. (2018b). Mixed-order polynomials are used to ensure inf–sup stability, and no pressure stabilization is therefore considered in the present work. An interesting question to be investigated is whether the design of the penalty parameters by means of dimensional analysis ensuring consistent physical units is not only appropriate for problems of forced convection, but also natural convection without the need to readjust the formulation or the penalty parameters. This chapter investigates in detail the formulation of the transport term in the temperature equation, with the main conclusion that the convective formulation is necessary for robustness in case of a velocity approximation that is not exactly mass-conserving. The motivation behind is as follows (see also Chippada et al. (1998)): The transport equation can be reformulated as

$$\frac{\partial \theta}{\partial t} + \theta \nabla \cdot \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\, \theta - a \nabla^2 \theta = 0 \,. \tag{3.7}$$

In the continuous case it holds $\nabla \cdot \boldsymbol{u} = 0$. Then, the temperature equation contains only derivatives of the temperature in either space or time, and also the Boussinesq term in the momentum equation contains only the temperature difference to a reference value. Hence, the overall temperature level θ_{ref} can be chosen arbitrarily without affecting the solution of the problem. In the discrete case, however, the divergence-free constraint is only fulfilled approximately for the present stabilized DG discretization, $\nabla \cdot \boldsymbol{u}_h \approx 0$. In this case, it can be expected that the temperature solution depends on the temperature level, since divergence errors of the velocity field are scaled by the temperature, i.e., this term increases for larger absolute values of the temperature. Since the temperature equation couples back to the incompressible Navier–Stokes equations, this effect might even lead to instabilities. Hence, the present work proposes to skip this potentially problematic term in the temperature equation and to use the convective form of the transport term in the temperature equation instead of the conservative form. The above discussion is of course simplified since it is based on the continuous model, and the reader is referred to Section 3.4 for a more thorough discussion considering the discrete-in-space problem, for which the reasoning is slightly more complicated. Nevertheless, theoretical and numerical investigations shown in this chapter justify the above conclusion.

During the development of the discretization methods presented in this chapter, it has been observed that some formulations produce stable results when choosing the parameters and spatial resolutions shown in the literature, but that the same formulations lacked robustness when changing parameters such as the absolute temperature level towards higher values (which should not affect results for a robust numerical method), or the spatial resolution towards more strongly under-resolved scenarios. While these observations might point to deficiencies of the present discretization schemes for certain formulations, these results might also shed light on previous studies in the sense that it appears to be unclear whether these formulations are really robust discretization schemes. Concretely, the formulations in Bassi et al. (2007), Busto et al. (2020), Franciolini et al. (2017), Piatkowski (2019) might not be robust because the conservative formulation $\nabla \cdot (\boldsymbol{u}\, \theta)$ is used for the transport equation along with a velocity field that is not exactly mass-conserving, i.e., these works do not argue that their discretization schemes fulfill a compatibility condition for the coupled flow–transport problem according to Dawson et al. (2004). Furthermore, energy stability of these schemes is unclear (see also the discussion in Chapter 2). The L^2-conforming formulation in Schroeder and Lube (2017) might not be robust because no normal-continuity penalty term is used (see also Akbas et al. (2018)), and because the penalty factor is chosen as a constant factor without paying attention to physically consistent units (see also Fehn et al. (2018b)). According to the theoretical consideration and numerical results shown in this chapter, the use of the convective formulation for the transport term can be expected to yield a robust solver for the flow–transport coupling. In summary, the goal of this chapter is to provide insights into which formulations are required to obtain a robust discretization scheme. According to the author's opinion, a necessary ingredient to achieve this goal is to explicitly show results for formulations that lack robustness. For this reason, the present work focuses on robustness tests with a qualitative assessment of results, while a detailed quantitative assessment of results is omitted given that this is not expected to provide further insights compared to the state-of-the-art.

Principally, the constant gravitational force $\boldsymbol{g}$ in the momentum equation (3.1) could also be skipped since this term only adds a hydrostatic component to the pressure solution in the continuous case, and does not affect the velocity solution for pressure-robust discretizations in the discrete case. Some works using non pressure-robust discretizations skip this term and solve only for the dynamic pressure variations. From the point of view of pressure-robustness, this might result in an easier setup that might prevent numerical inaccuracies or even instabilities. Since the present discretization scheme is not strictly pressure-robust and since the aim of this chapter is to identify instabilities, the hydrostatic part is kept in the equations in order to test the scheme in a potentially more challenging setup. For practical applications, it might nevertheless be reasonable to skip this term for improved accuracy due to a smaller pressure norm.

Another motivation for developing DG-based discretization schemes as shown in this chapter is to avoid the need for stabilizing the temperature equation, which is typically necessary when discretizing this equation with continuous finite element methods and when considering convection-dominated problems, see for example the work by Kronbichler et al. (2012) dealing with problems of earth mantle convection at high Rayleigh numbers and using an entropy viscosity method for stabilization of the temperature equation.

3.2 Temporal discretization of scalar transport equation

A weakly-coupled solution strategy without sub-iterations between the incompressible Navier–Stokes problem and the scalar transport problem is used here for reasons of computational efficiency. A method-of-lines approach is chosen with time integration based on BDF and extrapolation schemes similar to the time integration strategy used in Kronbichler et al. (2012) to solve natural convection problems under the Boussinesq approximation. In the current implementation, the three time integration strategies presented in Section 2.3 can be used to solve the incompressible Navier–Stokes problem in case of coupled flow–transport problems. For brevity, equations are shown only for the coupled solution approach in the following. To resolve the velocity–temperature coupling, a high-order extrapolation of the temperature is used in the Boussinesq term on the right-hand side of the momentum equation. Then, if convective terms are treated implicitly, the following problem needs to be solved (where $\boldsymbol{u}^{n+1}$ in the temperature equation is already available from the incompressible flow problem)

$$\begin{aligned}
\frac{\gamma_0^n \boldsymbol{u}^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n} + \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}^{n+1}) - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{n+1}) + \nabla p^{n+1} = \\
\left(1 - \beta \left(\sum_{i=0}^{J-1} \beta_i^n \theta^{n-i} - \theta_{\mathrm{ref}}\right)\right) \boldsymbol{g} \,, \\
\nabla \cdot \boldsymbol{u}^{n+1} = 0 \,, \\
\frac{\gamma_0^n \theta^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \theta^{n-i}}{\Delta t_n} + \left(\boldsymbol{u}^{n+1} \cdot \nabla\right) \theta^{n+1} - a \nabla^2 \theta^{n+1} = 0 \,,
\end{aligned} \tag{3.8}$$

using the notation introduced in Section 2.3. If convective terms are treated explicitly, the following problem needs to be solved

$$\begin{aligned}
\frac{\gamma_0^n \boldsymbol{u}^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n} + \sum_{i=0}^{J-1} \beta_i^n \nabla \cdot \boldsymbol{F}_{\mathrm{c}}(\boldsymbol{u}^{n-i}) - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{n+1}) + \nabla p^{n+1} = \\
\left(1 - \beta \left(\sum_{i=0}^{J-1} \beta_i^n \theta^{n-i} - \theta_{\mathrm{ref}}\right)\right) \boldsymbol{g} \,, \\
\nabla \cdot \boldsymbol{u}^{n+1} = 0 \,, \\
\frac{\gamma_0^n \theta^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \theta^{n-i}}{\Delta t_n} + \sum_{i=0}^{J-1} \beta_i^n \left(\boldsymbol{u}^{n-i} \cdot \nabla\right) \theta^{n-i} - a \nabla^2 \theta^{n+1} = 0 \,.
\end{aligned} \tag{3.9}$$

If the convective term is formulated in conservative formulation in the transport equation, the formulation $\nabla \cdot (\boldsymbol{u}^{n+1}\theta^{n+1})$ is used in the implicit case and $\sum_{i=0}^{J-1} \beta_i^n \nabla \cdot (\boldsymbol{u}^{n-i}\theta^{n-i})$ in the explicit case. Note that the choice of an explicit vs. implicit formulation of convective terms can be made independently for the incompressible flow solver and the scalar transport solver in the present implementation. In case of an explicit treatment of the convective term in both the incompressible flow problem and the transport equation with a time step restriction according to the CFL condition, the minimum time step size of the two sub-problems is used as the global time step size in order to advance both solvers synchronously in time. In case of adaptive time

stepping, the two solvers of the partitioned algorithm need to communicate the new time step size after each time step, in addition to the exchange of solution vectors between the weakly-coupled solvers. As for the incompressible flow solvers, the diffusive term in the transport equation is formulated implicitly to avoid the more severe time step restriction related to differential operators with second derivatives.

An implicit formulation of the convective term in the temperature equation might be particularly attractive from the point of view of computational costs for applications dealing with creeping flow problems such as earth mantle convection. This type of problem is typically modeled by solving only the steady Stokes equations instead of the full incompressible Navier–Stokes equations. Since the temperature equation is much cheaper to solve than the Stokes problem, overcoming explicit time step restrictions stemming from the temperature equation often proves more efficient than for an isolated incompressible Navier–Stokes problem (the number of time steps is reduced and solving the Stokes problem typically does not become more expensive for larger time step sizes).

3.3 Discontinuous Galerkin discretization of scalar transport equation

This section briefly describes the DG discretization of the temperature equation for the conservative and convective formulations of the transport term. The reader is referred to Section 2.4 for an introduction of the notation used here as well as the DG discretization of the incompressible flow problem. The finite element space chosen for the temperature equation is

$$\mathcal{V}_h^\theta = \left\{ \theta_h \in L^2(\Omega_h) \ : \ \theta_h \left(\boldsymbol{x}^e(\boldsymbol{\xi}) \right) |_{\Omega_e} = \tilde{\theta}_h^e(\boldsymbol{\xi})|_{\tilde{\Omega}_e} \in \mathcal{V}_{h,e}^\theta = \mathcal{Q}_{k_\theta}(\tilde{\Omega}_e) \ \forall e \right\} \ , \tag{3.10}$$

where k_θ is the polynomial degree of the shape functions for the scalar variable θ that may be chosen independently of k_u, k_p. Unless specified otherwise, $k_\theta = k_u$ is chosen here. The weak DG formulation of equation (3.3) or equation (3.4) reads: Find $\theta_h \in \mathcal{V}_h^\theta$ such that

$$m_{h,\theta}^e \left(\iota_h, \theta_h \right) + c_{h,\theta}^e \left(\iota_h, \theta_h, \boldsymbol{u}_h \right) + a \, l_h^e \left(\iota_h, \theta_h \right) = 0 \ , \tag{3.11}$$

for all $\iota_h \in \mathcal{V}_{h,e}^\theta$ and for all elements $e = 1, ..., N_{\text{el}}$, where $m_{h,\theta}^e$ denotes the scalar mass matrix operator and where l_h^e is the SIPG discretization of the scalar Laplace operator discussed in Section 2.4.2.3. The DG discretization of the convective term in conservative form, equation (3.3), is given as

$$c_{h,\theta}^e \left(\iota_h, \theta_h, \boldsymbol{u}_h \right) = - \left(\nabla \iota_h, \boldsymbol{u}_h \theta_h \right)_{\Omega_e} + \left(\iota_h, \left(\boldsymbol{u}_h \theta_h \right)^* \cdot \boldsymbol{n} \right)_{\partial \Omega_e} \ , \tag{3.12}$$

using the following local Lax–Friedrichs flux

$$\left(\boldsymbol{u}_h \theta_h \right)^* = \{\{ \boldsymbol{u}_h \theta_h \}\} + \frac{\lambda_\theta}{2} [\![\theta_h]\!] \ , \tag{3.13}$$

where $\lambda_\theta = \max \left(|\boldsymbol{u}_h^- \cdot \boldsymbol{n}|, |\boldsymbol{u}_h^+ \cdot \boldsymbol{n}| \right)$. For the alternative convective formulation, the DG discretization reads

$$c_{h,\theta}^e \left(\iota_h, \theta_h, \boldsymbol{u}_h \right) = + \left(\iota_h, \left(\boldsymbol{u}_h \cdot \nabla \right) \theta_h \right)_{\Omega_e} + \left(\iota_h, \{\{ \boldsymbol{u}_h \}\} \cdot \boldsymbol{n} \left(\theta_h^* - \theta_h \right) \right)_{\partial \Omega_e} \ , \tag{3.14}$$

using the following upwind flux

$$\theta_h^* = \{\{\theta_h\}\} + \frac{1}{2}\text{sign}\left(\{\{\boldsymbol{u}_h\}\}\cdot\boldsymbol{n}\right)[\theta_h] \ . \tag{3.15}$$

For the temperature field, Dirichlet and Neumann boundary conditions can be prescribed where the mirror principle is used for the weak imposition of boundary conditions as has been described in Section 2.4 for the incompressible Navier–Stokes equations. Since the term $\boldsymbol{u}_h\theta_h$ is nonlinear, an over-integration strategy according to the 3/2-rule can be used (assuming $k_\theta = k_u$) in the current implementation as an alternative to the standard quadrature rule with $k_\theta + 1$ quadrature points. Unless specified otherwise, this over-integration strategy will be used for the numerical results shown in this chapter.

3.4 Stability of flow–transport coupling for the discrete problem

In Section 3.1, the requirement has been formulated that the numerical solution should not depend on the temperature reference value θ_{ref} for robust flow–transport solvers. To test whether the absolute temperature level impacts the numerical solution for the discrete problem, one can insert $\theta_h(\boldsymbol{x},t) = \theta_{\text{ref}} + \Delta\theta_h(\boldsymbol{x},t)$ with $\theta_{\text{ref}} = \text{const}$ into the weak formulation and exploit the fact that the transport term is linear in θ for both convective and conservative formulations. For a robust numerical method, the term containing θ_{ref} should vanish. In case of the conservative formulation, this term becomes

$$c_{h,\theta}^e\left(\iota_h,\theta_{\text{ref}},\boldsymbol{u}_h\right) = \theta_{\text{ref}}\left(-\left(\nabla\iota_h,\boldsymbol{u}_h\right)_{\Omega_e} + \left(\iota_h,\{\{\boldsymbol{u}_h\}\}\cdot\boldsymbol{n}\right)_{\partial\Omega_e}\right) \stackrel{?}{=} 0 \ , \tag{3.16}$$

The above equation looks like the continuity equation (2.77) with the velocity divergence term according to equation (2.97). However, care has to be taken since the test function belongs to the finite element space for the temperature, which might be richer than the space for the pressure. Hence, the above term does not become zero in general. Indeed, this term vanishes for normal-continuous and exactly divergence-free velocity approximations under the assumption of exact numerical integration according to equation (2.166). However, for L^2-conforming discretizations, this term vanishes only if (i) the coupled solution approach is used where the divergence and continuity penalty terms are applied in the momentum equation (2.187) instead of a separate postprocessing equation (2.191), so that the continuity equation (2.188) is fulfilled, (ii) the velocity divergence term is discretized in weak form, equation (2.97), or integrals are evaluated exactly so that strong and weak formulations of the discrete velocity divergence term are identical, and (iii) the temperature weighting functions are within the space of pressure weighting functions, $\mathcal{V}_h^\theta \subseteq \mathcal{V}_h^p$. Applying the penalty terms in a postprocessing step can be expected to perturb the discrete continuity equation (2.188). The discrete continuity equation will also not be fulfilled exactly for projection-type solution methods. Using mixed-order polynomials for velocity and pressure, $k_p = k_u - 1$, requires $k_\theta \leq k_p < k_u$ in order to fulfill the above condition for the temperature weighting functions.

Instead, one obtains the following result in case of the convective formulation

$$c_{h,\theta}^e\left(\iota_h,\theta_{\text{ref}},\boldsymbol{u}_h\right) = -\left(\iota_h,\left(\boldsymbol{u}_h\cdot\nabla\right)\theta_{\text{ref}}\right)_{\Omega_e} + \left(\iota_h,\{\{\boldsymbol{u}_h\}\}\cdot\boldsymbol{n}\left(\theta_{\text{ref}}-\theta_{\text{ref}}\right)\right)_{\partial\Omega_e} = 0 \ , \tag{3.17}$$

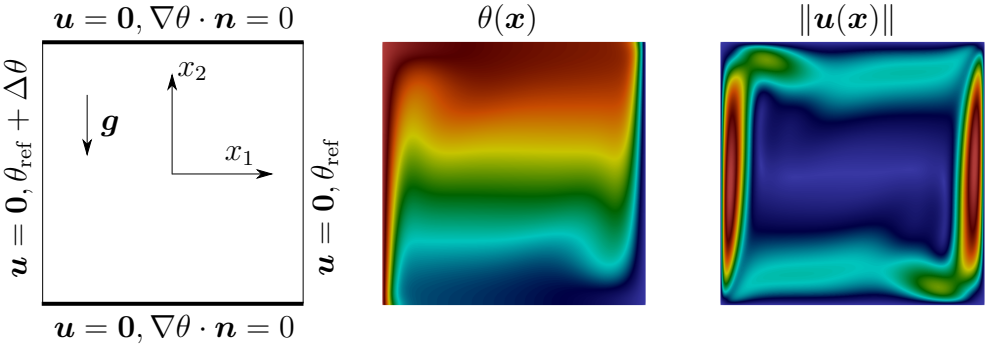


Figure 3.1: Differentially heated cavity: illustration of geometry and boundary conditions as well as steady-state solution for $\mathrm{Ra} = 10^6, \mathrm{Pr} = 1$.

i.e., this problematic term vanishes for all discrete velocity fields $\boldsymbol{u}_h$, in particular for those that are not exactly divergence-free or that are discontinuous between elements. Note that this line of argumentation is identical to the compatibility condition for flow–transport problems discussed in Dawson et al. (2004), where this property of discretization schemes for flow–transport problems is denoted as zeroth-order accuracy and is examined for different continuous and discontinuous Galerkin schemes. Zeroth-order accuracy means that a constant temperature field is preserved in the discrete case independently of the discrete velocity field. A discontinuous Galerkin discretization of the transport term in convective formulation as preferred here has, however, not been discussed in Dawson et al. (2004). Furthermore, the point here is that this property does not only affect accuracy, but essentially may trigger numerical instabilities for coupled flow–transport problems such as problems of natural convection under the Boussinesq approximation discussed in this chapter. One can conclude that the convective formulation of the transport term is compatible to all incompressible Navier–Stokes solvers discussed in this work without requiring exactly mass-conserving velocity fields or a certain polynomial degree for the temperature shape functions.

Remark 3.1 *The diffusive term is unproblematic w.r.t. shifting the temperature level by a constant value, since all terms of the SIPG discretization contain either derivatives or jumps that vanish in case of a constant solution, i.e.,* $l_h^e\left(\iota_h, \theta_{\text{ref}}\right) = 0$.

Remark 3.2 *According to the above argumentation, one might suspect that the formulations proposed in Bassi et al. (2007), Busto et al. (2020), Franciolini et al. (2017), Piatkowski (2019) using the conservative formulation of the transport term (see Table 3.1) do not guarantee robustness regarding the flow–transport coupling if* $k_\theta > k_p$.

Remark 3.3 *Note that the discussion regarding the stability of the flow–transport coupling is very similar to that of pressure-robustness according to equation* (2.168)*. While exactly mass-conserving formulations heal both problems, it should be emphasized that the effect discussed here should be considered independently of the aspect of pressure-robustness. A pressure-robust method is not necessarily exactly mass-conserving and might therefore still suffer from the problem discussed here.*

Remark 3.4 *Note that there is a strong analogy in the argumentation between the aspect of zeroth-order accuracy discussed here and the geometric conservation law properties of ALE formulations for moving meshes (see Section 8.4.5), addressing the question whether the discrete formulation is able to preserve a constant flow field independently of the mesh velocity (free-stream preservation property). Interestingly (or expectedly), the convective formulation is also the natural formulation for ALE-type problems.*

3.5 Numerical results

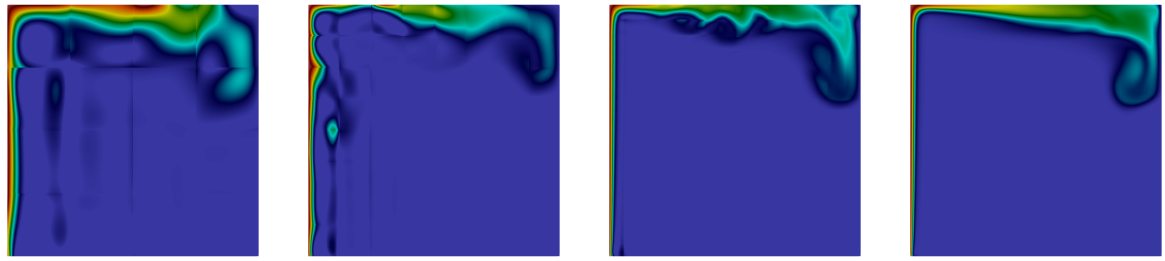
This section shows numerical results for benchmark problems frequently studied in the context of natural convection solvers. The robustness of the conservative and convective formulations is investigated for the differentially heated cavity problem. Then, the problem of a rising thermal bubble in the limit of vanishing viscosity and thermal diffusivity is studied using the convective formulation. Finally, results are presented for Rayleigh–Bénard convection problems in two and three space dimensions, and an application on spherical domains is studied which is motivated by earth mantle convection problems.

3.5.1 Two-dimensional differentially heated cavity

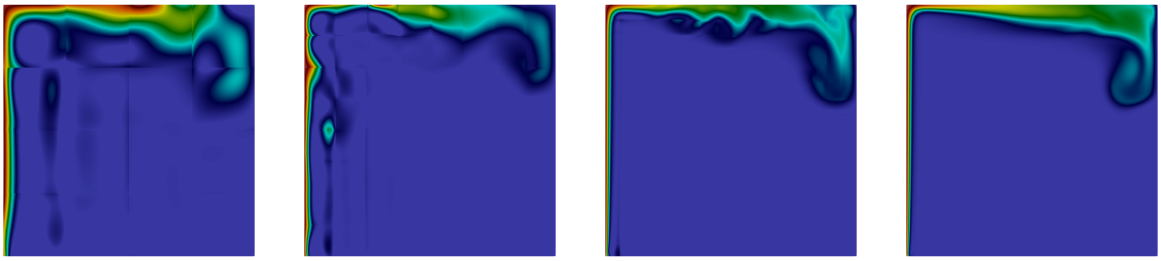
As a first example, the heated cavity problem is considered, which is a standard benchmark for natural convection flow solvers and has been studied for example in Akbas et al. (2018), Bassi et al. (2007), Busto et al. (2020), Schroeder and Lube (2017) in the context of high-order DG discretizations. A large Rayleigh number of $\mathrm{Ra} = 10^8$ is considered here, focusing on aspects of robustness of the discretization and under-resolved application scenarios.

The problem setup is described in Figure 3.1. The domain is a rectangular box $\Omega = [-0.5, 0.5]^2$ in two space dimensions with no-slip boundary conditions on all boundaries, $\boldsymbol{u} = \boldsymbol{0}$. Dirichlet boundary conditions are prescribed for the temperature at the left and right boundaries, where the left boundary is heated, $\theta_{\mathrm{left}} = \theta_{\mathrm{ref}} + \Delta\theta > \theta_{\mathrm{right}} = \theta_{\mathrm{ref}}$. The lower and upper boundaries are adiabatic, $\nabla\theta \cdot \boldsymbol{n} = 0$. At time $t = 0$, set $\theta = \theta_{\mathrm{ref}}$ and $\boldsymbol{u} = \boldsymbol{0}$. The following parameters are chosen: $L = 1, \boldsymbol{g} = (0, -10)^{\mathsf{T}}, \mathrm{Pr} = 1, \mathrm{Ra} = 10^8, \Delta\theta = 1, \beta = 1/300$. Then, the diffusivities are given as $\nu = \sqrt{g\beta\Delta\theta L^3 \mathrm{Pr}/\mathrm{Ra}}$ and $a = \nu/\mathrm{Pr}$. The simulation is performed over a time interval of $0 \leq t \leq T = 10L/U$, where $U = \sqrt{g\beta\Delta\theta L}$ is the characteristic velocity. Note that a much longer time interval would be necessary to reach the steady-state solution for this high Rayleigh number. Figure 3.1 visualizes the steady-state temperature and velocity fields for a smaller Rayleigh number of $\mathrm{Ra} = 10^6$ and otherwise identical parameters.

The incompressible flow problem is solved using the coupled solution approach. The BDF2 time integration scheme is used for the incompressible flow problem and also for the temperature equation. Convective terms are treated explicitly, where adaptive time stepping is used with $\mathrm{Cr} = 0.3$. Absolute and relative solver tolerances are set to $\varepsilon_{\mathrm{abs}} = 10^{-12}$ and $\varepsilon_{\mathrm{rel}} = 10^{-6}$. The mesh is uniform Cartesian with $(2^l)^2$ elements. In the following, the convective and conservative formulations of the transport term in the temperature equation are compared. Figure 3.2 shows results for the convective form and Figure 3.3 for the conservative form. In both cases, four different mesh resolutions are considered, $l = 2, \ldots, 5$ ($4^2, \ldots, 32^2$ elements) with degree $k_u = k_\theta = k = 3$ each, and two values of the reference temperature, $\theta_{\mathrm{ref}} = 0$ and 300.



(a) $l = 2, \theta_{\text{ref}} = 0$ (b) $l = 3, \theta_{\text{ref}} = 0$ (c) $l = 4, \theta_{\text{ref}} = 0$ (d) $l = 5, \theta_{\text{ref}} = 0$



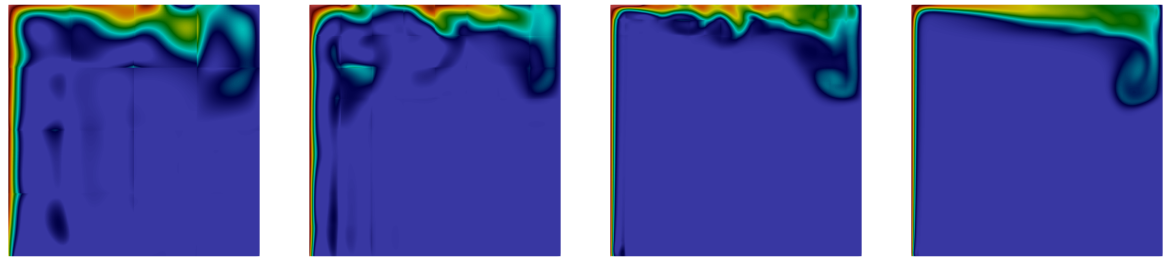
(e) $l = 2, \theta_{\text{ref}} = 300$ (f) $l = 3, \theta_{\text{ref}} = 300$ (g) $l = 4, \theta_{\text{ref}} = 300$ (h) $l = 5, \theta_{\text{ref}} = 300$

Figure 3.2: Heated cavity problem at $\text{Ra} = 10^8$: visualization of temperature field $\theta(\boldsymbol{x}, t = T)$ for *convective* formulation of transport term considering different mesh refinement levels $l = 2, \ldots, 5$ for polynomial degree $k = 3$ and for different values of the reference temperature, $\theta_{\text{ref}} = 0$ and $\theta_{\text{ref}} = 300$.

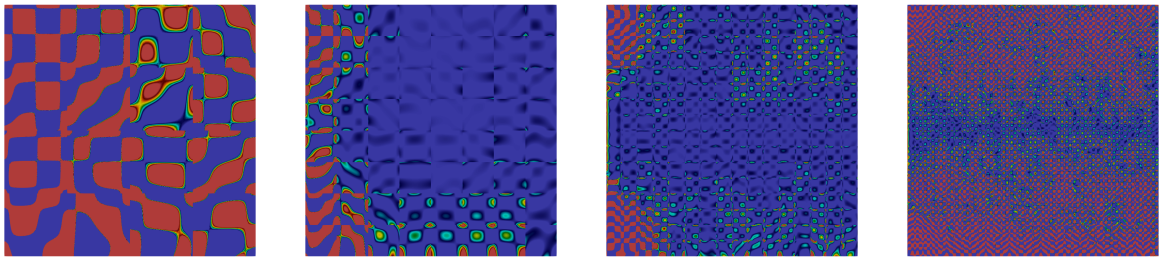
For the convective form, results are identical for both values of the reference temperature as expected theoretically. Under-resolution effects are visible for coarse spatial resolutions and the temperature field appears to be well-resolved for the higher mesh refinement levels. For the conservative formulation, stable results are obtained for $\theta_{\text{ref}} = 0$. The results are in qualitative agreement with those for the convective formulation. For $\theta_{\text{ref}} = 300$, however, the simulation becomes unstable for all refinement levels, where Figure 3.3 shows temperature fields obtained before the simulation crashes. The results are in agreement with the theoretical considerations in Section 3.4. According to these results, it would be interesting to test the robustness of some of the discretization schemes listed in Table 3.1 for the test case studied here. In general, an interesting robustness test for discretizations using the conservative formulation of the transport term is to use a temperature degree $k_\theta > k_p$ and then increase θ_{ref}, since in this case $\mathcal{V}_h^\theta \not\subseteq \mathcal{V}_h^p$, which might trigger instabilities for discretization schemes that are robust for $k_\theta \leq k_p$ according to the considerations in Section 3.4. The work by Schroeder and Lube (2017) uses the convective formulation that appears to be unproblematic according to the present results.

3.5.2 Two-dimensional rising thermal bubble

The rising thermal bubble test case follows the setup shown in Busto et al. (2020). For this example, the limit of vanishing diffusivities is considered, $\nu = 0$ and $a = 0$, which is the setup



(a) $l = 2, \theta_{\text{ref}} = 0$ (b) $l = 3, \theta_{\text{ref}} = 0$ (c) $l = 4, \theta_{\text{ref}} = 0$ (d) $l = 5, \theta_{\text{ref}} = 0$



(e) $l = 2, \theta_{\text{ref}} = 300$ (f) $l = 3, \theta_{\text{ref}} = 300$ (g) $l = 4, \theta_{\text{ref}} = 300$ (h) $l = 5, \theta_{\text{ref}} = 300$

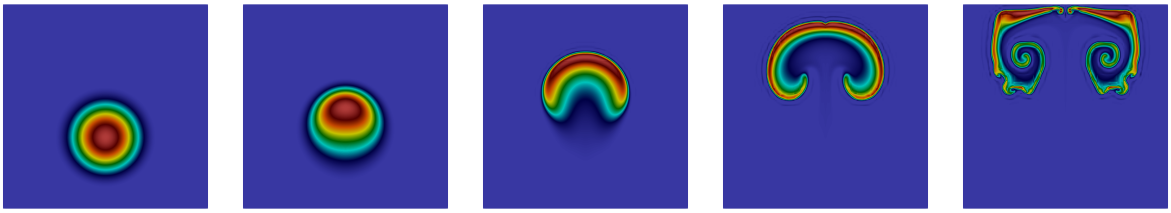
Figure 3.3: Heated cavity problem at $\text{Ra} = 10^8$: visualization of temperature field $\theta(\boldsymbol{x}, t = T)$ for *conservative* formulation of transport term considering different mesh refinement levels $l = 2, \ldots, 5$ for polynomial degree $k = 3$ and for different values of the reference temperature, $\theta_{\text{ref}} = 0$ and $\theta_{\text{ref}} = 300$.

most challenging for a discretization scheme to remain numerically stable and produce accurate results. The computational domain is a rectangular box, $\Omega = [0, 1000]^2$. No-slip boundary conditions are prescribed for the incompressible flow problem, and Dirichlet boundary conditions with $\theta = \theta_0$ for the temperature. The initial velocity field is $\boldsymbol{u} = \boldsymbol{0}$, and the initial temperature field describes a warm bubble with center $\boldsymbol{x}_{\text{b}} = (500, 350)^{\text{T}}$

$$\theta(\boldsymbol{x}, t = 0) = \theta_0 + \begin{cases} 0 \,, & \text{if } r > r_{\text{b}} \,, \\ \frac{\theta_{\text{b}}}{2} \left(1 + \cos\left(\frac{\pi r}{r_{\text{b}}}\right)\right) \,, & \text{if } r \leq r_{\text{b}} \,, \end{cases} \tag{3.18}$$

where $r = \|\boldsymbol{x} - \boldsymbol{x}_{\text{b}}\|$ is the radial distance to the center $\boldsymbol{x}_{\text{b}}$, $r_{\text{b}} = 250$ the bubble radius, and $\theta_{\text{b}} = 0.5$ the maximum temperature difference of the warm bubble. Furthermore, the reference temperature is $\theta_0 = 303.15$, the thermal expansion coefficient is $\beta = 1/\theta_0$, and the gravitational force is $g = 9.81$. The simulated time interval is $0 \leq t \leq 1000$.

The dual splitting scheme is used for the solution of the incompressible flow problem with standard parameters as defined in Section 2.6.1. Regarding the scalar transport equation, only the convective formulation is studied. As for the previous example, it was found that the conservative form of the transport term lacks robustness, especially for a large value of the reference temperature. The BDF2 time integration scheme is used with an explicit formulation of convective terms and adaptive time stepping with $\text{Cr} = 0.3$ is used. Absolute and relative solver



(a) $\theta(\boldsymbol{x}, t = 0)$ (b) $\theta(\boldsymbol{x}, t = 200)$ (c) $\theta(\boldsymbol{x}, t = 400)$ (d) $\theta(\boldsymbol{x}, t = 600)$ (e) $\theta(\boldsymbol{x}, t = 800)$

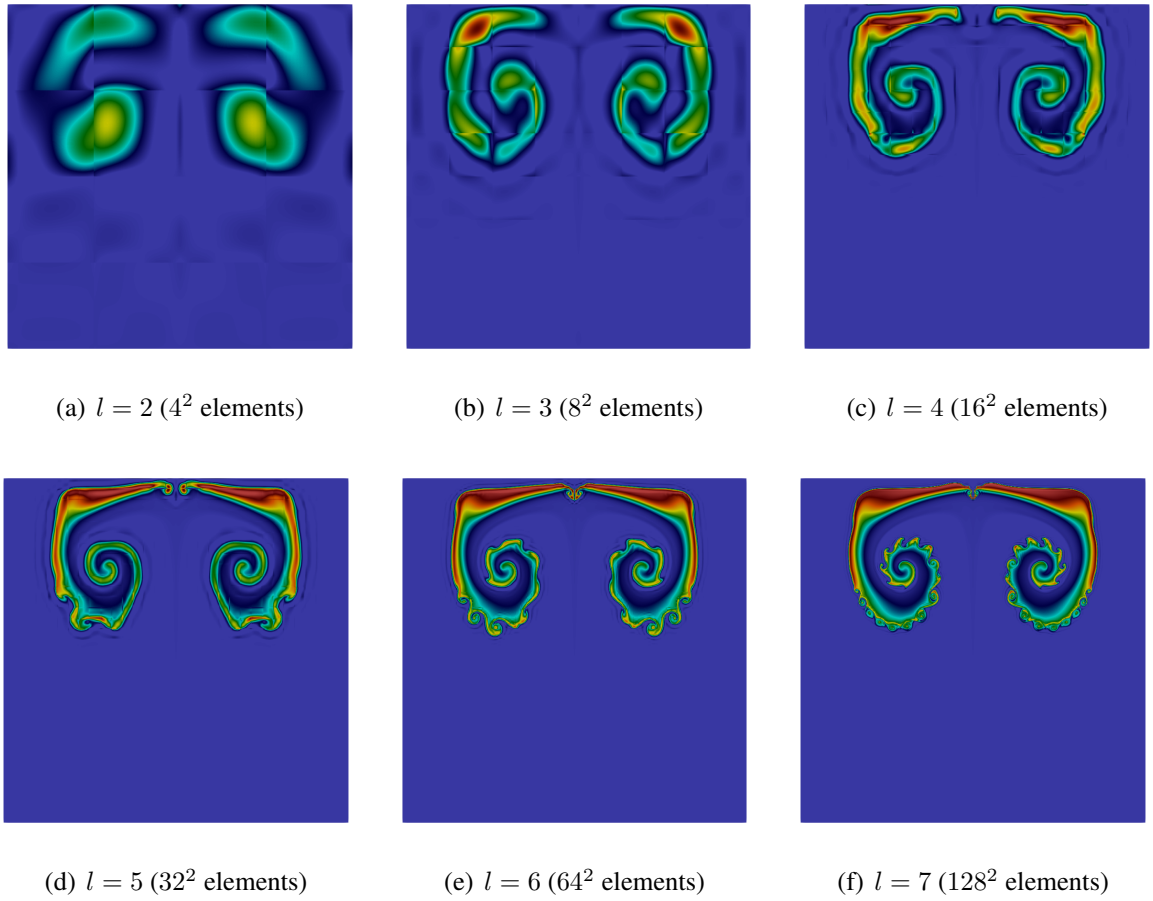
Figure 3.4: Rising thermal bubble: visualization of temperature field for a mesh with refinement level $l = 5$ (32^2 elements) and polynomial degree $k = 3$.

tolerances are set to $\varepsilon_{\text{abs}} = 10^{-12}$ and $\varepsilon_{\text{rel}} = 10^{-6}$. The mesh is uniform Cartesian with $(2^l)^2$ elements.

Figure 3.4 visualizes the temperature field as a function of time for a spatial resolution of 32^2 elements with polynomial degree $k_u = k_\theta = k = 3$. The bubble begins to rise due to buoyancy forces and develops a mushroom-like shape. At later times, more complex flow structures develop. The tails of the bubble begin to roll up and small flow features form that are triggered by Kelvin-Helmholtz instabilities. Figure 3.5 shows the temperature field at time $t = 800$ for a sequence of spatial resolutions considering meshes from 4^2 to 128^2 elements with polynomial degree $k = 3$. Due to the symmetry of the mesh, the results also remain symmetric w.r.t. the axis $x_1 = 500$. These results demonstrate that the proposed discretization scheme produces a clean solution without signs of instabilities and for resolutions much coarser than in Busto et al. (2020), where artificial diffusivity is used for the Eulerian solver and where a mesh composed of 5172 triangles (and degree $k = 4$) is considered very coarse. These results also illustrate how increasingly fine flow structures can be resolved by an increasing spatial resolution, or how small features get smeared in under-resolved scenarios.

3.5.3 Rayleigh–Bénard convection

The Rayleigh–Bénard convection problems studied here are inspired by the examples shown in Busto et al. (2020), Franciolini et al. (2017), Piatkowski (2019). The domain is $\Omega = [0, L] \times [0, H]$ in two dimensions and $\Omega = [0, L] \times [0, H] \times [0, L]$ in three dimensions with $L/H = 8$. The bottom boundary at $x_2 = 0$ is heated, $\theta_{\text{b}} > \theta_{\text{ref}}$, relative to the top boundary at $x_2 = H$, $\theta_{\text{t}} = \theta_{\text{ref}} = 0$. No-slip boundary conditions are prescribed for the velocity at the top and bottom boundaries, and periodic boundary conditions are prescribed in x_2 (and x_3) directions. The following parameters are set: $\text{Pr} = 1$, $g = 10$, $\beta = 1/300$, $H = 1$, $U = 1$. Then, prescribing Re or $\text{Ra} = \text{Re}^2\text{Pr}$ yields $\nu = UH/\text{Re}$, $a = \nu/\text{Pr}$, $\Delta\theta = U^2/(g\beta H)$. To introduce some imperfections and provoke the formation of thermal plumes, a sine-like perturbation of the temperature is prescribed at the bottom boundary. These perturbations are prescribed only over the first non-dimensional time unit (decreasing like $1 - t/(H/U)$ and turning into a spatially constant Dirichlet boundary condition for $t > H/U$) to make sure that the solution obtained at later times does not directly depend on the initial perturbations.



(a) $l = 2$ (4^2 elements) (b) $l = 3$ (8^2 elements) (c) $l = 4$ (16^2 elements)

(d) $l = 5$ (32^2 elements) (e) $l = 6$ (64^2 elements) (f) $l = 7$ (128^2 elements)

Figure 3.5: Rising thermal bubble: visualization of temperature field $\theta(\boldsymbol{x}, t = 800)$ for a mesh refinment study with refinement levels $l = 2, \ldots, 7$ ($4^2, \ldots, 128^2$ elements) and polynomial degree $k = 3$.

The dual splitting scheme is used for this problem, the BDF2 scheme, and adaptive time stepping with $\mathrm{Cr} = 0.4$. Absolute and relative solver tolerances are set to $\varepsilon_{\mathrm{abs}} = 10^{-12}$ and $\varepsilon_{\mathrm{rel}} = 10^{-6}$ unless specified otherwise. Figure 3.6 shows results for $\mathrm{Ra} = 10^8$ on a 256×32 Cartesian mesh and polynomial degree $k = 3$. The temperature field is shown for several time snapshots illustrating the formation of thermal plumes in the beginning. At later times, the formation of typical Rayleigh–Bénard convection rolls can be observed. Due to the high Rayleigh number, the behavior of the coupled flow–transport problem is highly unsteady and chaotic, so that no steady state is reached. The pictures also illustrate how the fluid heats up continuously over time (which goes on until an equilibrium state is reached in which the heat flux over the top boundary equals the heat flux over the bottom boundary). In case of an (alternative) adiabatic boundary condition at the top boundary, the fluid would finally reach a temperature of $\theta = \theta_{\mathrm{b}}$ throughout the domain.

Figure 3.7 shows results of a three-dimensional Rayleigh–Bénard convection problem for a Rayleigh number of $\mathrm{Ra} = 10^6$ as shown in Piatkowski (2019). A mesh with $128 \times 16 \times 128$ elements with polynomial degree $k = 3$ is considered and the simulation is run over the time

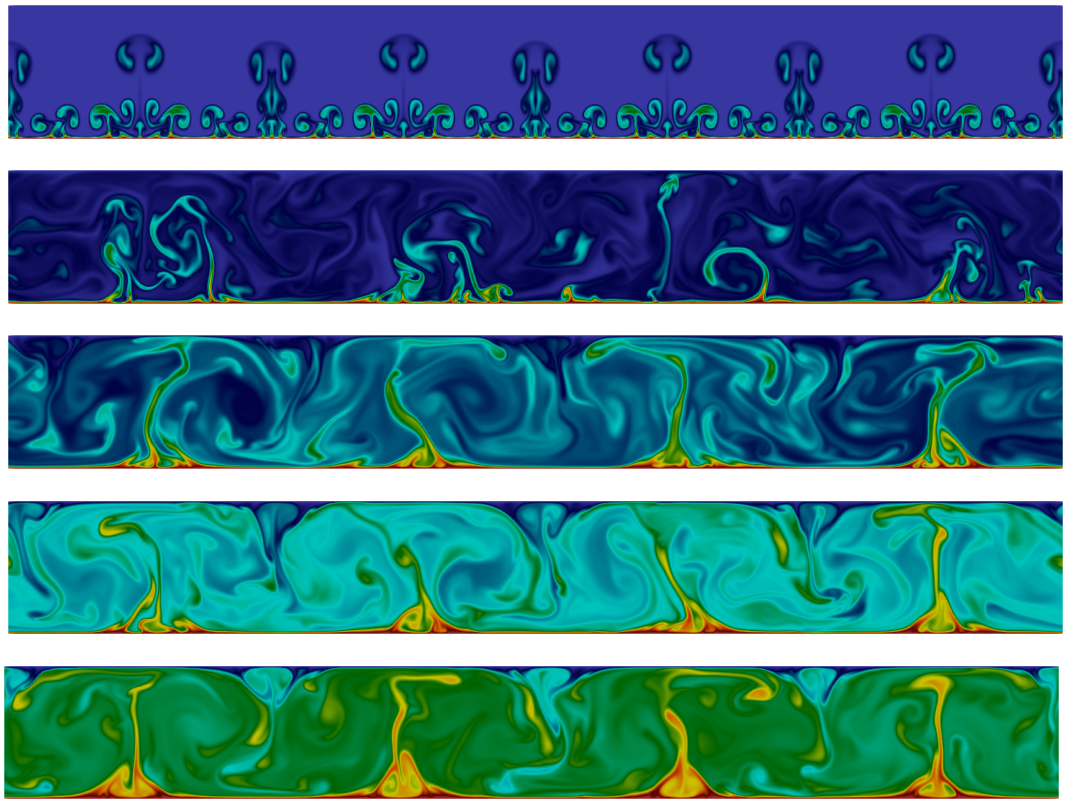
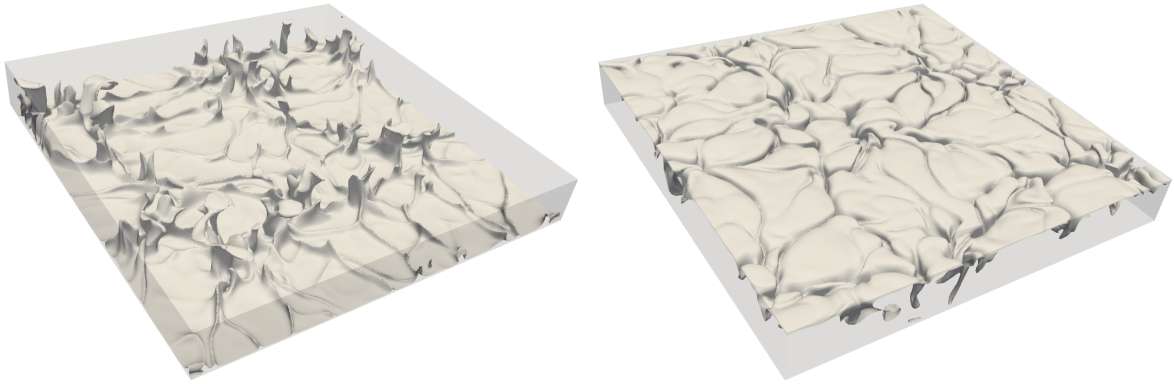


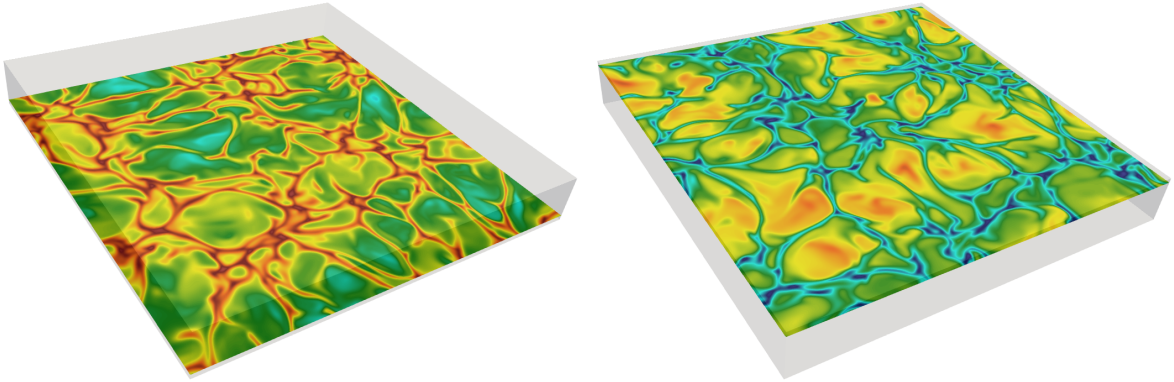
Figure 3.6: 2D Rayleigh–Bénard convection at $\mathrm{Ra} = 10^8$: temperature field at $t/(H/U) = 16, 32, 64, 100, 200$ (from top to bottom) obtained on a 256×32 mesh with polynomial degree $k = 3$ (red indicates a high temperature and blue a low temperature).

interval $0 \leq t/(H/U) \leq 200$. The time stepping parameters are identical to the two-dimensional case. Figure 3.7 illustrates how hot thermal plumes are rising up from the heated bottom plate, while cold plumes are falling down from the cold top plate. Note that the temperature field looks somewhat different than in Piatkowski (2019) because the same global color scale is used here for different slices.

In order to give insights into the computational efficiency of the present solver, performance numbers are compared to results published recently in Franciolini et al. (2017), where a simulation at $\mathrm{Ra} = 10^8$ with approximately 52 MDoF and global degree $k = 5$ simulated over a time interval of $200H/U$ has been reported to require a wall time of less than one day on 68 Intel Xeon nodes with 1224 cores in total. For the same setup in terms of Rayleigh number and simulated time interval, the present solver requires for a problem with approximately 51 MDoF and tensor degree $k = 5$ a wall time of 23.5 hours for a relative tolerance of $\varepsilon_{\mathrm{rel}} = 10^{-3}$ and 41.9 hours for $\varepsilon_{\mathrm{rel}} = 10^{-6}$, but using only a single Intel Haswell node with 24 cores. This points to a performance advantage of one to two orders of magnitude for the present solver. Note that this performance advantage is achieved despite the fact that the mixed implicit/explicit solver used here can be expected to require significantly more time steps. This already indicates that the ability to use large time step sizes (Franciolini et al. 2017) might not necessarily be a convincing



(a) iso-surfaces of temperature for $\theta = 2\theta_b/3$ (left) and $\theta = \theta_b/3$ (right)



(b) contour plots of temperature in planes located at $x_2/H = 0.1$ (left) and $x_2/H = 0.9$ (right)

Figure 3.7: 3D Rayleigh–Bénard convection at $\mathrm{Ra} = 10^6$: visualization of temperature field at final time $t/(H/U) = 200$ obtained on a $128 \times 16 \times 128$ mesh with polynomial degree $k = 3$. The same color scale is used for all figures (where red indicates a high temperature and blue a low temperature).

argument for the use of implicit solvers. It is the content of subsequent chapters to shed light on these numbers.

3.5.4 Earth mantle convection

As a final application problem, a simplified model for earth mantle convection is studied as analyzed in Gmeiner et al. (2015a). From a physical point of view, the problem is very similar to the Rayleigh–Bénard convection problems studied in the previous section, where a main difference is that a spherical geometry is considered here. The simplified model considers quasi-steady Stokes flow with constant viscosity under the Boussinesq approximation coupled to an unsteady

Figure 3.8: 3D mantle convection at $\mathrm{Ra} = 10^7, \mathrm{Pe} = 1$: iso-surfaces of temperature field $\theta/(\theta_r - \theta_R) = 0.2$ at $t/T_0 = 10, 15, 50, 100$ (from left to right) obtained on a mesh with refine level $l = 4$ and polynomial degree $k = 3$ (approximately 56 MDoF in total).

convection–diffusion problem for the temperature in the following non-dimensional form

$$-\nabla^2 \boldsymbol{u} + \nabla p = -\mathrm{Ra}\frac{1}{\mathrm{Pe}}\theta\left(-\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\right) \,, \tag{3.19}$$

$$\nabla \cdot \boldsymbol{u} = 0 \,, \tag{3.20}$$

$$\frac{\partial \theta}{\partial t} + (\boldsymbol{u} \cdot \nabla)\,\theta - \frac{1}{\mathrm{Pe}}\nabla^2 \theta = 0 \,. \tag{3.21}$$

Due to the large viscosity of the rock material in earth's mantle, the inertial term and convective term are neglected in the momentum equation. The problem is solved on a spherical shell with inner radius $r = 0.55$ and outer radius $R = 1$. The velocity and temperature fields are zero at start time $t = 0$. For the temperature, a Dirichlet value of $\theta_r = 1$ is prescribed at the inner radius and a value of $\theta_R = 0$ at the outer radius. As for the previous example, temperature fluctuations are added to the Dirichlet boundary value at the inner boundary at small times in order to break the symmetry. No-slip boundary conditions are prescribed for the velocity on all boundaries. The mesh is `dealii::GridGenerator::hyper_shell()` with 48 elements for the coarse grid that is refined uniformly l times, where polynomial degree $k = 3$ is considered in the following. A high-order mapping of degree $k_\mathrm{m} = k$ is used here for an accurate representation of the geometry, which is assumed spherical here but could also follow a more complex description of the surface topology imposed via transfinite interpolation techniques or mesh movement algorithms (see also Section 9.1.3). Following Gmeiner et al. (2015a), the non-dimensional parameters are set to $\mathrm{Ra} = 10^7$ and $\mathrm{Pe} = 1$. Defining a time unit as $T_0 = H/U$ with $H = R - r$ and $U = \sqrt{\mathrm{Ra}/H^2}$, the simulations are run over the time interval $0 \leq t/T_0 \leq 100$. Figure 3.8 shows results of 3D simulations for the simplified mantle convection model. Iso-surfaces of the temperature field are presented for different instants of time showing the development of thermal plumes and their chaotic behavior at later times.

Since the steady Stokes equations have to be solved, the coupled solution approach is used here (and no divergence and continuity penalty terms are applied in this example). Adaptive time stepping is used for the temperature equation, which solely determines the global time step size for this coupled flow–transport problem since the flow problem itself does not introduce a time step restriction. In the following, both an explicit treatment of the convective term (as used in Gmeiner et al. (2015a) with explicit CFL-type time step restriction) and an implicit treatment of the convective term (in order to relax the CFL condition and improve overall efficiency)

Table 3.2: 3D mantle convection at $\mathrm{Ra} = 10^7, \mathrm{Pe} = 1$: throughput per time step Δt and computational costs per time unit T_0 for polynomial degree $k = 3$ and mesh refinement levels $l = 2, 3, 4$ resulting in problem sizes of $\{7.0 \cdot 10^6, 5.6 \cdot 10^7, 4.5 \cdot 10^8\}$ degrees of freedom solved on $\{1, 1, 8\}$ Intel Haswell compute nodes with 24 cores each.

l	Throughput per time step [MDoF/s/core]		Computational costs per time unit [CPUh]	
	explicit	implicit	explicit	implicit
2	0.401	0.321	0.41	0.05
3	0.380	0.233	4.49	1.06
4	0.278	0.173	74.3	23.4

are considered. For the explicit case, Courant numbers in the range $\mathrm{Cr} = 0.2 - 0.4$ are used depending on the spatial resolution. For the implicit case, the Courant number is $\mathrm{Cr} = 2$ in all cases. To solve the Stokes problem, a block-triangular preconditioner is used with a cph-multigrid preconditioner with Chebyshev(5) smoother for the velocity block and the inverse pressure mass matrix preconditioner for the Schur-complement block (see also Chapter 5 for explanations). The inverse mass matrix preconditioner is also used for the unsteady heat equation to obtain a computationally efficient iterative solver. Linear systems of equations are solved to a relative tolerance of $\varepsilon_{\mathrm{rel}} = 10^{-3}$.

The main interest in this example is the aspect of computational efficiency of the solver. In Gmeiner et al. (2015a), a state-of-the-art high-performance finite element solver for tetrahedral grids is presented, for which each time step of a problem with $6.5 \cdot 10^9$ unknowns requires a wall time of approximately 60 s on 8 nodes (with 32 cores each) of Intel Xeon E7-4830 CPUs. To compare the node level performance, a suitable quantity is the throughput in terms of degrees of freedom solved per second of wall time per time step and per core (see Chapters 5 and 6), which results in a throughput of 0.423 MDoF/s/core for the hierarchical hybrid multigrid Stokes solver in Gmeiner et al. (2015a). Results for the present solver are listed in Table 3.2 for three different mesh resolutions, considering both explicit and implicit formulations of the convective term. A metric even better suited for such comparisons is to consider computational costs for a fixed physical time interval that is simulated (here T_0). The corresponding results are also listed in Table 3.2. However, results for this metric have not been specified in Gmeiner et al. (2015a). The explicit solver achieves a throughput approaching the performance numbers from Gmeiner et al. (2015a). The throughput decreases somewhat with the mesh refinement level and the scalar transport equation consumes less than 10% of the overall costs. The throughput is lower for the implicit formulation since the scalar transport equation now takes a larger share of the overall computational costs, and the throughput decreases more strongly with increasing refinement level since the iteration counts for the scalar transport equation (inverse mass matrix preconditioner is used) increase significantly with increasing l in case of the implicit formulation with $\mathrm{Cr} = 2$ (aspects of iterative solvers and preconditioners are discussed in more detail in Chapter 5). Nevertheless, the implicit formulation is significantly more efficient in all cases, with speed-up factors of 8.1 ($l = 2$), 4.2 ($l = 3$), and 3.2 ($l = 4$) compared to the explicit formulation.

In summary, these results show that the present high-order DG solver can compete with highly-optimized software for earth mantle convection problems that uses optimized stencil-like implementation techniques for linear finite elements to achieve optimal efficiency. The present results are also encouraging given that a generic DG solver is used here that can be applied to arbitrary geometries with its only (or main) restriction being hexahedral meshes, which is for example no restriction for earth mantle convection applications, see Kronbichler et al. (2012), Rudi et al. (2015). Of course, it should be taken into account that the present results have been obtained on newer hardware, and that the high-order approach used in the present work can be expected to be more accurate than the linear finite element approach used in Gmeiner et al. (2015a). The use of an implicit solver potentially points to significant performance improvements also for other earth mantle convection solvers as presented in Gmeiner et al. (2015a), Kronbichler et al. (2012), provided an increase in time step size does not have a detrimental effect on accuracy (see also Chapter 6 for more holistic discussions).

3.6 Conclusion and outlook

This chapter has addressed the solution of natural convection problems by high-order L^2-conforming DG discretizations. To ensure robustness of the coupled flow–transport problem, a key ingredient is that a compatibility condition is fulfilled, an aspect that has not been addressed by many previous works. The most straight-forward way to ensure robustness regarding the flow–transport coupling is to use the convective formulation of the transport term in the temperature equation, since this formulation does not pose special requirements regarding the mass conservation properties of the velocity field or regarding the polynomial degree used for the temperature relative to the pressure degree. Furthermore, the use of a discontinuous Galerkin discretization for the temperature equation might prove advantageous compared to continuous finite element discretizations since no convection-stabilization appears to be necessary.

While natural convection problems are an important field of applications in terms of pressure-robustness of incompressible Navier–Stokes discretization schemes, the stabilized approach with divergence and continuity penalty terms appears to work very well in this context without requiring a readjustment of parameters compared to purely incompressible flow problems studied before. Hence, an interesting aspect for future investigations would be to study whether the stabilized L^2-conforming scheme exhibits deficiencies for practical applications compared to H^{div}-conforming discretizations that are exactly pressure-robust, given that no problems have been encountered so far for the examples studied here. According to the present results, it can also not be excluded that previous studies focusing on pressure-robustness interpreted problems as pressure-robustness issues that are actually related to the flow–transport coupling as discussed here. The present work suggests to separate the two concepts of pressure-robustness of an incompressible flow solver and compatibility of the flow–transport coupling.

Numerical results demonstrated the versatility and computational efficiency of the present DG framework that appears to be highly competitive compared to state-of-the-art solvers from the literature. While results on computational efficiency have been reported in this chapter without further explanations, it is of course no coincidence that a substantial speed-up has been achieved compared to the DG solver proposed in Franciolini et al. (2017), and that a performance competing with an optimized linear finite element solver proposed in Gmeiner et al. (2015a) was

achieved. Indeed, the present solver shares many properties with the approach in Gmeiner et al. (2015a) from an algorithmic and implementation point of view. Upcoming chapters will explain the reasons behind these performance numbers and shed light on the computational efficiency of the present solver in great detail.

4 Matrix-free implementation

Matrix-free methods describe algorithms that do not require the explicit storage of a matrix in order to solve the algebraic system of equations resulting from the discretization of a PDE. An intuitive way to think of matrix-free methods is a finite difference stencil evaluated on a uniform Cartesian grid, where the action of the matrix can be described by just a few coefficients characterizing the stencil. However, matrix-free methods can be realized for all the main types of discretizations, such as finite difference, finite volume, finite element, and discontinuous Galerkin schemes, as well as for complex geometries and deformed meshes. The main motivation for matrix-free methods is to enable the solution of larger problems due to reduced memory requirements, but more importantly, to solve a problem faster due to a significant reduction in the number of operations and in the transfer of data from main memory.

In the context of finite element methods, matrix-free methods are typically realized by on-the-fly operator evaluation using numerical quadrature. The efficiency of matrix-free finite element operator evaluation as compared to matrix-based approaches essentially depends on the polynomial degree of the shape functions and the type of elements. For quadrilateral and hexahedral element shapes, the tensor-product structure of the shape functions and of the quadrature rule can be exploited by a technique called sum-factorization, which leads to improved computational complexity in terms of operation counts and which is crucial in achieving efficient matrix-free methods. Moreover, the computer hardware under investigation and its characteristics in terms of floating point operations available relative to bytes of data transferable from main memory have an influence on the optimal implementation strategy for a given polynomial degree. The aim of this chapter is to provide the theoretical background required to understand these design choices and, through a review of literature, explain the rapid development and paradigm shift (both in terms of hardware and implementations/algorithms) that took place over the last two decades. In this sense, this chapter reveals why certain design choices made in the course of this thesis lead to efficient flow solvers with finite element operator evaluation as a main ingredient. The developments of matrix-free algorithms presented here is not a novelity of the present thesis but original work of Kronbichler and Kormann (2012, 2019). This chapter applies these matrix-free methods to problems of computational fluid dynamics, and documents the computational efficiency of this approach for the incompressible Navier–Stokes equations on modern computer hardware. Minor parts of this chapter present content that has already been published in Fehn et al. (2018a, 2019c, 2020).

The outline of this chapter is as follows. Section 4.1 discusses the state-of-the-art, Section 4.2 briefly introduces hardware characteristics and the roofline performance model, and Section 4.3 reviews important trends in computer hardware. Section 4.4 explains the basics of matrix-free operator evaluation and discusses differences to other implementation techniques. Numerical results investigating the efficiency of the present matrix-free implementation for the incompressible Navier–Stokes equations are shown in Section 4.5. Finally, this chapter is summarized in Section 4.6.

4.1 State-of-the-art

Matrix-free methods are predominantly used when it comes to large-scale, high-performance computations exploiting the largest supercomputers available. Independently of the type of discretization approach, this is the state-of-the-art implementation technique in HPC for fast PDE solvers, see for example Gmeiner et al. (2015a), Ichimura et al. (2015), May et al. (2014), Rudi et al. (2015) and the recent exa-scale initiatives in Arndt et al. (2020b), Bastian et al. (2020a), Bauer et al. (2020), Fischer et al. (2020b).

Matrix-free methods for high-order finite element or spectral element methods and the idea of sum-factorization date back to the work of Orszag (1980) published in the year 1980. A decade later (around 1990), the methodology was explored systematically by Fischer (1990), Fischer and Patera (1991), Fischer et al. (1988) in terms of optimal-complexity algorithms for tensor-product elements and high-performance, parallel implementations. In these works, the matrix-free technique with sum-factorization was described for continuous, collocation-type spectral element discretizations applied to the incompressible Navier–Stokes equations. Another decade later (around 2000), Fischer et al. (2002), Tufo and Fischer (1999) presented results for three-dimensional incompressible turbulent flows on complex geometries in a massively-parallel context. Again a decade later (2010-today), the interest in these methods has increased significantly, first for continuous finite element discretizations (Brown 2010, Cantwell et al. 2011, Kronbichler and Kormann 2012, May et al. 2014, Vos et al. 2010), and later also for discontinuous Galerkin discretizations for tensor-product elements (Hindenlang et al. 2012, Kempf et al. 2020, Kronbichler and Kormann 2019, Kronbichler et al. 2019, Kronbichler and Allalen 2018, Kronbichler et al. 2017, Müthing et al. 2017) and simplicial elements (Moxey et al. 2020a). The broad interest in these methods can also be explained by changes in computer hardware, reducing the break-even polynomial degree (the polynomial degree at which matrix-free methods become more efficient than matrix-based ones) continuously over time, and rendering matrix-free methods more efficient than their matrix-based variants already for quadratic elements of degree $k = 2$ on modern hardware (Kronbichler and Kormann 2012, 2019, May et al. 2014).

In more detail, trend-setting implementation techniques have been presented in Kronbichler and Kormann (2012, 2019) with innovative design choices such as vectorization over elements to exploit the SIMD capabilities of modern hardware in combination with highly-optimized sum-factorization kernels. These works demonstrate that optimizing both compute parts and memory access is essential for optimal throughput, which is related to the fact that the arithmetic intensity of matrix-free operator evaluation is in the range of the Flop-to-Byte ratio of current hardware. While many works have put their focus on the Flop metric, the early work by Tufo and Fischer (1999) already highlights the memory bandwidth as a potentially limiting resource for high-order, matrix-free continuous spectral element methods, an observation confirmed by most recent, leading contributions. In the DG context, the works by Kronbichler and Kormann (2019), Kronbichler and Allalen (2018), Kronbichler et al. (2017) discuss different strategies for the computation and scheduling of face integrals, where computing all face integrals corresponding to an element after each other, i.e., driving all work by a loop over elements, is found to be more efficient than separate loops for face integrals. This is due to a finite difference like data access pattern with improved temporal data locality for the approach with element-wise face integrals. DG discretizations for operators with second derivatives require the evaluation of gradients for face integrals, where typically all nodal degrees freedom contribute to the gradient

for the usual Gauss or Gauss–Lobatto bases. An innovative Hermite-like basis described in Kronbichler et al. (2019) requires only two layers of nodes to evaluate gradients on faces, since all other 1D shape functions have zero value and gradient at the boundaries of the unit interval $[0, 1]$. This Hermite-like basis optimizes the data access pattern and improves the throughput of matrix-free operator evaluation (Kronbichler and Kormann 2019, Kronbichler and Allalen 2018) and of multigrid solvers with appropriate smoothers (Kronbichler et al. 2019). The contributions by Kronbichler et al. are complemented by alternative vectorization strategies and aspects of code generation in Kempf et al. (2020), Müthing et al. (2017).

The present work considers matrix-free implementations targeting modern CPU hardware. Recently, several works addressed the topic of GPU acceleration for high-order finite element discretizations, see Kronbichler and Ljungkvist (2019), Ljungkvist (2017), Remacle et al. (2016), Świrydowicz et al. (2019) for Poisson and Helmholtz-like model problems (forming building blocks of incompressible flow solvers) and Franco et al. (2020), Karakus et al. (2019), Loppi et al. (2018), Otero et al. (2019) for incompressible flows solvers.

By now, matrix-free implementations have found widespread use in academic software for high-order finite element methods, e.g., in the `deal.II` (Arndt et al. 2020a), `DUNE` (Bastian et al. 2020b), `FLEXI` (Krais et al. 2020a), `MFEM` (Anderson et al. 2020), `Nek5000` (Fischer et al. 2020a), `Nektar++` (Moxey et al. 2020b), `PyFR` (Witherden et al. 2014) projects. Recently, benchmark problems have been defined and a comparative performance study between different finite element libraries has been performed in Fischer et al. (2020b).

An in-depth discussion of the topic is beyond the scope of this work, in particular the tight interconnection between low-level code optimizations and hardware concepts, all of which are required for optimal performance. Instead, this chapter intentionally chooses a high-level presentation of the general methodology, while the important aspects of low-level performance optimizations of the implementation used in this work are discussed in the studies by Kronbichler and Kormann (2012, 2019), which are highly recommended as further reading material. The contributions by Anderson et al. (2020), Cantwell et al. (2011), Vos et al. (2010) include illustrative graphical interpretations and are recommended for readers not particularly interested in aspects of computer science, but in the general concepts and how matrix-free operator evaluation is embedded into finite element codes, e.g., as compared to matrix-based methods. Finally, the reader is referred to the standard textbooks on spectral element methods (Deville et al. 2002, Karniadakis and Sherwin 2013, Kopriva 2009), and to Kronbichler (2021b) for a recent overview of the state-of-the-art.

4.2 Hardware characteristics and the roofline performance model

This chapter focuses on the most simple performance model of a CPU, the roofline performance model (Williams et al. 2009). According to this model, a CPU can perform arithmetic operations such as additions and multiplications whereby its execution units are fed with data streamed from main memory and passing the cache hierarchy of the CPU. Latency between operations or other expensive instructions are assumed to be negligible. According to such a simplified model, a CPU's performance is described by two characteristic hardware quantities, the maximum number

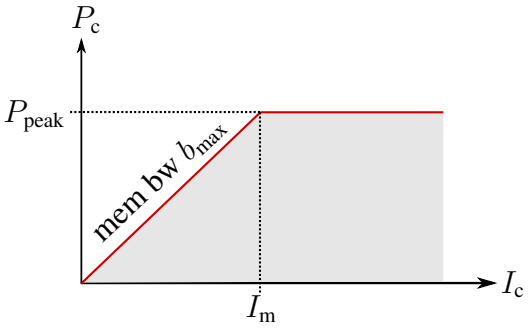


Figure 4.1: Visualization of the roofline performance model.

of floating point operations (Flop) that can be performed per unit of time (peak performance)

$$P_{\text{peak}} \text{ in } \frac{\text{Flop}}{\text{s}} , \tag{4.1}$$

and the amount of data that can be transferred from main memory per unit of time (memory bandwidth)

$$b_{\max} \text{ in } \frac{\text{Byte}}{\text{s}} . \tag{4.2}$$

An important quantity is the ratio of peak performance and memory bandwidth, denoted as machine intensity in this work

$$I_{\text{m}} = \frac{P_{\text{peak}}}{b_{\max}} \text{ in } \frac{\text{Flop}}{\text{Byte}} . \tag{4.3}$$

Another frequently used term is Flop-to-Byte ratio. Note that the machine intensity used here is the inverse of the machine balance used in Hager and Wellein (2010). The name *machine* intensity indicates that it is a hardware-specific quantity. From a software perspective, the ratio of floating point operations to data transfer from main memory of a particular code run on a machine is the code intensity $I_{\text{c}} = P_{\text{c}}/b_{\text{c}}$, also denoted as arithmetic or operational intensity. The roofline performance model (Williams et al. 2009) allows to characterize a code as compute-bound or memory-bound for a given hardware, depending on the code intensity I_{c} relative to what the hardware offers, the machine intensity I_{m}. If $I_{\text{c}} > I_{\text{m}}$, the code demands more floating point operations than can be delivered by the hardware, which is why such a code will fully utilize the compute resources, while the memory bandwidth is under-utilized. If $I_{\text{c}} < I_{\text{m}}$, the hardware offers more compute resources than required by the code, and the memory bandwidth will be the resource that limits the speed at which the code is executed according to the roofline model. The code performance can then be written as

$$P_{\text{c}} = \min\left(P_{\text{peak}}, b_{\max} I_{\text{c}}\right) . \tag{4.4}$$

A visualization of the roofline model is shown in Figure 4.1, where the ridge point is located at $(I_{\text{m}}, P_{\text{peak}})$. A code operating at this point fully utilizes all resources offered by the hardware, which does not imply that the speed of such a code can not be improved. For example, another algorithm implementing the same functionality might require less data transfer from main memory

or less operations, and might therefore let the code run faster despite the fact that the operational point leaves the ridge point in the roofline model. In practice, the operational point of a code will not be located exactly on the roofline, but within the gray area in Figure 4.1. This is due to the simplifying assumptions of the roofline model in the sense that data transfer and computations overlap perfectly and only one of the two machine quantities is the limiting resource for the whole code. Hence, characterizing a code in terms of the roofline model (by measuring the rates of data transfer and arithmetic operations) allows to quantify to which extent an algorithm exploits the capabilities of the hardware (how close does the point come to the roofline?), but not whether the algorithm itself is efficient or optimal.

To quantify the speed of the implementation, a different metric called throughput is used throughout this chapter, which is defined as the number of unknowns (degrees of freedom in finite element methods) processed per second of wall time and per compute unit (one node or one core per node)

$$E = \frac{N_{\mathrm{DoFs}}}{t_{\mathrm{wall}} N_{\mathrm{cores|nodes}}} \,. \tag{4.5}$$

Modern computer hardware is much more than floating point performance and memory bandwidth. A particularly important concept of CPU hardware, sometimes referred to in this work, is the concept of caches. Caches can be thought of as a smaller memory located closer to the CPU, providing higher bandwidth than main memory. Caches are organized in levels (typically level L1, L2, and L3 caches on modern hardware), which increase in size/capacity, but reduce in bandwidth and increase in latency in the given order. The aim of caches is to hide the latency and low bandwidth of main memory and to keep data ready for repeated use during a computation. The latter property is called temporal data locality, i.e., data touched previously resides in the cache and can be reused "instantaneously" if required, without transferring the data again from the slow main memory.

On modern hardware, data can not be transferred from main memory bit-wise or byte-wise, but is organized in units of so-called cache-lines (typically 64 Byte on current hardware). Since the memory bandwidth is a limited resource (an aspect discussed in more detail below), it is essential to make optimal use of the memory bandwidth and not pollute it with useless data that is not required for computations. This leads to the concept of spatial data locality, i.e., data located contiguously in memory should be worked on contiguously by the CPU. For scientific computing codes to be efficient, this requires an appropriate design of data structures. The abstract notion "data access pattern" already used above refers to the concepts of spatial and temporal data locality. One of the simplifications made in this chapter is that algorithms and implementations discussed here are assumed to be well-designed in this regard without diving into this topic in great detail.

The Single Instruction Multiple Data (SIMD) paradigm allows to perform elementary instructions simultaneously on an array of integer or floating point numbers, i.e., the size or width of registers is larger than just one integer or floating point value. This enables to execute more operations per clock cycle, thereby increasing the peak performance of a CPU. Since the register width is fixed for a given hardware, single precision computations can theoretically be performed twice as fast as double precision computations since twice as many data items fit into the same register width. From a software perspective, the technique that exploits this kind of parallelism offered by modern hardware is called vectorization. Put differently, a non-vectorized code for

which most lanes remain empty can only reach a small fraction of the peak performance of today's hardware.

The above concepts and further aspects of caches (mapping, prefetching, write-allocate or read-for-ownership) are discussed in more detail in Hager and Wellein (2010). This textbook also discusses other "advanced" techniques to accelerate the execution of a given instruction stream such as pipelining, superscalarity, out-of-order execution.

4.3 Recent trends in computer hardware

Computer hardware has undergone significant progress over the last decades and can be characterized by an exponential growth of "complexity" over time. An empirical law formulated by Gordon Moore in the 1960s states that the number of transistors on a chip doubles every one to two years (Moore 1965), a trend that still holds today. Figure 4.2 shows the development of microprocessors over the last 48 years. During that period, the number of transistors increased by a factor of 10^7 and shows a $N/N_0 = 2^{(t-t_0)/t_2}$ growth over time t, which implies a doubling period of $t_2 = 2.1$ years in accordance with Moore's law. As a sidenote, it is interesting to realize that also the peak performance of the world's largest supercomputer increases exponentially over time in line with Moore's law at a doubling period of approximately 1.1 years, even though the curve has somewhat flattened in recent years.[1] Figure 4.2 reveals that a paradigm shift took place around 2005, namely the transition from single-core to multi-core architectures. Until 2005, the clock frequency increased continuously over time, while it is stagnating since 2005 at around $2 - 4$ GHz. Since 2005, a continuous increase in the number of cores per chip can be observed with up to 100 on today's hardware. Hence, the increase in floating point performance is no longer driven by an increasing clock frequency (post frequency era), but mainly by parallelism through multi-core architectures and SIMD units of increasing width. The stagnation in clock frequency is related to the power consumption of computer chips, which scales with the third power of the clock frequency (Hager and Wellein 2010). Apart from the economical and environmental pressure on reducing energy consumption, this aspect renders the cooling of the hardware an engineering challenge and limits the clock frequency (energy wall). The power dissipation is around 200 W for current high-end CPU hardware chips. On the scale of the largest supercomputers installed worldwide, power consumption or energy efficiency can be expected to become an increasingly important topic as well. Given that the largest supercomputers already have a power consumption of $20 - 30$ MW (and that the largest power plants, in comparison, deliver a power of several GW), the pressure stemming from economical and environmental aspects on energy-efficient supercomputers becomes evident.

In terms of the roofline performance model, both peak performance and memory bandwidth are quantities of type speed, i.e., they increase over time as progress is made in computer hardware and follow some form of Moore's law. This development is shown in Figure 4.3. An important trend in computer hardware is that peak performance is growing faster than memory bandwidth. While arithmetics and memory have been more "balanced" for former hardware of the single-core era, the gap between floating point performance and memory bandwidth is becoming wider for most recent hardware of the multi-core era. However, a more detailed analysis

[1] See the TOP500 list https://www.top500.org/ for detailed information.

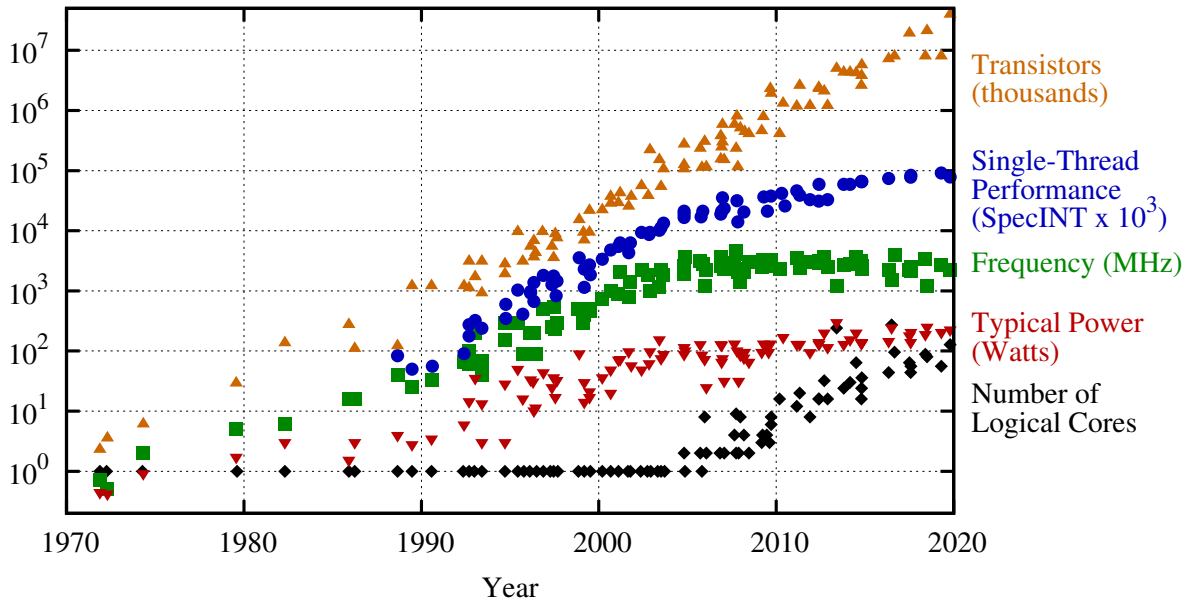


Figure 4.2: Microprocessor trend data over the last 48 years (Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten. New plot and data collected for 2010-2019 by K. Rupp and publicly available at https://github.com/karlrupp/microprocessor-trend-data is reused with minor modifications).

reveals that without SIMD and FMA units, the growth in peak performance is very similar to the growth of the memory bandwidth.[2] In other words, the increasing core count of the multi-core era (at constant clock frequency) is able to compensate the growth in memory bandwidth, while the growing gap is driven by single-core features (SIMD, FMA, or other specialized instructions on future hardware). Figure 4.4 shows the development of the Flop-to-Byte ratio over the last decade and illustrates the paradigm shift from balanced machines ($I_{\mathrm{m}} \approx 1$) to imbalanced machines ($I_{\mathrm{m}} \gg 1$). This trend renders the memory bandwidth the limiting resource for many problems in scientific computing, a situation described as DRAM gap or memory wall (Wilkes 2001). This trend can be expected to have an influence on the algorithm or type of implementation that yields optimal efficiency for a given problem. To give a concrete example in the context of PDE solvers discussed here, this trend favors matrix-free algorithms over matrix-based algorithms, rendering the on-the-fly computation of "matrix-entries" faster (depending on the polynomial degree) than streaming them from main memory in case of sparse matrix-vector products. To take advantage of the growing machine intensity, it is imperative that the implementation exploits the vectorization capabilities of modern hardware. These aspects are discussed in detail in the rest of this chapter. An interesting question is to which extent this trend will continue in the future. For example, the new Nvidia Tesla A100 architecture released in 2020 has a lower (double precision) Flop-to-Byte ratio than its predecessor V100. Finally, it can be expected that the energy wall will have an impact on future hardware developments. To sum up and in order to

[2] Private communication with Jan Eitzinger (RRZE).

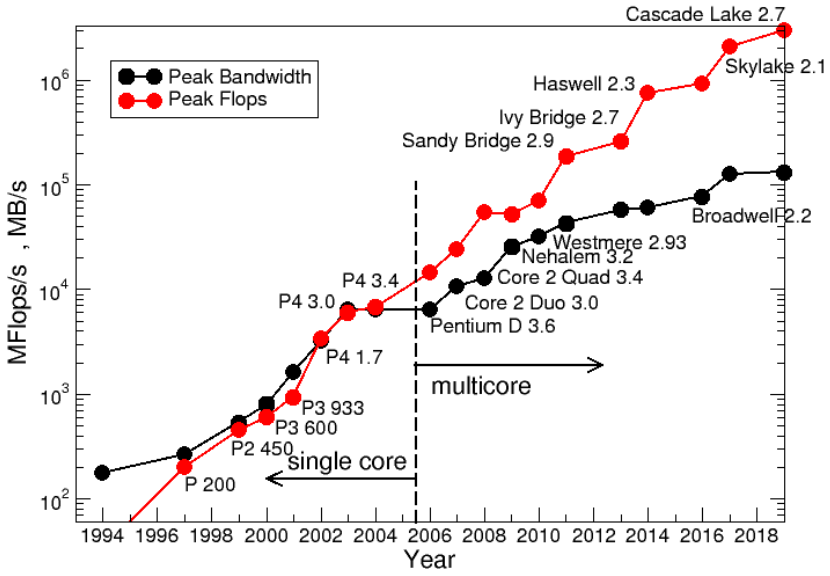


Figure 4.3: Development of peak performance and memory bandwidth over last 25 years for Intel architectures. Plot and data collected by Jan Eitzinger (RRZE) is reused with permission.

clarify notation, the term "modern computer hardware" is used in this work in order to refer to the following hardware characteristics and trends:

- cache-based multi-core CPU architectures with increasing core count per chip,
- vectorization capabilities through registers of increasing SIMD width,
- and improvements in memory bandwidth developing at slower pace, leading to machines of increasing Flop-to-Byte ratio.

Remark 4.1 *Figure 4.4 reveals that the Flop-to-Byte ratio of GPU hardware is not higher than that of CPU hardware. In the literature, GPU implementations are often advertised over CPU implementations for arithmetically intense algorithms such as matrix-free high-order DG discretizations due to computations being cheaper relative to memory transfer on the GPU (Klöckner et al. 2009). This argument is not supported by the data given in Figure 4.4. Nevertheless, the higher peak performance and memory bandwidth of current GPU hardware potentially allows a speed-up by a (small) integer factor when porting a CPU code to the GPU in a given power budget (Kronbichler and Ljungkvist 2019). Note also that peak performance and memory bandwidth are currently growing faster for GPU hardware than for CPU hardware according to the numbers in Ibeid et al. (2020).*

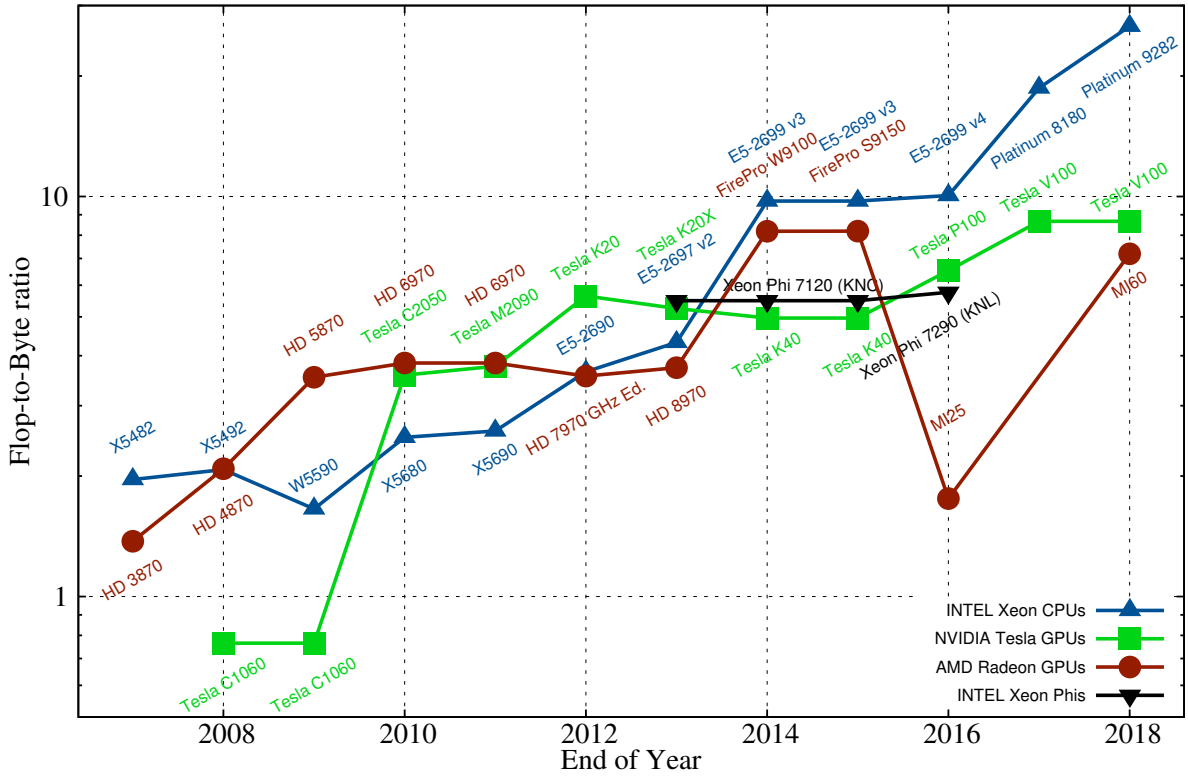


Figure 4.4: Trend in Flop-to-Byte ratio (double precision) over the last decade for CPU and GPU architectures of different manufacturers (data collected by Karl Rupp and publicly available at https://github.com/karlrupp/cpu-gpu-mic-comparison is reused with minor modifications).

4.4 From matrix-based to matrix-free operator evaluation

This section discusses different abstraction levels of finite element operator evaluation, leading to different implementation strategies characterized by different computational complexities in terms of memory requirements and floating point operations. These different implementation strategies can be classified as either matrix-based or matrix-free evaluation techniques. Matrix-based refers to the classical assembly of finite element matrices stored in a sparse matrix data format with the subsequent application of sparse matrix-vector products, and matrix-free refers to on-the-fly operator evaluation without storing matrices explicitly for all elements (but includes the case of storing a matrix for a single element that can be reused for all other elements). The basic methodology is explained by the example of the volume integral of the SIPG discretization of the Laplace operator, mainly for ease of notation due to the simplicity of this operator. Having discussed these different operator evaluation concepts, a generalization to other linear and non-linear operators as well as face integrals in DG is straight-forward given that these more complex operators make use of essentially the same basic ingredients. This section explains why the classical abstraction of using a linear algebra interface with a clear separation of finite element modules and (non-)linear solver modules operating on sparse matrices comes along with a

slow execution of the code for high-polynomial degrees, and explains the advantages and disadvantages of alternative, matrix-free evaluation techniques.

Independently of the implementation technique chosen for a finite element method, integrals in the weak form are typically evaluated by numerical quadrature on the reference element and summation over all elements. The aim is to compute the action of the matrix-vector product $\mathbf{y} = \mathbf{A}\mathbf{u}$ as required for example in iterative solvers. Then, consider the contribution of element e for test function ℓ_i

$$\begin{aligned}
(\mathbf{A}_e\mathbf{u}_e)_i &= (\nabla\ell_i, \nabla u_h)_{\Omega_e} = \int_{\Omega_e} (\nabla_{\boldsymbol{x}}\ell_i)^{\mathsf{T}} \nabla_{\boldsymbol{x}} u_h^e d\boldsymbol{x} \\
&= \int_{\tilde{\Omega}_e} ((\boldsymbol{J}^e)^{-\mathsf{T}} \nabla_{\boldsymbol{\xi}}\ell_i)^{\mathsf{T}} ((\boldsymbol{J}^e)^{-\mathsf{T}} \nabla_{\boldsymbol{\xi}} u_h^e) |\det \boldsymbol{J}^e| d\boldsymbol{\xi} \\
&\approx \sum_q \underbrace{(\nabla_{\boldsymbol{\xi}}\ell_i(\boldsymbol{\xi}_q))^{\mathsf{T}}}_{\left(\mathbf{I}_e^{\mathsf{T}}\right)_{iq}} \underbrace{\left(\boldsymbol{J}_q^e\right)^{-1} (w_q |\det \boldsymbol{J}_q^e|) \left(\boldsymbol{J}_q^e\right)^{-\mathsf{T}}}_{(\mathbf{D}_e)_{qq}} \sum_j \underbrace{\nabla_{\boldsymbol{\xi}}\ell_j(\boldsymbol{\xi}_q)}_{(\mathbf{I}_e)_{qj}} u_j^e \\
&= \left(\mathbf{I}_e^{\mathsf{T}}\mathbf{D}_e\mathbf{I}_e\mathbf{u}_e\right)_i, \forall i = 1, \ldots, (k+1)^d \, .
\end{aligned} \tag{4.6}$$

The integral over the physical domain is first transformed to the reference element, giving rise to geometry terms such as the Jacobian $\boldsymbol{J}^e$. Integration is then performed by Gaussian quadrature, introducing the quadrature weight w_q and replacing the integral by a sum over all quadrature points. The last row shows how the elementwise computation of integrals can be interpreted in terms of an abstract notation that identifies reoccurring building blocks. One of these building blocks is the interpolation operator $\mathbf{I}_e$ that computes the gradient (in reference coordinates) of the solution at all quadrature points by interpolation of the basis functions (using the polynomial expansion of the solution function). The differential operator $\mathbf{D}_e$ applies the PDE operator for all quadrature points and depends on geometric data associated to the current element e for non-Cartesian element geometries. The integration operator $\mathbf{I}_e^{\mathsf{T}}$ multiplies by the gradient of the test function and sums over all quadrature points (=integration). It can be easily seen from equation (4.6) that the integration step is the transpose of the interpolation step. Interpolation and integration are done in reference coordinates and do not depend on the current element e. This property will be exploited for some of the implementation techniques discussed below. Using the notation introduced above, the global operator evaluation can be written as

$$\mathbf{y} = \sum_{e=1}^{N_{\text{el}}} \mathbf{S}_e\mathbf{A}_e\mathbf{u}_e = \sum_{e=1}^{N_{\text{el}}} \mathbf{S}_e\mathbf{A}_e\mathbf{G}_e\mathbf{u} \, . \tag{4.7}$$

The gather operation $\mathbf{G}_e$ extracts the local degrees of freedom associated to element e, $\mathbf{u}_e = \mathbf{G}_e\mathbf{u}$. The scatter operation $\mathbf{S}_e = \mathbf{G}_e^{\mathsf{T}}$ adds contributions of the integral into the global residual vector according to the mapping of local-to-global degrees of freedom. The matrix-vector notation is used for ease of notation only, i.e., these operations are not necessarily implemented as sparse matrix-vector products for reasons of computational efficiency (for example, one can exploit that the degrees of freedom of an element are contiguously in memory in the DG case).

Then, the challenge to obtain a computationally efficient numerical method can be formulated as follows: Which parts of the above algorithm should be precomputed and stored, and which parts should be evaluated and re-computed on-the-fly? Various levels of abstraction leading to different implementation techniques can be distinguished as explained in the following.

Table 4.1: Comparison of operator evaluation costs for matrix-based and matrix-free approaches in terms of data transfer from main memory, arithmetic operations, and computational intensity (setup costs are neglected).

	matrix-based	static condensation / HDG	matrix-free w.o. sum-fac.	matrix-free
operations	$\mathcal{O}\left(k^{2d}\right)$	$\mathcal{O}\left(k^{2(d-1)}\right)$	$\mathcal{O}\left(k^{2d}\right)$	$\mathcal{O}\left(k^{d+1}\right)$
data transfer	$\mathcal{O}\left(k^{2d}\right)$	$\mathcal{O}\left(k^{2(d-1)}\right)$	$\mathcal{O}\left(k^{d}\right)$	$\mathcal{O}\left(k^{d}\right)$
intensity I_c	$\approx 1/8 \ldots 1/6$	$\approx 1/8 \ldots 1/6$	$\mathcal{O}(k^d) \gg 1$	$\mathcal{O}(k) > 1$

4.4.1 Assembling a sparse matrix

The classical low-order finite element procedure is to assemble a sparse matrix $\mathbf{A}$, and subsequently compute the sparse matrix-vector product when required in iterative solvers

$$\mathbf{y} = \mathbf{A}\mathbf{u} \ , \tag{4.8}$$

where the global sparse matrix is obtained by assembly (also termed direct stiffness summation)

$$\mathbf{A} = \sum_{e=1}^{N_\mathrm{el}} \mathbf{S}_e \mathbf{A}_e \mathbf{G}_e \ . \tag{4.9}$$

Assembling the matrix requires $\mathcal{O}(k^{3d})$ operations in case of a naive implementation, i.e., summing k^d quadrature points for k^{2d} matrix entries. By the use of sum-factorization techniques discussed below, assembly costs can be reduced to $\mathcal{O}(k^{2d+1})$. Application of the matrix-vector product requires $\mathcal{O}(k^{2d})$ memory transfer and arithmetic operations, see Table 4.1. Only two floating point operations (one addition and one multiplication) are performed per double-precision matrix entry loaded from memory (assuming that the matrix does not fit into the cache), resulting in a low intensity of $I_\mathrm{c} = 1/4$. Taking into account additional data transfer for indexing and due to non-optimal use of caches, a more realistic estimate of the intensity is a value of $1/8$ to $1/6$, see also Hager and Wellein (2010). Sparse matrix-vector products are therefore clearly memory-bound on current hardware, see Figure 4.5 for an interpretation in terms of the roofline model. Assembly costs are neglected in Table 4.1, but might be performance relevant if frequent re-assembly is necessary.

4.4.2 Assembling elementwise matrices

An alternative is to only store the elementwise matrices $\mathbf{A}_e$ without forming the global sparse matrix $\mathbf{A}$

$$\mathbf{y} = \sum_{e=1}^{N_\mathrm{el}} \mathbf{S}_e \mathbf{A}_e \mathbf{G}_e \mathbf{u} \ . \tag{4.10}$$

This approach also requires $\mathcal{O}(k^{2d})$ operations for evaluation, as well as $\mathcal{O}(k^{2d})$ memory storage and data transfer for generally deformed meshes. Compared to the first approach, this approach

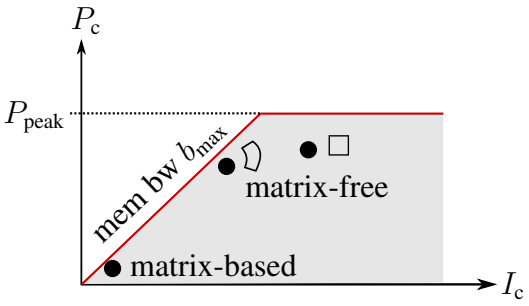


Figure 4.5: Characterization of matrix-based and matrix-free (with sum factorization) algorithms for operator evaluation in terms of the roofline performance model. Schematic illustration of results published in (Kronbichler and Wall 2018, Figure 2), where the different symbols indicate meshes composed of Cartesian and deformed elements.

can have advantages in terms of memory requirements in case of trivial geometries, in the sense that all elements have the same geometry (Jacobian matrix potentially scaled by a simple factor) and the matrix $\mathbf{A}_e$ needs to be stored only once for a single element and might reside in the cache (for moderately large polynomial degrees) when traversing through all elements of the mesh during operator evaluation. For this special case, data transfer reduces to $\mathcal{O}(k^d)$ (streaming of vectors only) so that this approach might be denoted as matrix-free. This approach has been used in Ljungkvist (2014) for uniform meshes composed of tensor-product elements. In case of simplicial elements, the elemental matrices are the same for all elements (up to a scaling factor) if the mapping is linear with straight-sided faces, which is why this level of abstraction is a popular approach for high-order finite element solvers on meshes composed of simplicial elements, see Heinecke et al. (2014), Klöckner et al. (2009). In the DG case, additional matrices are needed for all possible combinations of face orientations. For such "trivial" geometries, the computational intensity is $I_c = \mathcal{O}(k^d)$. This approach has gained interest since it achieves a significant fraction of the peak performance, but the algorithm is clearly compute-bound on current hardware for large k and, therefore, sub-optimal in terms of throughput.

Remark 4.2 *For general finite element spaces including continuous Galerkin methods, it is often argued that using such an element-wise interface with corresponding block data structures can have advantages in terms of faster data access due to a better utilization of cache lines and caches (Anderson et al. 2020, Vos et al. 2010), since the degrees of freedom of one element are stored contiguously in memory (data locality), see also Remark 4.6 related to this topic. Put differently, this perspective gives indications of how to optimally order degrees of freedom in memory for continuous Galerkin discretizations in order to optimally exploit the available memory bandwidth (Moxey et al. 2020b).*

4.4.3 Matrix-free evaluation without sum-factorization

While $\mathbf{A}_e$ is the same for all elements only under certain assumptions on the element geometry, the interpolation operator $\mathbf{I}_e$ operates in reference space and is, therefore, the same for all elements also for deformed elements. This leads to a true matrix-free evaluation technique that can

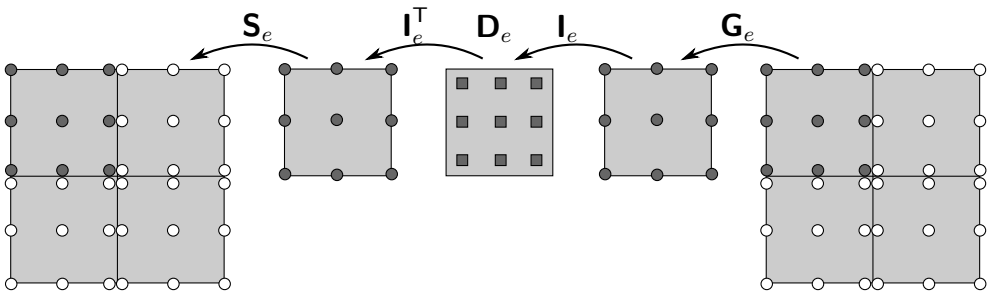


Figure 4.6: Illustration of matrix-free operator evaluation for the computation of volume integrals for $d = 2$ for a discontinuous, nodal basis with degree $k = 2$ and $k + 1 = 3$ interpolation and quadrature points per coordinate direction. Note that this illustration shows the non-vectorized case with the volume integral performed for a single element only.

be written as

$$\mathbf{y} = \sum_{e=1}^{N_\mathrm{el}} \mathbf{S}_e \mathbf{I}_e^\mathsf{T} \mathbf{D}_e \mathbf{I}_e \mathbf{G}_e \mathbf{u} \,. \tag{4.11}$$

This variant stores the interpolation operator $\mathbf{I}_e \in \mathbb{R}^{d(k+1)^d \times (k+1)^d}$ explicitly. Compute $\forall q$ (considering a three-dimensional domain with $d = 3$)

$$\sum_{i=1}^{N_\mathrm{DoFs}^e} (\mathbf{I}_e)_{qi} (\mathbf{u}_e)_i \,, \text{ where } (\mathbf{I}_e)_{qi} = \nabla_{\boldsymbol{\xi}} \ell_i(\boldsymbol{\xi}_q) = \begin{pmatrix} \ell_{i_1,\xi_1}(\xi_{1,q_1}) \ell_{i_2}(\xi_{2,q_2}) \ell_{i_3}(\xi_{3,q_3}) \\ \ell_{i_1}(\xi_{1,q_1}) \ell_{i_2,\xi_2}(\xi_{2,q_2}) \ell_{i_3}(\xi_{3,q_3}) \\ \ell_{i_1}(\xi_{1,q_1}) \ell_{i_2}(\xi_{2,q_2}) \ell_{i_3,\xi_3}(\xi_{3,q_3}) \end{pmatrix} \,. \tag{4.12}$$

Since the interpolation operator is the same for all elements also for deformed geometries, it needs to be stored only once. Assuming that the interpolation operator resides in the cache, this approach reduces the data transfer to $\mathcal{O}(k^d)$. The memory requirements of $\mathcal{O}(k^d)$ originate from solution vectors proportional to the degree of freedoms and geometry data proportional to the number of quadrature points. The operation count of this approach is $\mathcal{O}(k^{2d})$ with higher proportionality constant than for the matrix-based variants discussed above, see the third column in Table 4.1. The resulting algorithm can be classified as compute-bound on current hardware for medium and large polynomial degrees. Such an approach typically achieves a high fraction of the peak performance for large polynomial degrees, but is sub-optimal in terms of throughput due to the complexity of $\mathcal{O}(k^{2d})$. This approach is the level of abstraction chosen in `PyFR` (Witherden et al. 2014). The fact that results are only shown for $k \leq 4$ in that work points to the limitations of this approach for large k on current hardware. Nevertheless, for element types that do not support sum-factorization (see below), this strategy might be a viable alternative and could gain further interest in case that the Flop-to-Byte ratio continues to grow in the future. An illustration of the matrix-free evaluation process is provided in Figure 4.6.

4.4.4 Matrix-free evaluation with sum-factorization

In equation (4.12), the tensor-product structure of the shape functions and the quadrature rule can be exploited for the interpolation and integration steps

$$\sum_{i=1}^{N_{\mathrm{DoFs}}^e} \nabla_{\boldsymbol{\xi}} \ell_i(\boldsymbol{\xi}_q) \left(\mathbf{u}_e\right)_i = \sum_{i_1=1}^{k+1} \begin{pmatrix} \ell_{i_1,\xi_1}(\xi_{1,q_1}) \\ \ell_{i_1}(\xi_{1,q_1}) \\ \ell_{i_1}(\xi_{1,q_1}) \end{pmatrix} \sum_{i_2=1}^{k+1} \begin{pmatrix} \ell_{i_2}(\xi_{2,q_2}) \\ \ell_{i_2,\xi_2}(\xi_{2,q_2}) \\ \ell_{i_2}(\xi_{2,q_2}) \end{pmatrix} \sum_{i_3=1}^{k+1} \begin{pmatrix} \ell_{i_3}(\xi_{3,q_3}) \\ \ell_{i_3}(\xi_{3,q_3}) \\ \ell_{i_3,\xi_3}(\xi_{3,q_3}) \end{pmatrix} \left(\mathbf{u}_e\right)_{i_1 i_2 i_3} \ . \tag{4.13}$$

This optimization technique is called sum-factorization and replaces the sum over all nodes i by d sums over the one-dimensional nodes $i_1, \ldots, i_d$ for each of the d components of the gradient. Interpolation in d space dimensions is decomposed into a sequence of 1D interpolations for each coordinate direction. Applying d one-dimensional interpolation kernels for d gradients gives rise to d^2 kernels. Each of these kernels can be interpreted as a dense matrix-matrix product of either the 1D shape values matrix or shape derivative matrix with the matrix of solution coefficients, which is a reshuffling of the solution vector $\mathbf{u}_e$ such that one index forms the rows and the other two indices the columns of that matrix. This amounts to $2(k+1)^2$ operations (additions and multiplications) to be performed for all $(k+1)^{d-1}$ 1D stripes of the remaining $d-1$ dimensions. All in all, this leads to $2d^2(k+1)^{d+1}$ operations compared to $2d(k+1)^{2d}$ for a naive evaluation. Below, advanced techniques are described that allow to further reduce operation counts, e.g., only $2d$ instead of d^2 kernels for the interpolation of the gradient. Data transfer from main memory such as solution vectors and geometric data stored in quadrature points scales as $(k+1)^d$, see the last column in Table 4.1. Matrix-free operator evaluation with sum-factorization is the state-of-the-art implementation technique for tensor-product elements, see for example Kronbichler and Kormann (2012, 2019), Müthing et al. (2017). Depending on the polynomial degree, the arithmetic intensity operates somewhere around the machine intensity for current hardware, which is illustrated in Figure 4.5. Put differently, current hardware does not offer enough floating point compute capabilities to render sum-factorization superfluous. Hence, when talking about matrix-free implementations in the rest of this work, this shall imply that sum-factorization is exploited without mentioning this explicitly. For simplicial elements, this technique is currently used rather rarely, since sum-factorization is more difficult in this case with overhead due to the more complicated indexing. A recent work by Moxey et al. (2020a) explores sum-factorized matrix-free implementations for non tensor-product elements. The following subsections summarize a list of design choices optimizing node-level performance for the matrix-free implementation used in this work. A detailed discussion of these aspects can be found in Kronbichler and Kormann (2012, 2019).

4.4.4.1 Minimizing arithmetic operations and optimizing sum-factorization kernels

Although modern hardware can sustain more Flops than Bytes can be transferred from memory, operations are not for free and need to be reduced to a mininum in order to achieve optimal performance on current hardware, a main result of the work by Kronbichler and Kormann (2019). Regarding the evaluation of the gradient of the solution at all quadrature points through sum-factorization, the operations can be reduced from d^2 to $2d$ kernels by first interpolating into a col-

location basis (d kernels) and subsequently evaluating the gradient in the collocation basis (another d kernels), a technique called collocation derivative (Kronbichler and Kormann 2019). Another optimization technique reducing the number of operations for the one-dimensional kernels exploits the symmetry of the one-dimensional shape functions and is called even-odd decomposition (Kopriva 2009). The asymptotic complexity remains unchanged and is still $\mathcal{O}(k^{d+1})$. In the current `deal.II` implementation, the length of the innermost loops over 1D nodes and quadrature points is a compile time constant, realized through a `C++` template parameter and allowing the compiler to produce optimal code, e.g., by loop unrolling.

4.4.4.2 Evaluation of geometry terms

Regarding the differential operator D_e applied on the quadrature point level, several abstraction levels can be distinguished in terms of which geometry terms and which operator-specific terms such as variable coefficients should be stored, or recomputed on-the-fly. For the special case of affine element geometries, a single Jacobian $\boldsymbol{J}^e$ can be used at all quadrature points of an element, which is exploited by the present implementation to minimize memory access. Taking the example of the Laplace operator discussed above, a separate Jacobian $\boldsymbol{J}_q^e$ is precomputed for each quadrature point for deformed geometries and stored as $\left(\boldsymbol{J}_q^e\right)^{-\mathsf{T}}$, which is then accessed during the operator evaluation and represents the main memory traffic (together with the determinant of the Jacobian, this amounts to d^2+1 double precision values per quadrature point). For the computation of face integrals (see Section 4.4.5), the quantity $\boldsymbol{n}^{\mathsf{T}}\left(\boldsymbol{J}_q^e\right)^{-\mathsf{T}}$ is pre-computed at each quadrature point for both sides of a face, which amounts to $2d$ values per face quadrature point (in $d-1$ dimensions). For such an implementation (as used in this work), the arithmetic intensity differs significantly between Cartesian and deformed elements, as illustrated in Figure 4.5. Since the arithmetic intensity of matrix-free implementations is in a similar range as the Flop-to-Byte ratio of modern hardware (Kronbichler and Kormann 2019, Kronbichler et al. 2019, Müthing et al. 2017), it is non-trivial to decide which compromise between increase in memory transfer or floating point operations shows the best efficiency, see for example the work by Remacle et al. (2016). The work by Müller et al. (2019) concludes that memory-bound spectral element codes that stream metric terms from memory could be accelerated by recomputing metric terms on-the-fly. These aspects are discussed in more detail in Kronbichler and Kormann (2019), where several variants for the treatment of geometry terms are presented and compared, and in Davydov et al. (2020) for nonlinear elasticity problems, where operations at quadrature points are more complex compared to a simple Poisson problem due to the material model used in that work. The arithmetic intensity of the matrix-free approach in relation to the current hardware landscape requires a PDE or application expert to have profound knowledge about matrix-free techniques and low-level code optimization to identify the optimal implementation strategy. This inter-disciplinarity introduces challenges, e.g., how to combine the separation-of-concerns principle or black-box concept with optimal implementations. While storing certain quantities that are more expensive to recompute appears to be a good compromise on current hardware in order to balance operations and memory access, a shift towards recomputing more and more quantities could take place on future hardware if the Flop-to-Byte ratio continues to grow.

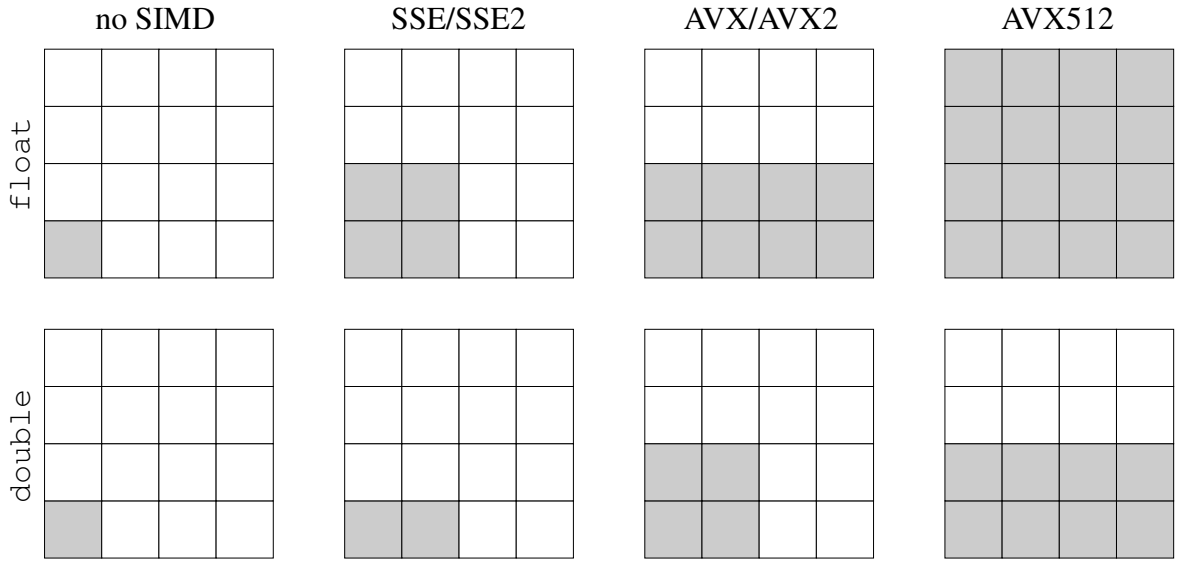


Figure 4.7: Illustration of vectorization over elements for single-precision (top row) and double-precision (bottom row) computations and for different instruction set extensions in Intel language (columns).

4.4.4.3 Vectorization over elements and faces

The matrix-free operator evaluation performs the same operations for all elements, the only difference is that integrals over different elements operate on different parts of the solution vector $\mathbf{u}$ and the geometry information $\boldsymbol{J}_q^e$ has to be stored and loaded separately for each element in case of deformed element geometries. In order to exploit the single-instruction-multiple-data (SIMD) vectorization capabilities of modern hardware with wide SIMD units, the present implementation groups together several elements or faces and performs the integrals in the weak form concurrently for this batch of elements or faces. This technique has first been proposed in Kronbichler and Kormann (2012), and has been adopted in other frameworks in later works by Moxey et al. (2020a), Sun et al. (2020), where this technique is denoted as cross-element vectorization. The basic data type for the operations in the matrix-free evaluation process is therefore `VectorizedArray<Number>`, with `Number` being a template for a `C++` data type such as `double` or `float`, currently realized as a wrapper around intrinsics in `deal.II`. For CPU hardware used in the present work with support for AVX2 or AVX512, vectorization is done over 4 or 8 elements/faces in double precision and 8 or 16 in single precision, respectively. An illustration is given in Figure 4.7. Note that the data might need to be reshuffled when accessing vector data and before the computations from the array-of-struct (array of elements with struct of degrees of freedom of type `Number`) memory layout into an array-of-struct-of-array (each item of the struct is of type `VectorizedArray<Number>`) layout. For meshes with the number of elements/faces not being a multiple of the vectorization width, parts of the vectorized array remain empty for these corner cases. While this vectorization strategy appears to be the most efficient one on current hardware (Kronbichler and Kormann 2019), it also comes along with disadvantages. For example, cross-element vectorization increases pressure on the caches due to

larger temporary data arrays so that the critical polynomial degree at which caches are exhausted for the element-wise integrals is reduced by this technique. According to the results in Kronbichler and Kormann (2019), this strategy nevertheless pays off compared to intra-element vectorization. Another disadvantage is related to strong scalability, since the coarser granularity of the cross-element vectorized case implies that the strong-scaling limit is reached once the number of elements per core is lower than the SIMD width. Also, Gauss–Seidel iteration techniques for preconditioning depend on how the elements are grouped together into batches. Finally, adaptively refined meshes are less homogeneous and might pose restrictions regarding which elements can be processed at once, causing SIMD arrays to be filled only partially. Other vectorization strategies are chosen in Müthing et al. (2017), exploiting that the solution and its gradients fill the SIMD array in case of three space dimensions ($d = 3$), double-precision computations, and AVX2 instruction set extensions with a register width of 256 bit. The study by Kempf et al. (2020) aims at overcoming the limitations of this special vectorization strategy and proposes more flexibility w.r.t. vectorization in a code-generation context.

4.4.4.4 Complexity and throughput

For volume integrals, the number of operations scales as $\mathcal{O}(k^{d+1})$, i.e., the complexity is linear in k per degree of freedom. Similarly, face integrals have a complexity of $\mathcal{O}(k^d)$ due to the lower dimensionality of faces, $d_f = d-1$. Solution coefficients have to be stored per degree of freedom and geometry information per quadrature point. Accordingly, the memory requirements and data transfers to/from main memory scale as $\mathcal{O}(k^d)$, i.e., the complexity is constant per degree of freedom. As a result, it is unclear whether the overall efficiency of the implementation shows a complexity that is constant or linear in k. The behavior depends on the relative costs of volume integrals compared to face integrals as well as the hardware under consideration, i.e., whether the method can be characterized as compute-bound or memory-bound. Overall, matrix-free operator evaluation has an intensity comparable to what current hardware offers. An important result from Kronbichler and Kormann (2019) is that the present matrix-free implementation tends to become memory-bound when implemented with a minimum of arithmetic operations on modern hardware with high Flop-to-Byte ratios. In practice, it is often observed that the throughput of the present matrix-free implementation measured in degrees of freedom per second depends only mildly on the polyomial degree, suggesting an almost constant complexity up to moderately high polynomial degrees, see also the results in Section 4.5. The fact that the observed complexity is better than the theoretical complexity of volume integrals can be explained by face integrals and data access (both showing constant complexity per unknown), which are performance relevant for moderately high polynomial degrees, see also Kronbichler and Kormann (2019), Müthing et al. (2017).

Remark 4.3 *The rapid evolution in hardware and – related to this paradigm shift – the uncertainty regarding optimal numerical algorithms manifests itself in a common mis-communication that can be observed in the literature, see for example Heinecke et al. (2014), Kempf et al. (2020), Müthing et al. (2017), Sun et al. (2020), Witherden et al. (2014), namely that an algorithm should have a high Flop-to-Byte ratio to be computationally efficient on modern hardware and that memory-bound algorithms are per-se blamed inefficient. The argumentation is skewed in the sense that mainly the last row of Table 4.1 is considered as performance-relevant. However,*

arithmetic intensity is a relative quantity that does not contain information about the complexity of the algorithm. The aspect that matrix-free operator evaluation with sum-factorization shows the best complexity on general geometries both in terms of operations and memory access is often under-represented. Moreover, this also originates from a perspective that considers algorithms as high-performance realizations if they achieve a significant fraction of the peak performance, and computer science awards such as the Gordon Bell Prize that award preferably this artificial metric rather than the actual speed of the implementation (Bell et al. 2017). Other examples are the well-known LINPACK benchmark and the TOP500 list focusing on Flops rather than memory bandwidth, although data access is considered the most important performance-limiting factor in HPC (Hager and Wellein 2010). A more generic and balanced notion of high-performance computing is given in Hager and Wellein (2010), where high-performance computing is understood as optimizing throughput by removing bottlenecks. A good example in the context of spectral element (collocation-type) discretizations of the compressible Navier–Stokes equations is the work by Müller et al. (2019). Through performance optimizations, the authors obtain a memory-bound algorithm, and conclude that further performance improvements could be possible by reducing memory transfer via an on-the-fly computation of metric terms, see also Section 4.4.4.2. If the operation counts of a compute-bound algorithm can be reduced so that it becomes increasingly memory-bound, the numerical method becomes more efficient overall in terms of the throughput metric (time-to-solution). In general, the overall goal is (i) to minimize both arithmetic operations and memory access, and (ii) to balance operations and memory access in a way that the throughput is maximized, which might lead to different design choices and algorithms depending on the hardware. Regarding the first aspect, a matrix-free approach with optimal complexity outperforms a matrix-based approach in the limit $k \to \infty$ indepedently of the Flop-to-Byte ratio of the hardware, since such an algorithm reduces both memory transfer and operations. In terms of absolute numbers, the proportionality constant for the $\mathcal{O}(k^{d+1})$ arithmetic operations is such that matrix-free evaluation with sum-factorization requires less operations than sparse matrix-vector products for polynomial degrees of $k = 2$ to 3 and higher in three space dimensions for DG (Kronbichler et al. 2017). Regarding the second aspect, high-order discretizations realized with optimal complexity have a Flop-to-Byte ratio that fits well to modern hardware (Kronbichler and Kormann 2019, Kronbichler et al. 2019, Müthing et al. 2017), which is why such algorithms have seen a significant speed-up over the last years, while the single core performance of classical matrix-based approaches has not improved significantly, see the numerical results for various processor generations shown in Arndt et al. (2020b), Kronbichler and Allalen (2018). For this reason, matrix-free operator evaluation with sum-factorization has been shown to be faster than sparse matrix-vector products already for polynomial degree $k = 2$ on modern hardware (Kronbichler and Kormann 2012, 2019, May et al. 2014). This break-even point became continuously smaller over the last years (consider the publications by Cantwell et al. (2011), Kronbichler and Kormann (2012, 2019), May et al. (2014), Vos et al. (2010) in chronological order), due to developments in computer hardware, but also due to algorithmic improvements of matrix-free techniques that minimize operation counts or make optimal use of compute resources through vectorization (Kronbichler and Kormann 2019). In contrast, matrix-free algorithms with sub-optimal computational complexity (Heinecke et al. 2014, Witherden et al. 2014) become prohibitively expensive for large polynomial degrees on current hardware, explaining why results are typically not shown for polynomial degrees beyond $k = 3$ or 4 in three space dimensions in the literature.

Algorithm 4.1 Matrix-free operator evaluation with compact face integrals

for $e = 1, \dots, N_{\text{el}}$ **do** ▷ volume integrals
  $\mathsf{I}_e\mathsf{G}_e$: gather solution coefficients $\mathbf{u}_e$ and interpolate solution (or its gradient)
  D_e: evaluate differential operator for all volume q-points
  $\mathsf{S}_e\mathsf{I}_e^\mathsf{T}$: test by test function (or its gradient), sum over q-points, and scatter
end for
for $f = 1, \dots, N_{\text{faces,int}}$ **do** ▷ interior face integrals
  $\mathsf{I}_f\mathsf{G}_f$: gather solution coefficients for both sides and interpolate solution (or its gradient)
  D_f: compute numerical flux for all face q-points, apply geometry terms
  $\mathsf{S}_f\mathsf{I}_f^\mathsf{T}$: test by test function (or its gradient) for both sides, sum over q-points, and scatter
end for
for $f = N_{\text{faces,int}} + 1, \dots, N_{\text{faces}}$ **do** ▷ boundary face integrals
  $\mathsf{I}_f\mathsf{G}_f$: gather interior solution coefficients and interpolate solution (or its gradient)
  $\forall q$: compute exterior states $\{u^+, \nabla u^+ \cdot \boldsymbol{n}\}(\boldsymbol{\xi}_q) = f\left(\{u^-, \nabla u^- \cdot \boldsymbol{n}\}(\boldsymbol{\xi}_q), g_q, h_q\right)$
  D_f: compute numerical flux for all face q-points, apply geometry terms
  $\mathsf{S}_f\mathsf{I}_f^\mathsf{T}$: test by test function (or its gradient), sum over q-points, and scatter
end for

4.4.4.5 Single precision for increased throughput

The matrix-free algorithm outlined above is perfectly suited for single-precision computations where the number of elements grouped into one SIMD batch is twice as large as compared to double-precision computations, which allows to speed up computations by a factor of up to 2, see for example Kronbichler et al. (2017). This is possible due to the fact that the amount of data transferred from main memory reduces by a factor of two in case of single precision (allowing twice the throughput in terms of elements processed per time from the point of view of memory bandwidth), and the vectorization strategy with explicit vectorization over elements/faces also allows twice the throughput in terms of arithmetics. While double-precision computations are often desirable in terms of accuracy for PDE solvers, this technique can successfully be exploited in preconditioners such as multigrid, see Chapter 5.

4.4.5 Face integrals and discontinuous Galerkin methods

Face integrals are amenable to matrix-free operator evaluation as well, with the difference that integrals are computed over a $d-1$ dimensional domain, that the geometric terms arising from the transformation of the integral from physical space to reference space are different, and that the solution or its gradient has to be interpolated from the element interior onto the faces in order to evaluate numerical fluxes in the face quadrature points. Again, sum-factorization can be used as for the volume integrals in order to devise algorithms of optimal computational complexity, requiring $\mathcal{O}(k \cdot k^{d-1})$ operations for the interpolation onto the face and $\mathcal{O}(k^2 \cdot k^{d-1-1})$ operations for the subsequent interpolation within the face. An important question is how to organize the loop over faces, where two options are discussed here:

1. Compact face integrals (see Algorithm 4.1): Face integrals are organized independently of volume integrals with separate loops over elements (volume integrals), interior faces,

Algorithm 4.2 Matrix-free operator evaluation with element-wise face integrals

for $e = 1, \dots, N_{\text{el}}$ **do** ▷ volume and face integrals
  G_e: gather solution coefficients $\mathbf{u}_e$
  I_e: interpolate solution (or its gradient) into all volume q-points
  D_e: evaluate differential operator for all volume q-points
  I_e^T: test by test function (or its gradient) and sum over q-points
  for $f = 1, \dots, N_{\text{faces},e} = 2d$ **do** ▷ face integrals
    I_{f,e^-}: interpolate solution (or its gradient) into the q-points of face f
    if interior face **then**
      $\mathsf{I}_{f,e^+}\mathsf{G}_{f,e^+}$: gather solution coefficients $\mathbf{u}_{e^+}$ of neighbor and interpolate
    else
      $\forall q$: compute exterior states $\{u^+, \nabla u^+ \cdot \boldsymbol{n}\}(\boldsymbol{\xi}_q) = f\left(\{u^-, \nabla u^- \cdot \boldsymbol{n}\}(\boldsymbol{\xi}_q), g_q, h_q\right)$
    end if
    $\mathsf{D}_{f,e}$: compute numerical flux for all face q-points, apply geometry terms
    $\mathsf{I}_{f,e}^\mathsf{T}$: test by test function (or its gradient) and sum over q-points
    $\mathsf{S}_{f,e}$: accumulate into element-local data structures
  end for
  S_e: write element-local data into global DoF-vector $\mathbf{u}$
end for

and boundary faces. The computation of face integrals should happen in temporal proximity to volume integrals for improved data locality and cache reuse. This is realized by interleaving volume and face integrals (unlike the global for-loops used in Algorithm 4.1). For this option, numerical fluxes are computed only once on interior faces and tested simultaneously with test functions from both elements adjacent to a face. This approach is generically written as

$$\mathsf{A}\mathbf{u} = \sum_{e=1}^{N_{\text{el}}} \mathsf{S}_e \mathsf{I}_e^\mathsf{T} \mathsf{D}_e \mathsf{I}_e \mathsf{G}_e \mathbf{u} + \sum_{f=1}^{N_{\text{faces,int}}} \mathsf{S}_f \mathsf{I}_f^\mathsf{T} \mathsf{D}_f \mathsf{I}_f \mathsf{G}_f \mathbf{u} + \sum_{f=N_{\text{faces,int}}+1}^{N_{\text{faces}}} \mathsf{S}_f \mathsf{I}_f^\mathsf{T} \mathsf{D}_f \mathsf{I}_f \mathsf{G}_f \mathbf{u} \,. \quad (4.14)$$

Similar to the evaluation of volume integrals, G_f extracts the relevant degrees of freedom of the two elements e^-, e^+ required for the computation of the face integral over a face $f = \partial\Omega_{e^-} \cap \partial\Omega_{e^+}$, i.e., $\left(\mathbf{u}_{e^-}^\mathsf{T}, \mathbf{u}_{e^+}^\mathsf{T}\right)^\mathsf{T} = \mathsf{G}_f \mathbf{u}$, and $\mathsf{S}_f = \mathsf{G}_f^\mathsf{T}$ is a scatter operation that adds contributions to both elements adjacent to a face.

2. Element-wise face integrals (see Algorithm 4.2): Face integrals are understood as operations belonging to a particular element in addition to the volume integral performed on this element. A consequence of this approach is that numerical fluxes are computed twice, whenever one element requires this flux for testing with its own test functions. This approach is generically written as

$$\mathsf{A}\mathbf{u} = \sum_{e=1}^{N_{\text{el}}} \mathsf{S}_e \left(\mathsf{I}_e^\mathsf{T} \mathsf{D}_e \mathsf{I}_e \mathsf{G}_e \mathbf{u} + \sum_{f=1}^{N_{\text{faces},e}} \mathsf{S}_{f,e} \mathsf{I}_{f,e}^\mathsf{T} \mathsf{D}_{f,e} \mathsf{I}_{f,(e^-,e^+)} \mathsf{G}_{f,(e^-,e^+)} \mathbf{u} \right) \,. \quad (4.15)$$

The scatter operation $\mathsf{S}_{f,e}$ accumulates into the element-local data structures, while S_e finally scatters into the global DoF-vector once the computations for element e are completed. Note that gathering and interpolating data has to be done for both elements e^- and e^+ adjacent to f, while the flux computation, testing, and scattering is done for element $e = e^-$ only. Computing the numerical flux once more for element e^+ increases the arithmetic operations for this strategy. At the same time, results already computed for volume integrals can be reused for the computation of face integrals and reduce operation counts. Apart from the aspect of operation counts, this approach is characterized by a favorable data access pattern similar to finite difference methods. Note also that this is the natural formulation for block-Jacobi and block-Gauss–Seidel preconditioners where block refers to the degrees of freedom of one element.

According to the above discussion, it is non-trivial to answer a priori which strategy is computationally more efficient. Different strategies for the computation of face integrals have been analyzed in Kronbichler and Kormann (2019), Kronbichler and Allalen (2018), Kronbichler et al. (2017) in terms of throughput. The favorable data access pattern of element-wise face integrals renders this approach more efficient than separate face loops for operators such as the SIPG discretization of the Laplace operator or a Lax–Friedrichs discretization of advection terms according to Kronbichler and Kormann (2019). For operators with complex numerical flux computations requiring more arithmetic work, the approach with separate face loops might nevertheless be faster, a topic that has not been explored in detail to date. Since the element-wise computation of face integrals has not been available in `deal.II` by the time of writing, the first option with separate face integrals is used in the course of this thesis.

Remark 4.4 *For general operators, the interpolation operators might not be the same for the solution function and the test function, i.e., the symmetry of the volume integral of the Laplace operator w.r.t. solution and test functions is a special case. This complexity is hidden in the above equations* (4.14) *and* (4.15) *by defining the interpolation operator* I *and the quadrature point operator* D *accordingly, i.e., the interpolation operator is formally extended to always include the computation of values and gradients and the quadrature point operator might no longer be a diagonal operator. Of course, this is done only for ease of notation, since such a "padding with zeros" would increase operation counts and reduce computational efficiency. An alternative formulation would be to use different and operator-dependent interpolation operators for the solution function and the test function according to the particular operator of interest. Interpolation operators for face integrals using sum-factorization are detailed in Kronbichler and Kormann (2019). For ease of notation, operator evaluation is written as a sequence of matrix-vector products, but optimal-complexity algorithms such as sum-factorization are used, e.g., to apply* I *to a vector.*

Remark 4.5 *As already mentioned, the variant with compact face integrals interleaves volume and face integrals for improved data locality. An alternative strategy for the computation of face integrals is a separate loop for face integrals with separate global data structures for faces and is used, e.g., in Hindenlang et al. (2012), Klöckner et al. (2009). The study by Kronbichler and Kormann (2019), where this variant is called "global trace storage", identified this approach as computationally least efficient due to increased memory transfer with several sweeps through global vectors.*

Remark 4.6 *The block data structure in the case of discontinuous finite element spaces is often used to advertise discontinuous over continuous Galerkin methods (given that DG methods naturally have this block data structure while data access might be irregular for continuous Galerkin discretizations due to degrees of freedom shared between elements). The performance numbers shown in Kronbichler and Wall (2018) do not justify the use of DG methods for reasons of improved data locality. For example, the total number of degrees of freedom increases for DG and additional face integrals take a significant share of the overall computational costs. Moreover, the quadrature-point based geometry data on generally deformed elements (with regular data access pattern also for continuous discretizations) amounts to much more data than the solution vectors as argued in Ljungkvist (2017). The theoretical performance model in Müller et al. (2019) for a continuous Galerkin discretization does not favor a DG storage scheme over the CG storage scheme with noncontiguous memory access in terms of optimal runtime.*

4.4.6 Other matrix-free techniques optimized for trivial geometries

For Cartesian meshes, there exist other specialized techniques that can be seen as a combination of the approach from Section 4.4.2 using elementwise matrices and the approach from Section 4.4.4 using a matrix-free evaluation with sum-factorization. Under the assumption of Cartesian meshes and for certain PDE operators, the operator is separable and the elementwise matrix can be factorized into a tensor-product structure. The whole operator can then be evaluated by applying one sweep of one-dimensional sum-factorization kernels, instead of two sweeps for the interpolation and integration steps of the matrix-free operator evaluation with sum-factorization. Accordingly, this approach has the same complexity in terms of memory transfer ($\mathcal{O}(k^d)$) and operation counts ($\mathcal{O}(k^{d+1})$), but reduces the absolute number of arithmetic operations. Combining static condensation with tensor-product methods, a linear complexity of $\mathcal{O}(k^d)$ in operation counts can be achieved (Huismann et al. 2017, 2019). Due to the restrictions of this approach to Cartesian meshes, it is not discussed in more in detail in this work.

4.4.7 A note on hybridizable discontinuous Galerkin methods

This section briefly characterizes hybridizable discontinuous Galerkin (HDG) methods, see for example Kirby et al. (2012), Yakovlev et al. (2016), in terms of the above categorization of implementation strategies. HDG methods aim at reducing the size of the global matrix system. This is achieved by introducing additional unknowns called trace variables at the interface between elements. Inner degrees of freedom are then eliminated by static condensation and the global problem is formulated in terms of the degrees of freedom on the mesh skeleton only. This results in a matrix-based technique, and Table 4.1 lists the complexity of this approach compared to other evaluation techniques. From these estimates, it can be expected that the HDG approach is efficient in two space dimensions, $d = 2$. In three space dimensions, $d = 3$, the HDG approach has the same complexity as matrix-free operator evaluation in terms of operation counts, $\mathcal{O}(k^4)$, but a higher complexity of $\mathcal{O}(k^4)$ in terms of data transfer from main memory, compared to $\mathcal{O}(k^3)$ for matrix-free operator evaluation. From these theoretical estimates it is difficult to answer whether the HDG approach is competitive to matrix-free operator evaluation. A look at the computational intensity of the HDG approach reveals that the $\mathcal{O}(k^4)$ data transfer is the performance-limiting factor of HDG methods on hardware that offers Flop-to-Byte ratios

of one and larger. A recent performance study by Kronbichler and Wall (2018) for a Poisson model problem reveals that matrix-free operator evaluation is significantly faster than the HDG method on modern CPU hardware when the comparison is made for a setup in which the different approaches are sufficiently optimized to allow general conclusions regarding their relative efficiency. To understand this behavior, it is necessary to internalize the hardware developments that took place over the last two decades, most importantly the trend of increasing Flop-to-Byte ratio according to Figure 4.4. Moreover, the setup costs of HDG exhibit a complexity in k that is higher than the operator evaluation costs listed in Table 4.1, and might not be negligible especially for problems that require a frequent re-assembly of operators. An interesting point of view is provided in Kronbichler and Wall (2018) in the sense that HDG methods would need to adapt concepts from matrix-free operator evaluation in order to bring the performance of this method closer to fast matrix-free techniques.

4.5 Numerical results

This section presents numerical results characterizing the efficiency of the matrix-free implementation with sum-factorization used in this work for on-the-fly operator evaluation. All operators listed in Section 4.5.1 refer to discontinuous Galerkin discretizations. Section 4.5.2 describes the experimental setup. Sections 4.5.3 details how the throughput behaves as a function of the problem size by the example of the scalar Laplace operator. Section 4.5.4 analyzes the throughput as a function of the polynomial degree, considering first the scalar Laplace operator and subsequently also other incompressible Navier–Stokes operators. To complement these results, Section 4.5.5 shows results of a roofline analysis for these operators.

4.5.1 Operators

The operators selected in this chapter are the pressure Poisson operator

$$\mathsf{y} = \mathsf{L}\mathsf{p} \,, \tag{4.16}$$

the inverse velocity mass matrix operator

$$\mathsf{y} = \mathsf{M}^{-1}\mathsf{u} \,, \tag{4.17}$$

the penalty operator composed of the mass operator and the divergence and continuity penalty operators

$$\mathsf{y} = (\mathsf{M} + \Delta t_n \mathsf{A}_\mathsf{D} + \Delta t_n \mathsf{A}_\mathsf{C})\,\mathsf{u} \,, \tag{4.18}$$

the Helmholtz-like operator using the Laplace formulation of the viscous term

$$\mathsf{y} = \left(\frac{\gamma_0^n}{\Delta t_n}\mathsf{M} + \mathsf{V}\right)\mathsf{u} \,, \tag{4.19}$$

the nonlinear convective operator using the conservative formulation

$$\mathsf{y} = \mathsf{c}\,(\mathsf{u}) \,\,, \tag{4.20}$$

Table 4.2: Performance specifications for Intel Skylake system of SuperMUC-NG at LRZ in Garching, Germany. Assuming an AVX512 clock frequency of 2.3 GHz, 48 cores per node with 2 FMA units each result in a peak floating point performance of 3.5 TFlop/s per node.

Processor		Memory and Caches	
Processor type	Intel Xeon Platinum 8174	Memory per node (thin/fat)	96/768 GByte
Frequency	2.3 GHz	Theoretical memory bandwidth	256 GByte/s
Cores per node	48 (2 sockets, 24 cores each)	STREAM memory bandwidth	205 GByte/s
SIMD width	512 bit (AVX512)	Cache size (L2 + L3) per node	$2 \cdot 57$ MByte

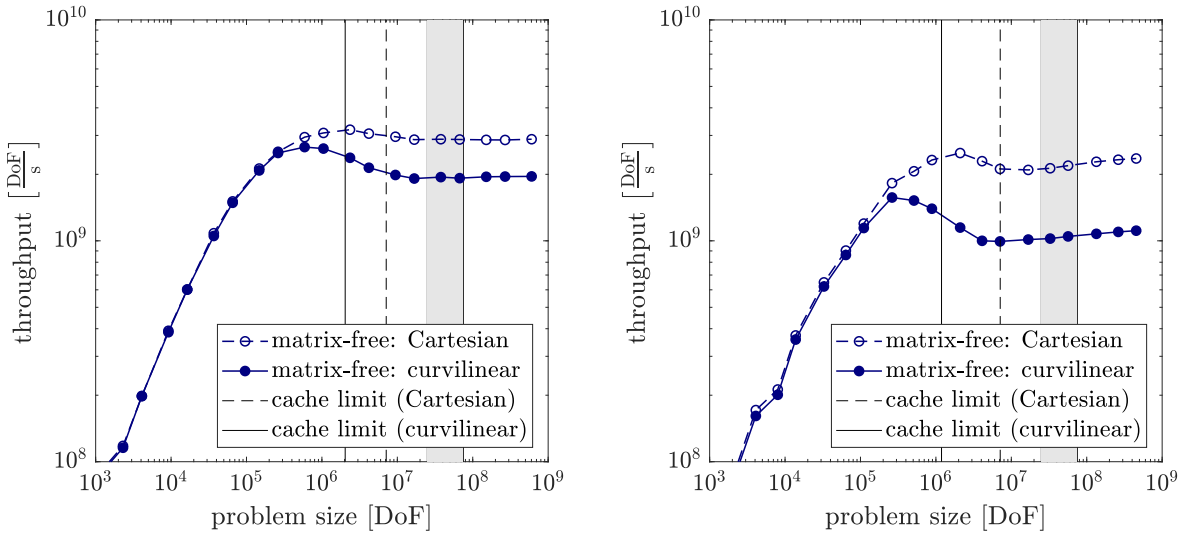
and the linearized momentum operator using the Laplace formulation of the viscous term and the conservative formulation of the convective term

$$\mathbf{y} = \left(\frac{\gamma_0^n}{\Delta t_n} \mathbf{M} + \mathbf{C}_{\mathsf{lin}} \left(\mathbf{u}^{(k)} \right) + \mathbf{V} \right) \Delta \mathbf{u} \, . \tag{4.21}$$

These operators are selected here since they form the main performance-relevant operators of incompressible Navier–Stokes solvers as explained in detail in Chapter 5. Operators composed of different basic operators are evaluated in a single matrix-free evaluation step with one sweep through the data.

4.5.2 Setup of experiments

The numerical experiments shown in this section are performed on an Intel Skylake architecture with AVX512 vectorization. Table 4.2 lists the specifications of the SuperMUC-NG supercomputer in Garching, Germany. The GNU compiler `g++` version 9.2 with optimization flags `-std=c++17 -march=native -O3` is used. Unless specified otherwise, all measurements are conducted for a fully loaded node with 48 MPI processes in order to populate all memory controllers and to avoid that the heterogeneity of the memory access affects the results. The background is that access to main memory is a resource shared by several cores and one core would have access to more than $1/48$ of the peak memory bandwidth if the experiment was run with only a single MPI process, which would then lead to artificially high throughput numbers. This chapter focuses on the node-level performance, while parallel scalability of the present matrix-free implementation is investigated in Kronbichler and Wall (2018) on the level of operator evaluation, and in Chapter 6 on the application-level of incompressible Navier–Stokes solvers. The throughput in degrees of freedom processed per second refers to one matrix-free evaluation of the discretized operator. In this context, a careful setup of the numerical experiments is required in order to ensure reproducibility of the results. The wall time for operator evaluation is averaged over 100 applications (inner iterations) of the discretized operator and the minimum over 10 consecutive runs (outer iterations) is taken. Moreover, the overall wall time of operator evaluation (inner times outer iterations) should take at least one second, otherwise another 10 outer iterations are conducted until this threshold is reached. The discretized operators



(a) two-dimensional problem ($d = 2$)

(b) three-dimensional problem ($d = 3$)

Figure 4.8: Throughput of matrix-free operator evaluation versus problem size measured for scalar Laplace operator on a d-dimensional cube geometry with Cartesian or curvilinear mesh considering polynomial degree $k = 3$.

are evaluated on a structured Cartesian mesh or curvilinear mesh of a cube geometry with $N_{\text{el,1d}}^d$ elements and periodic boundary conditions, where $N_{\text{el,1d}}$ is chosen depending on the polynomial degree of the shape functions in order to meet a predefined range of degrees of freedom. The coarse grid is refined uniformly until the desired number of unknowns is reached and consists of either 1^d, 3^d, or 5^d elements so that the total number of unknowns is within the predefined range of DoFs.

4.5.3 Throughput versus problem size

In a first step, the throughput in DoF/s of the matrix-free operator evaluation is studied for the scalar Laplace operator (pressure Poisson operator for incompressible Navier–Stokes solvers) as a function of the problem size for a fixed polynomial degree, where $k = 3$ is considered exemplarily. The increasing problem size is then realized by mesh refinements. The experiment is conducted on a fat memory node in order to study large problem sizes. Figure 4.8 shows results obtained for $d = 2, 3$ on Cartesian and curvilinear meshes. The size of the L2 and L3 caches listed in Table 4.2 corresponds to $14.25 \cdot 10^6$ double precision values. For the Cartesian mesh, the memory transfer through the caches is due to the input and output vectors (2 double precision values per degree of freedom), so that the critical number of degrees of freedom is 7.1 MDoF to saturate caches. For curvilinear meshes, $d^2 + 1$ double values are stored for the geometry per degree of freedom (for the leading order volume integrals) and need to be streamed from main memory during operator evaluation in addition to the 2 double values for the input and output vectors. Hence, the critical problem size saturating caches in case of curvilinear meshes is 2.0 MDoF for $d = 2$ and 1.2 MDoF for $d = 3$. These bounds are shown as vertical lines

in Figure 4.8. A pronounced cache effect can be observed for the curvilinear mesh, while it is less pronounced for the Cartesian mesh. For larger problem sizes, the cache effect does not disappear in form of a sharp border, but rather as a transition zone which corresponds well to the theoretical estimate. For very large problem sizes, the matrix-free code operates in the saturated regime characterized by a constant throughput, indicating optimality of the implementation with respect to problem size. In three space dimensions, the throughput grows slightly in the saturated regime which might be explained by less MPI communication overhead for larger problem sizes due to a better volume-to-surface ratio. The range of problem sizes used below for throughput measurements is indicated by a gray band, ensuring that results are not compromised by cache effects. A significantly larger throughput is achieved for the Cartesian mesh compared to the curvilinear mesh. This aspect is related to the amount of data that needs to be transferred from main memory and is discussed in more detail in Section 4.5.4.

The range of problem sizes discussed above is denoted as throughput regime, where the term throughput regime refers to a situation in which either the compute capabilities of the hardware or the memory bandwidth are sufficiently utilized through sufficiently long loops around useful work, where floating point operations and streaming of data from memory is considered as useful work. Note that this notion is tightly coupled to the roofline performance model. For small problem sizes, the code does not operate in a throughput regime. Instead, efficiency is limited by latency effects. All data would fit into caches, but there is not enough work to keep execution units busy. For this reason, the difference in throughput between Cartesian and curvilinear meshes observed for large problem sizes diminishes for small problem sizes.

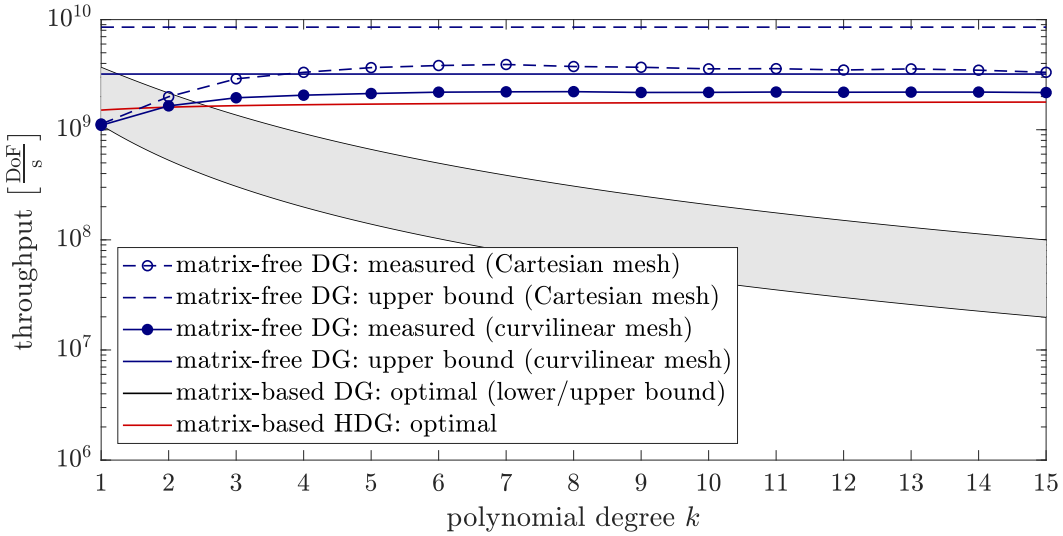
Repeating the same experiments in single-precision (`float`) yields an increase in throughput by a factor of $1.7-2.0$ in the saturated regime, as expected due to the explicit SIMD vectorization of the code. In the latency-bound regime at a low number of DoFs, the benefit of single-precision again diminishes and the curves approach those for double-precision computations.

4.5.4 Throughput versus polynomial degree measured in saturated regime

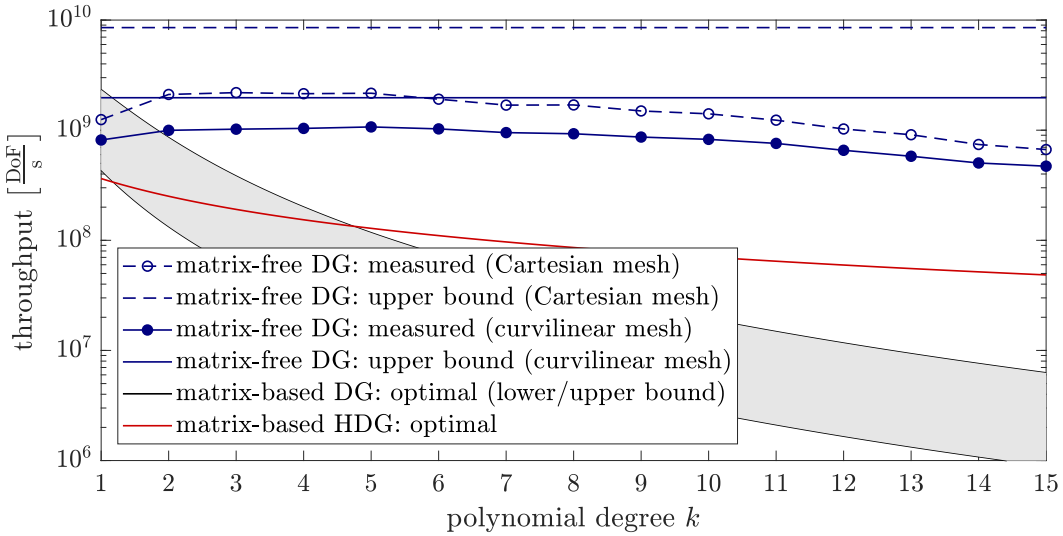
According to the results obtained in the previous section, a problem size of $25 - 75$ MDoF is chosen in order to avoid cache effects and to ensure that measurements are conducted in the regime of saturated caches. Of course, exploiting cache effects intentionally in order to achieve improved performance is reasonable for the simulation of practical problems. The throughput is measured for polynomial degrees in the range $1 \leq k \leq 15$. Figure 4.9 shows the throughput of the matrix-free versus matrix-based evaluation of the scalar Laplace operator. The matrix-free version is the performance *measured* for an interior penalty DG discretization with nodal Gauss–Lobatto basis. The corresponding horizontal lines represent theoretical upper bounds based on the following estimate: Assuming that the code is fully memory-bound, $2.56 \cdot 10^{10}$ double values can be transferred from memory per second when operating at the STREAM bandwidth. Considering first the Cartesian case where no geometry data needs to be streamed from main memory, the input and output vectors have to be read from memory (the latter due to the read-for-ownership policy), and the output vector has to be written to memory, resulting in 3 double values per degree of freedom or a throughput of $8.5 \cdot 10^9$ DoF/s. For the curvilinear case with deformed elements, the data transfer amounts to $3 + d^2 + 1$ (vectors, Jacobian, and Jacobian de-

terminant times quadrature weights) double values per degree of freedom for volume integrals. Additional data transfer for face integrals is neglected in this estimate (due to the lower dimensionality of face integrals), so that this theoretical limit is only strictly valid in the limit $k \to \infty$. This gives an upper bound of $3.2 \cdot 10^9$ DoF/s for $d = 2$ or $2.0 \cdot 10^9$ DoF/s for $d = 3$. The measured performance yields a throughput almost independent of the polynomial degree. For $d = 2$, the measured performance is approximately a factor of 2 smaller than the upper bound for the Cartesian case, and a factor of 1.5 for the curvilinear case for intermediate and large polynomial degrees. For $d = 3$, the gap to the upper bound is at least a factor of 4 for the Cartesian case, but is again as small as a factor of approximately 2 for the curvilinear case for intermediate polynomial degrees. These results already indicate that the matrix-free implementation is mainly memory-bound for the curvilinear mesh. The circumstance that the gap to the theoretical upper bound is somewhat larger in the Cartesian case (especially for $d = 3$) gives indications that the implementation might be partially compute-bound for the Cartesian mesh, an aspect that needs to be verified by a roofline analysis.

According to the analysis in Kronbichler and Kormann (2019), the gap to the upper memory-bound limit is due to a sub-optimal data access pattern for face integrals and can significantly be reduced mainly by three measures: (i) element-wise face integrals with finite difference data access pattern for improved data locality and cache usage, (ii) Hermite-like basis functions for operators with second derivatives in order to reduce the amount of neighbor data that needs to be read for the computation of face integrals, and (iii) shared memory parallelization within nodes in order to reduce the additional data traffic required in case of MPI communication. These optimizations are proposed in the recent works by Kronbichler and Kormann (2019), Kronbichler et al. (2019), Kronbichler and Allalen (2018), which suggest that two thirds of the theoretical optimal performance can be reached by these measures. Note that the gap can not be closed entirely since the theoretical upper bound neglects geometric data required for face integrals, and since this limit further assumes an entirely memory-bound code, optimal cache usage with a single sweep through the data (caches are assumed large enough), and infinite compute resources. For $d = 3$, the throughput is almost constant for small and intermediate polynomial degrees, which is due to face integrals and/or data access of complexity $\mathcal{O}(1)$ per degree of freedom. However, the throughput decreases slightly for large k, where two explanations are plausible. On the one hand, arithmetic operations for volume integrals have complexity $\mathcal{O}(k)$ per degree of freedom which would imply a $1/k$ decrease in throughput for large polynomial degrees if this effect becomes dominant. On the other hand, this behavior might be related to data locality and caches either originating from volume integrals or face integrals. Regarding the computation of volume integrals, the temporary data arrays required might no longer fit into caches for large polynomial degrees, so that parts of the data are evicted to main memory and need to be read several times during the computation of volume integrals. Regarding the computation of face integrals, the data transfer analysis in Kronbichler et al. (2019) reveals that the chosen Lagrange basis with MPI communication leads to a continuously growing data volume per DoF for increasing polynomial degree, which is attributed mainly to face integrals by a comparison to an alternative Hermite-like basis and alternative OpenMP parallelization. For the present experiments, the hypothesis that data access causes the decrease in throughput rather than arithmetic throughput is verified by measuring the floating point performance and the memory bandwidth, see also the roofline analysis in Section 4.5.5. This analysis reveals that the floating point performance and the Flop rate decrease for large k while the memory bandwidth is close to its



(a) two-dimensional problem ($d = 2$)



(b) three-dimensional problem ($d = 3$)

Figure 4.9: Throughput of matrix-free operator evaluation measured for scalar Laplace operator on a d-dimensional cube geometry with Cartesian or curvilinear mesh and comparison to theoretical optimal implementations of matrix-based variants.

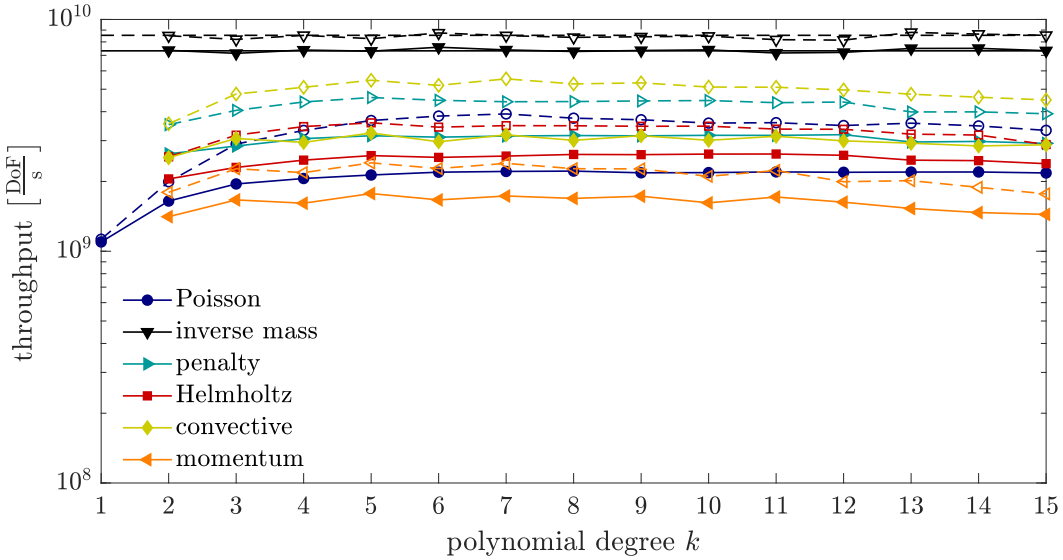
theoretical limit. However, from a theoretical perspective one would expect that the arithmetic intensity increases for large k (due to the complexity of volume integrals) in case of optimal data access. This is a clear indication that the decrease in throughput for large k in three space dimensions is caused by sub-optimal data access rather than the complexity of volume integrals.

The matrix-based implementation in Figure 4.9 assumes an *optimal* implementation of the matrix-vector product and neglects assembly costs. The theoretical lower and upper bounds shown for the matrix-based implementation are related to the type of DG basis and the "stencil-

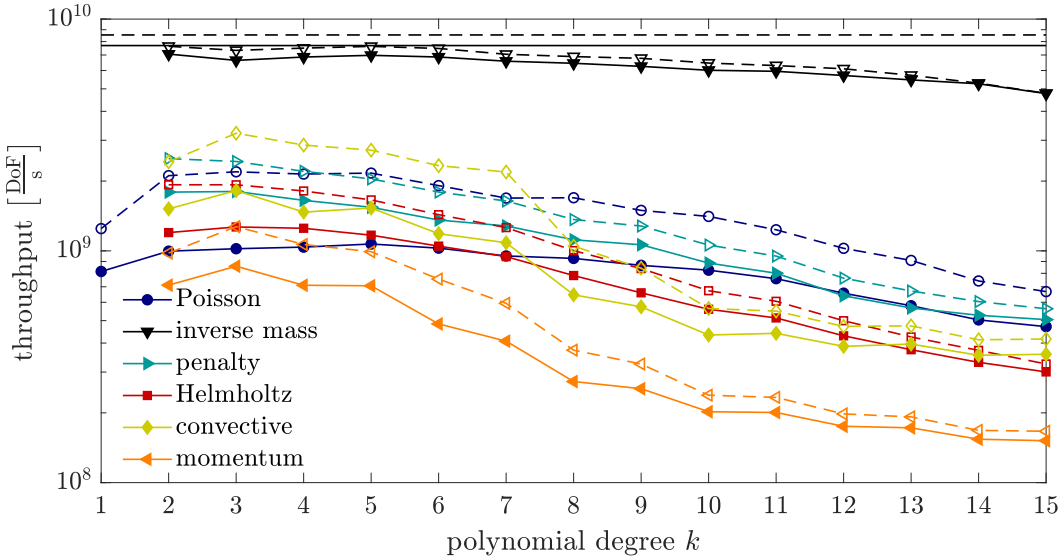
width" (sparsity of block-matrices of neighboring elements). If only the block-diagonal is taken into account and block matrices of neighboring elements are assumed sparse and are neglected, the matrix-based version will perform closer to the upper bound corresponding to $(k+1)^{2d}$ nonzeros per element in the matrix. If the block-matrices of neighboring elements are assumed dense, it will perform closer to the lower bound corresponding to $(2d+1)(k+1)^{2d}$ nonzeros per element. The assumption of an optimal implementation for the matrix-based variant implies that the code is assumed to run with full memory throughput at the STREAM bandwidth of the hardware where only the non-zeros of the matrix as well as the input and output vectors have to be read from/written to memory. Furthermore, optimal spatial and temporal data locality is assumed, i.e., perfect utilization of cache-lines and caches. The HDG variant assumes that each of the d faces of an element couples to $4d-1$ faces resulting in $d(4d-1)(k+1)^{2(d-1)}$ nonzeros per element, and an otherwise optimal implementation as for the matrix-based DG variant.

The matrix-free DG variant is faster than the matrix-based DG variant for $k \geq 3$ if $d = 2$, and for $k \geq 2$ if $d = 3$. A speed-up by orders of magnitude can be expected for large polynomial degrees. These results also illustrate that while the HDG approach is attractive in two space dimensions, it is clearly less efficient than matrix-free operator evaluation in three space dimensions. Note that actual implementations of the matrix-based variants will result in a lower throughput than the optimal results shown in Figure 4.9, e.g., due to non-optimal usage of caches and additional memory transfer for indexing. Further, this comparison neglects cost for assembling matrices as well as static condensation (for HDG), which needs to be performed in case of the matrix-based variants and which would further increase the gap between matrix-free and matrix-based approaches. For matrix-based variants, there is no difference in performance between Cartesian and deformed elements. For the matrix-free implementation, the gap in performance between Cartesian and deformed elements can be expected to become smaller in case of vectorial problems (e.g., velocity in case of the Navier–Stokes equations) as compared to the scalar Laplace problem shown here, since the amount of data required for the geometric information becomes smaller relative to the amount of data required for the DoF vectors when considering an increasing number of components of the vectorial problem.

Next, the throughput is investigated as a function of the polynomial degree k for the incompressible Navier–Stokes operators discussed above. The results are shown in Figure 4.10. The highest throughput is achieved by the inverse mass matrix operator, which is mainly memory-bound and operates close to the theoretical limit corresponding to the STREAM bandwidth. The theoretical limit is shown as horizontal lines and assumes that 3 doubles need to be streamed from memory in the Cartesian case (one for the input vector and two for the output vector due to the read-for-ownership policy), and one additional double value per d degrees of freedom for the Jacobian determinant in each quadrature point in the curvilinear case. A similar performance can be expected for vector-update operations occurring in time integration schemes and iterative solvers. Scaling and adding vectors results in a low arithmetic intensity (around or below one Flop per double value) and the code is clearly memory-bound (a so-called streaming operation). The throughput of discretized differential operators is generally lower and can be explained by the complexity of the individual operators. The Helmholtz operator differs from the Laplace operator only by an additional cheap mass matrix term and, therefore, achieves a comparable throughput. The penalty operator and convective operator achieve a somewhat higher throughput than the Helmholtz operator, which can be explained by the fact that these operators require only the value (but not the gradient) of the solution in order to evaluate fluxes for



(a) two-dimensional problem ($d = 2$)



(b) three-dimensional problem ($d = 3$)

Figure 4.10: Throughput of matrix-free operator evaluation for performance-relevant incompressible Navier–Stokes operators on a d-dimensional cube geometry with Cartesian mesh (dashed lines and open markers) or curvilinear mesh (solid lines and filled markers).

face integrals. The momentum operator forms the most complex operator since it combines the mass matrix, convective, and viscous operators. Additionally, the linearized momentum operator has to read an additional DoF vector for the linearized velocity field and evaluate this velocity in all quadrature points, increasing both memory transfer and arithmetic operations. For these reasons, the momentum operator achieves the lowest performance. Moreover, the nonlinear con-

vective operator and the linearized momentum operator use an increased number of quadrature points, $n_q = \lceil \frac{3k_u+1}{2} \rceil$, further reducing the throughput for these operators. The rounding to the next larger integer value explains the saw-tooth behavior observable for the throughput results of the convective and momentum operators.

Comparing the results for $d = 2$ and $d = 3$, a fundamentally different behavior can be observed, in accordance with the results shown above for the Laplace operator. In two space dimensions, the throughput is almost independent of the polynomial degree. Put differently, high-order methods are as efficient as low-order methods for $d = 2$ in terms of the matrix-free operator evaluation. In three space dimensions, the throughput is almost constant in the range $2 \leq k \leq 7$, but decreases for increasing polynomial degree, rendering very high polynomial degrees more expensive than moderately large polynomial degrees $2 \leq k \leq 7$. For the inverse mass matrix operator, the decrease in throughput for large k can be explained by the fact that the implementation tends to become compute-bound for large k. For the other operators, however, the roofline analysis in Section 4.5.5 reveals that the implementation does actually not enter the compute-bound regime for large k, but the decrease in throughput is due to data access, such as sub-optimal temporal data locality for volume integrals and an increase in data volume related to face integrals as discussed above for the Laplace operator. Another factor causing a decrease in throughput for large polynomial degrees when compared to the (inverse) mass matrix operator is MPI communcation overhead. For the same number of unknowns and compute cores, MPI communication overhead increases for large polynomial degrees due to the smaller number of elements per core as compared to low-order methods and the increased surface-to-volume ratio relevant for MPI ghost layer communication (Hager and Wellein 2010). The decrease in throughput for large k is particularly pronounced for the convective operator and the momentum operator with an increased number of quadrature points, where the throughput decreases significantly in the range $8 \leq k \leq 10$. The size of the temporary data arrays for volume integrals can be estimated as $n_q^d \cdot d \cdot (d+1) \cdot 8 \cdot 8$, where the individual factors account for the number of quadrature points $n_q = \lceil \frac{3k+1}{2} \rceil$, the d velocity components, the fact that both values and gradients need to be evaluated $(d + 1)$, a SIMD width of 8 double values for AVX512 with 8 Byte each. This estimate predicts that the L3 cache of 2.375 MByte per core is exceeded for $k \geq 10$ and the L2 cache of 1 MByte per core for $k \geq 7$. These numbers fit very well to the observed behavior and indicate that the data required for the computation of volume integrals does not fit into caches and needs to be streamed more than once from main memory so that the overall memory transfer increases "unnecessarily". Additional data might reside in caches related to the input and output vectors, and to geometry data with d^2 double values per quadrature point in case of deformed elements.

Similar results as those shown here have already been published in Fehn et al. (2018a) for an Intel Haswell architecture with AVX2 vectorization capabilities. The increased SIMD width of the Skylake architecture in combination with a cache size per core even smaller compared to the Haswell architecture lead to throughput results that are more sensitive w.r.t. the polynomial degree. Performance improvements compared to the Haswell architecture are mainly obtained in the range of moderately large polynomial degrees, see also the works by Arndt et al. (2020b), Kronbichler and Allalen (2018) for cross-platform comparisons of the matrix-free implementation used here. In Kempf et al. (2020), Müthing et al. (2017), Piatkowski (2019), throughput results for high-order DG discretizations of the incompressible Navier–Stokes equations

or its performance-relevant operators are published for a matrix-free implementation realized in the DUNE-PDELab framework and using comparable Intel Haswell/Skylake hardware. The present results (and those already published in Fehn et al. (2018a) for an Intel Haswell architecture) achieve a significantly higher throughput. The present results are also faster than the GPU implementation of matrix-free high-order continuous Galerkin discretizations of the incompressible Navier–Stokes equations presented in Franco et al. (2020), despite the fact that the present work considers more expensive DG methods with face integrals and the fact that the Nvidia V100 hardware used in Franco et al. (2020) provides a significantly higher absolute performance in terms of memory bandwidth (900 GByte/s) and floating point operations (7800 GFlop/s) than the Intel Skylake CPU hardware (see Table 4.2) used here.

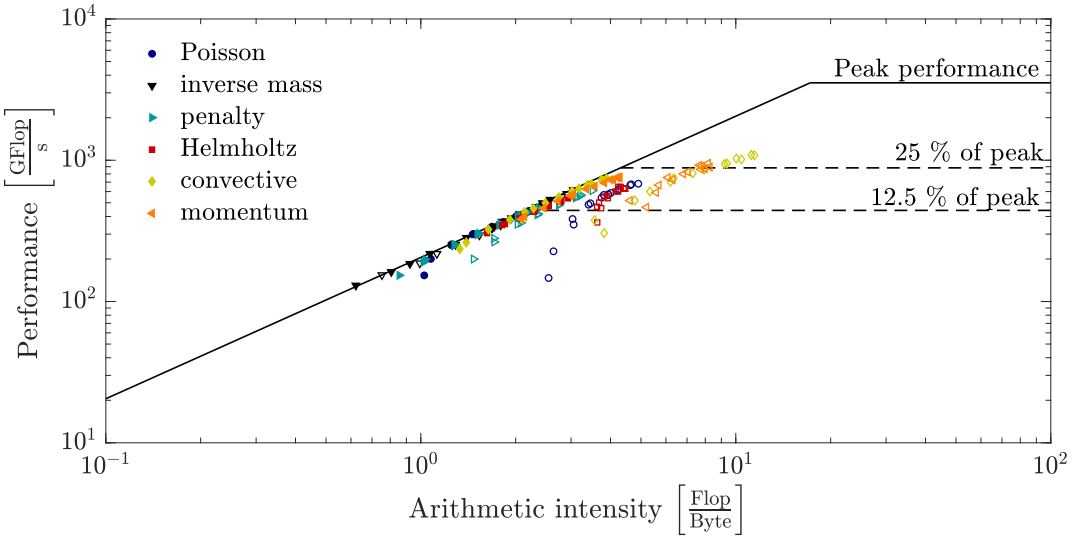
Remark 4.7 *A similar numerical investigation of the matrix-free operator evaluation for a DG discretization of the compressible Navier–Stokes equations implemented in the course of this thesis has already been published in Fehn et al. (2019c), where results are shown for an Intel Haswell architecture.*

4.5.5 Roofline analysis

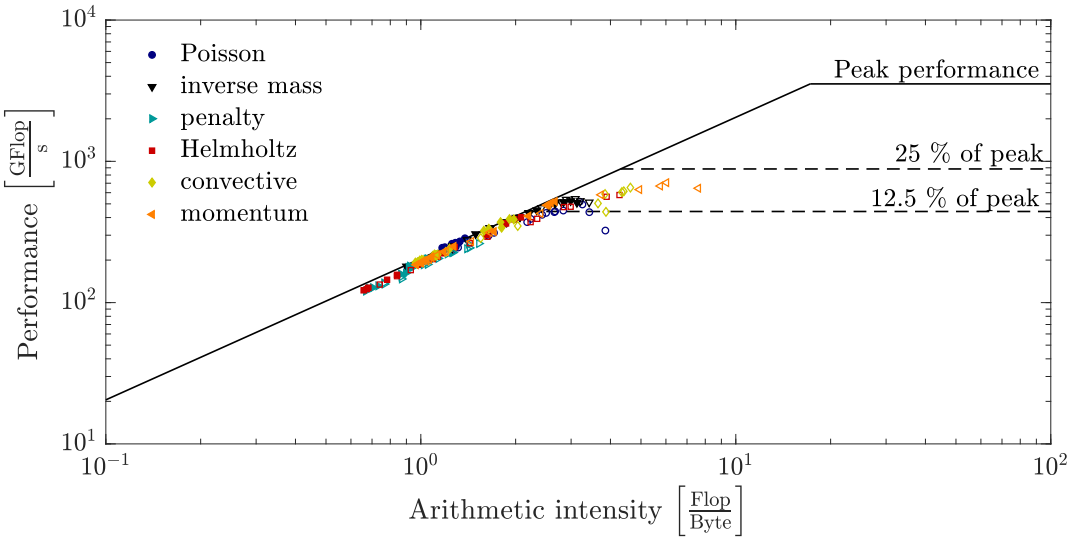
This section characterizes the matrix-free implementation of the performance-relevant incompressible Navier–Stokes operators in terms of the roofline model. For the throughput results shown above, the flop rate and the data transfer rate from/to main memory are measured with the open-source tool likwid (Treibig et al. 2010). From these measurements, the arithmetic intensity can be computed so that data points corresponding to the different operators and polynomial degrees can be inserted into the roofline plot of the considered hardware. The results are shown in Figure 4.11. The STREAM bandwidth is utilized as the maximum possible memory bandwidth. The peak performance assumes as clock-frequency of 2.3 GHz (corresponding to AVX512 vectorization). Additional horizontal lines indicating 25% and 12.5% are shown for ease of interpretation of the results.

For $d = 2$, the arithmetic intensity increases for increasing polynomial degree as expected theoretically under the assumption of optimal data locality and sufficiently large caches to hold temporary data, see also Table 4.1. The inverse mass matrix operator is memory-bound for the Cartesian mesh and the curvilinear mesh. For the other operators involving face integrals, all operators are memory-bound on the curvilinear mesh. For the Cartesian mesh, the results show an offset from the STREAM bandwidth limit where the distance to the maximum memory bandwidth seems to be influenced by the type of operations involved in face integrals. The penalty operator only evaluates the solution on faces, the Poisson and Helmholtz operators evaluate both the value and gradient of the solution on faces. The convective and momentum operators additionally evaluate the convective flux and use an increased number of quadrature points. In this order, the distance to the maximum memory bandwidth becomes larger. These results indicate that the $d = 2$ operator evaluation on Cartesian meshes is partly compute-bound or limited by instruction throughput or other latency effects.

For $d = 3$, the results follow a different pattern. Only the inverse mass matrix operator shows the expected behavior of increasing arithmetic intensity for increasing polynomial degree. The implementation is memory-bound for small and moderate polynomial degrees and tends to become compute-bound for large polynomial degrees with a performance of approximately 12.5%



(a) two-dimensional problem ($d = 2$)



(b) three-dimensional problem ($d = 3$)

Figure 4.11: Roofline analysis for performance-relevant incompressible Navier–Stokes operators on a d-dimensional cube geometry with Cartesian mesh (open markers) or curvilinear mesh (filled markers).

of peak. The other operators show an inverse behavior characterized by a high arithmetic intensity for low polynomial degrees and a lower arithmetic intensity for high polynomial degrees. As already mentioned above, this does not fit to the (simplified) theoretical model that counts operations and the minimum required transfer of data from main memory according to Table 4.1. The number of floating point operations can be considered as fixed for one operator evaluation, so that an arithmetic intensity lower than the expected theoretical value must be related to an unexpectedly high amount of memory transfer. This can occur if the required data is not streamed just

once from main memory during operator evaluation as would be optimal from the point of view of data access, but if the same data needs to be streamed repeatedly from main memory during one operator evaluation or if intermediate results can not reside in fast caches and are evicted from caches to the slower main memory. For these reasons, the matrix-free operator evaluation is mainly memory-bound in three space dimensions on deformed meshes, and also on Cartesian meshes for large polynomial degrees.

Overall, these results demonstrate the high optimization level of the matrix-free implementation in `deal.II`. Put differently, the fact that the compute part of the matrix-free algorithms is well-optimized in terms of minimizing operations and optimally exploiting SIMD vectorization enables to more and more encounter memory-bound situations on newest hardware with increasing Flop-to-Byte ratio. To put these results into perspective, the studies by Kempf et al. (2020), Müthing et al. (2017), Piatkowski (2019) consider a similar high-order DG discretizations and achieve a significant fraction of the peak floating point performance (up to around 50%), but a lower performance in the practically relevant throughput metric. This highlights that floating point performance does not give direct insights into the speed of an implementation. Analyzing the roofline performance of a code should be understood as an accompanying analysis helping to identify bottlenecks and to improve throughput. This appears to be worth mentioning given that many works, see for example the recent study by Świrydowicz et al. (2019) on GPU acceleration of high-order matrix-free finite element operator evaluation on tensor-product elements, primarily consider the Flop-metric, insinuating that this metric would be the target of code optimizations.

4.6 Conclusion and outlook

This chapter started with an introduction to the idea of matrix-free operator evaluation, with a particular emphasis on discontinuous Galerkin methods discretized on meshes composed of tensor-product elements. The matrix-free approach with sum-factorization is discussed in comparison to other matrix-free and matrix-based approaches. A review of developments in computer hardware gives further insights and explains why the matrix-free approach is particularly interesting on modern computer hardware. The design choices of the present matrix-free implementation are summarized and its practical performance is analyzed numerically by means of throughput measurements and a roofline analysis for an Intel Skylake computer architecture. The results demonstrate optimal algorithmic complexity and the achieved efficiency is put into perspective by comparisons to theoretical models with optimal implementation. The present matrix-free implementation is shown to perform close to the hardware limits for a selection of performance-relevant incompressible Navier–Stokes operators in two and three space dimensions and for Cartesian and deformed elements. The throughput as a function of the polynomial degree is almost constant for $d = 2$ and decreases moderately with increasing k for $d = 3$. As a preparation for subsequent chapters, this chapter demonstrates that both low-order and high-order methods can be implemented very efficiently by matrix-free techniques, i.e., the throughput shows only a mild dependency on the polynomial degree.

From that point onwards, further performance improvements would only be possible by changing the algorithm. Some of these measures have already been mentioned above, such as an alternative element-wise organization of face integrals and a Hermite-like basis (for operators with

second derivatives) in order to optimize data transfer for face integrals. In a similar direction, shared memory parallelization has been shown to further improve performance (Kronbichler and Kormann 2019). The present results confirm that temporal data locality becomes an increasingly pressing issue on modern hardware with wider SIMD units in case of the present implementation with vectorization over elements and especially for vectorial problems that further increase the size of temporary data arrays compared to scalar PDEs. An optimal implementation might then use vectorization over elements for small polynomial degrees, and vectorization within elements for large polynomial degrees. Vectorization within elements is particularly interesting for high-order elements: The temporary data arrays would become smaller and allow larger polynomial degrees to fit into caches, and also parallel scalability (see also Chapter 6) could be improved for high-order elements due to a finer granularity. Another route of optimization could be the treatment of geometric data for deformed elements, e.g., loading less geometry data from main memory and instead recomputing geometric information from the nodal coordinates could improve throughput by relaxing the memory bottleneck. However, the current state-of-the-art reveals that all of these questions are difficult to answer once and for all. Instead, it can be expected that different algorithms and implementations are optimal for different operators, different parameters such as the polynomial degree or Cartesian/deformed elements, and different hardware platforms.

5 Iterative solution techniques and preconditioning

The use of iterative solution techniques for algebraic systems of equations resulting from discretization in space and time is indispensable for large-scale applications. Together with robust preconditioners that ensure mesh-independent convergence rates (e.g., multigrid), these systems of equations can then be solved with optimal computational complexity, meaning that the computational costs scale linearly with the number of unknowns. This chapter discusses well-established block preconditioners for monolithic incompressible Navier–Stokes solvers. For both coupled and projection-type Navier–Stokes solvers, the main algorithmic component that is at the heart of a fast incompressible Navier–Stokes solver is an efficient multigrid preconditioner for the pressure Poisson problem. For reasons of computational efficiency, an emphasis is put on matrix-free preconditioners that can be realized with optimal computational complexity for large polynomial degrees. A novelty of this chapter is a matrix-free hybrid multigrid preconditioner for high-order discontinuous Galerkin discretizations that combines geometric, polynomial, and algebraic coarsening techniques with an additional transfer to continuous function spaces. A significant part of the implementation of the hybrid multigrid algorithm has been done by Münch (2018). The hybrid multigrid developments have already been published in Fehn et al. (2020) and are reproduced here. As a second main ingredient, this chapter addresses block preconditioners for the saddle-point problem arising from discretizations of the incompressible Navier–Stokes equations using a coupled solution approach. Efficient preconditioning techniques for projection-type Navier–Stokes solvers have already been described in Fehn et al. (2018a).

Section 5.1 reviews the state-of-the-art on multigrid methods for high-order finite element discretizations and describes novel contributions of the present work. Section 5.2 discusses the novel hybrid multigrid solver by the example of the Poisson equation. Numerical results for the hybrid multigrid solver are shown in Section 5.3. Then, Section 5.4 addresses block preconditioners for the incompressible Navier–Stokes equations. Section 5.5 highlights similarities in preconditioning between different Navier–Stokes solution strategies discussed in this work. The robustness of preconditioners for the incompressible Navier–Stokes equations is analyzed in Section 5.6 for a three-dimensional turbulent flow problem. Finally, Section 5.7 summarizes this chapter.

5.1 Introduction

When solving (non-)linear systems of equations by means of iterative solution techniques, the evaluation of nonlinear residuals as well as the application of linear(ized) operators in Krylov solvers are readily available in a matrix-free implementation environment (Brown 2010). More importantly, however, also preconditioners should rely on matrix-free algorithms with opti-

mal complexity, since the whole solver will otherwise run into the bottlenecks of matrix-based solvers characterized by immense memory requirements and overwhelming data transfer. For elliptic PDEs, multigrid methods (Trottenberg et al. 2001) are among the most efficient solution techniques (Gholami et al. 2016), since they allow a solution with linear complexity in the number of unknowns and since they are applicable to problems on complex geometries. In general, one can distinguish between geometric multigrid methods with an explicit construction of coarser representations of the discrete operator (typically through a hierarchy of meshes), and algebraic multigrid techniques that only require the matrix representation of the operator on the fine level and that construct coarser levels by purely algebraic techniques. Although attractive due to their black-box interface to linear solvers, algebraic multigrid methods are inherently matrix-based, which is why they show – as compared to their matrix-free geometric multigrid counterparts – severe limitations in problem size due to increased memory requirements for storing sparse matrices, increased setup costs, and slow execution on modern hardware with growing machine in-balance (Hager and Wellein 2010), see Chapter 4. As argued in Bauer et al. (2020), where the solution of a finite element problem with 10^{13} unknowns is demonstrated by the use of matrix-free multigrid algorithms, matrix-based methods are limited to significantly smaller problem sizes. For high-order spectral element discretizations, the limitations of matrix-based methods are even more severe (Chapter 4), so that the use of matrix-free algorithms becomes inevitable in designing performant PDE solvers. For example, it can be expected that optimal complexity of all solver components is crucial to render high-order discretization methods more efficient in under-resolved application scenarios such as turbulent flows (Fehn et al. 2018a).

5.1.1 Multigrid for high-order discretizations: state-of-the-art

In the context of high-order finite element discretizations, multigrid methods can be categorized into h-multigrid, p-multigrid, and algebraic multigrid (AMG) techniques.

Geometric or h-multigrid methods rely on a hierarchy of meshes. Robust h-multigrid techniques for high-order DG discretizations have been developed and analyzed in Brenner et al. (2009), Brenner and Zhao (2005), Gopalakrishnan and Kanschat (2003), Hemker et al. (2003) for uniformly refined meshes and in Kanschat (2004, 2008) for adaptively refined meshes. Recent improvements of these algorithms towards high-performance, matrix-free implementations have been developed in Kronbichler and Wall (2018), where a performance comparison of high-order continuous and discontinuous Galerkin discretizations as well as hybridizable discontinuous Galerkin methods is carried out. A GPU variant for continuous finite elements has been proposed in Kronbichler and Ljungkvist (2019). The parallel efficiency for adaptively refined meshes is discussed in Clevenger et al. (2020). Large-scale applications of these h-multigrid methods can be found in the field of computational fluid dynamics (CFD) and the incompressible Navier–Stokes equations (Fehn et al. 2018a, 2021b, Krank et al. 2017).

For spectral element discretizations, p-multigrid, or synonymously spectral element multigrid, is frequently used, where the multigrid hierarchy is obtained by reducing the polynomial degree of the shape functions. Coarsening and multigrid transfer is particularly simple since the function spaces of different multigrid levels are nested (also for element geometries deformed via high-order mappings) and since all operations are element-local. This approach has first been proposed and theoretically analyzed in Maday and Munoz (1988), Rønquist and Patera (1987), and later investigated, for example, in Helenbrook et al. (2003), Helenbrook and Atkins (2008), Huismann

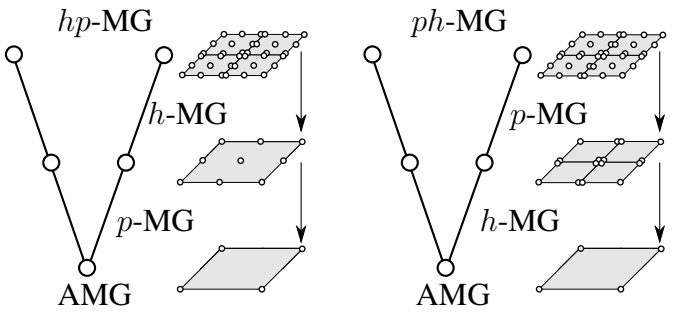


Figure 5.1: Illustration of combined geometric–polynomial–algebraic multigrid algorithms for nodal high-order discontinuous Galerkin discretizations.

et al. (2019), Lottes and Fischer (2005), Mascarenhas et al. (2010), Stiller (2016, 2017a,b). A related hierarchical scale separation solver is proposed in Aizinger et al. (2015). Polynomial multigrid techniques are also frequently used to solve the compressible Euler equations (Bassi et al. 2009, Bassi and Rebay 2003, Fidkowski and Darmofal 2004, Helenbrook and Mascarenhas 2016, Hillewaert et al. 2006, Luo et al. 2006, Mascarenhas et al. 2009, Nastase and Mavriplis 2006b, Rasetarinera and Hussaini 2001) and compressible Navier–Stokes equations (Bassi et al. 2011, Diosady and Darmofal 2009, Fidkowski et al. 2005, Ghidoni et al. 2014, Luo et al. 2012, Persson and Peraire 2008, Shahbazi et al. 2009).

Algebraic multigrid techniques extract all information from the assembled system matrix to generate coarser levels and are attractive as they can be applied in a black-box fashion. AMG is considered in Heys et al. (2005) for high-order continuous Galerkin discretizations and in Lasser and Toselli (2001), Olson and Schroder (2011), Prill et al. (2009) for (high-order) discontinuous Galerkin discretizations. AMG applied directly to high-order DG discretizations faces several challenges, among them the construction of robust smoothers for matrices that lose diagonal dominance, but most importantly the computational complexity of matrix-based algorithms (especially for three-dimensional problems) compared to their matrix-free counterparts, see Chapter 4. These limitations are also exemplified by the fact that AMG methods for high-order discretizations have only been applied to small two-dimensional problems in the works mentioned above. For reasons of computational complexity, it appears to be inevitable to combine algebraic multigrid techniques with some form of geometric coarsening to achieve a computationally efficient approach for practical applications (Bastian et al. 2012, Prill et al. 2009, Siefert et al. 2014).

Multigrid transfer operators are negligible in terms of computational costs when implemented in a matrix-free way with optimal complexity (Bastian et al. 2019, Kronbichler and Ljungkvist 2019) and when the solver is applied away from the strong-scaling limit. Therefore, multigrid smoothers and coarse-grid solvers remain as the main performance-relevant multigrid components. Adhering to the matrix-free paradigm poses two major challenges that need to be addressed and further improved:

- Matrix-free smoothers: To fully exploit the advantages of matrix-free algorithms with sum-factorization for operator evaluation, multigrid smoothers should make use of these algorithms as well, but this problem has so far only been solved for certain types of smoothers and mainly for elliptic problems. Polynomial smoothers based on the Chebyshev itera-

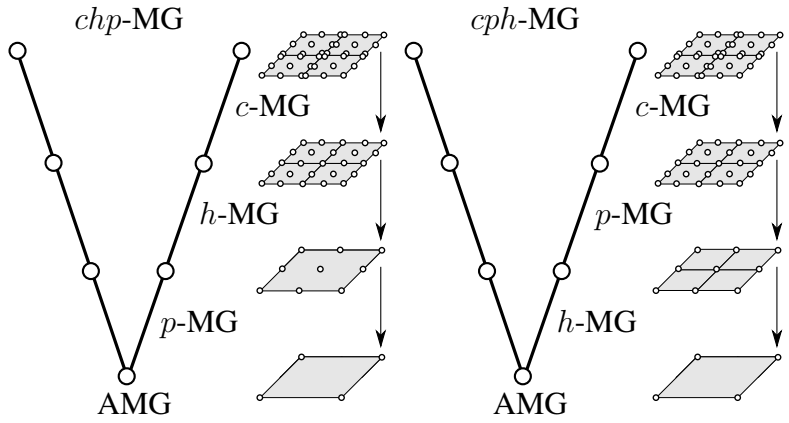


Figure 5.2: Illustration of combined geometric–polynomial–algebraic multigrid algorithms with additional c-transfer between discontinuous and continuous shape functions.

tion (Adams et al. 2003, Sundar et al. 2015) or explicit Runge–Kutta methods (Helenbrook et al. 2003, Hillewaert et al. 2006, Luo et al. 2006, 2012, Rueda-Ramrez et al. 2019) can be immediately realized in a matrix-free way, are inherently parallel, and are therefore widely used in a high-order context. More challenging are smoothers based on elementwise inversion of operators such as block Jacobi, block Gauss–Seidel, block ILU, or (overlapping) Schwarz smoothers. Many implementations rely on matrix-based algorithms for smoothers (Bassi et al. 2009, 2011, Diosady and Darmofal 2009, Fidkowski and Darmofal 2004, Fidkowski et al. 2005, Ghidoni et al. 2014, Helenbrook and Mascarenhas 2016, Mascarenhas et al. 2009, Nastase and Mavriplis 2006b, Persson and Peraire 2008, Rasetarinera and Hussaini 2001, Shahbazi et al. 2009), limiting applicability mainly to two-dimensional problems, while three-dimensional problems become prohibitively expensive for higher polynomial degrees due to the complexity of $\mathcal{O}(p^{2d+1})$ for the assembly, $\mathcal{O}(p^{3d})$ for factorizations, and $\mathcal{O}(p^{2d})$ for matrix-vector products. On Cartesian meshes and tensor-product elements, elementwise inversion of operators with optimal complexity is possible via the fast diagonalization method (Lynch et al. 1964), which has first been applied in Couzy and Deville (1994, 1995) in the context of spectral element discretizations and analyzed in more detail in Fischer et al. (2000), Fischer and Lottes (2005), Lottes and Fischer (2005), Stiller (2016, 2017a,b), Witte et al. (2020) in the context of overlapping Schwarz preconditioners and multigrid smoothers. Other tensor-product based preconditioners and smoothers for high-order DG discretizations exploiting fast matrix-free operator evaluation with sum-factorization and applicable to more complex operators and non-Cartesian meshes have been proposed recently in Bastian et al. (2019) using inner Krylov iterations to solve the local block-Jacobi problems and in Pazner and Persson (2018) using SVD-based tensor-product preconditioners.

- Hybrid multigrid algorithms: Due to the improved efficiency of matrix-free evaluation routines, the generation of coarser multigrid levels should rely on non-algebraic coarsening, i.e., mesh (or geometric) coarsening and reducing the polynomial degree of the function space for high-order methods, leading to the idea of hybrid hp- and ph-multigrid methods as illustrated in Figure 5.1. To benefit from the increased throughtput of matrix-free

methods as much as possible, it is reasonable to exploit all possibilities of geometric and polynomial coarsening in the multigrid hierarchy. For complex engineering applications, the number of geometric mesh levels is low (typically $0, 1, 2$) and coarse meshes might consist of millions of elements so that simple iterative solution techniques like a conjugate gradient iteration with Jacobi preconditioner used as coarse-level solver would become too expensive and dominate the overall computational costs of the multigrid solver. Applying algebraic multigrid techniques for the coarse-grid problem discretized with linear finite elements is a good compromise between effectiveness of coarse-level preconditioning and computational efficiency, given that sparse matrix-vectors products are competitive to matrix-free evaluation routines in this low-order regime. At the same time, smoothers for algebraic multigrid methods work best at low polynomial degrees due to a better diagonal dominance of the matrix as opposed to high-order shape functions (Kronbichler and Wall 2018). In Table 5.1, important contributions in the field of hybrid multigrid solvers are summarized. In Helenbrook et al. (2003), Helenbrook (2001), hybrid multigrid solvers combining p-MG with h-MG have been presented for high-order discretizations. In Nastase and Mavriplis (2006b), Shahbazi et al. (2009), p-multigrid is used along with algebraic multigrid for the coarse problem. In terms of p-multigrid, the works by Bastian et al. (2012), Dobrev et al. (2006), Siefert et al. (2014) can be categorized as two-level algorithms with transfer to continuous linear shape functions from the fine level to the coarse level, which is solved by an h-multigrid approach in Dobrev et al. (2006) and an algebraic multigrid approach in Bastian et al. (2012), Siefert et al. (2014). These works have the limitation that the underlying implementation is not matrix-free and, therefore, suffers from the complexity of matrix-based approaches. The main limitation of the approach in Sundar et al. (2012) for hexahedral meshes based on the octree concept is that it only supports linear continuous finite elements (a similar approach for tetrahedral elements is presented in Lu et al. (2014)). An extension towards p-multigrid has been done in Rudi et al. (2015), but results are limited to linear and quadratic shape functions. The approach in O'Malley et al. (2017) combines p-multigrid with AMG but uses expensive matrix-based implementations which could explain why results are only shown for quadratic elements. In more recent works, hybrid multigrid algorithms for high-order methods with completely matrix-free implementation are discussed in Bastian et al. (2019), extending a previous work by Bastian et al. (2012) towards a matrix-free implementation developed in Kempf et al. (2020), Müthing et al. (2017). That work does not exploit geometric coarsening (h-multigrid) and the high-order discretization with degree p is immediately reduced to a linear space within the multigrid algorithm (and is therefore categorized as a two-level algorithm rather than p-multigrid). Algebraic multigrid is employed for the coarse problem. An elaborate matrix-free implementation in the context of h-multigrid solvers is presented in Kronbichler and Wall (2018) based on the matrix-free implementation developed in Kronbichler and Kormann (2012, 2019) and available in the `deal.II` finite element library (Alzetta et al. 2018). That work clearly improves the performance numbers of sophisticated geometric multigrid solvers shown in Gholami et al. (2016). One drawback of this pure h-multigrid approach is that its applicability is limited to problems where the coarse grid is comparably simple. As a building block for the present contribution, this implementation has been extended by additional multigrid transfer operators for polynomial multigrid and transfer between discontinuous and continuous function spaces

Table 5.1: Hybrid multigrid algorithms: relevant publications from the literature addressing combined h-, p-, and algebraic multigrid methods are categorized in terms of high-order discretizations ($p > 2$), matrix-free implementations, h-MG, p-MG, and AMG. A similar overview has been given in Münch (2018). Legend: ✓ = fulfilled,(✓) = partly fulfilled, ✗= not fulfilled. The category p-MG is partly fulfilled (✓) if a two-level algorithm is considered with transfer from high-order space of degree p directly to linear space with $p = 1$.

Study	high-order	matrix-free	h-MG	p-MG	AMG
Helenbrook et al. (2003)	✓	✗	✓	✓	✗
Dobrev et al. (2006)	✗	✗	✓	(✓)	✗
Nastase and Mavriplis (2006b)	✓	✗	✗	✓	✓
Shahbazi et al. (2009)	✓	✗	✗	✓	✓
Bastian et al. (2012), Siefert et al. (2014)	✓	✗	✗	(✓)	✓
Sundar et al. (2012)	✗	✓	✓	✗	✓
Lu et al. (2014)	✗	✗	✓	✗	✓
Rudi et al. (2015)	✗	✓	✓	✓	✓
O'Malley et al. (2017)	✗	✗	✗	✓	✓
Bastian et al. (2019)	✓	✓	✗	(✓)	✓
Kronbichler and Wall (2018)	✓	✓	✓	✗	✗
present work (see Fehn et al. 2020)	✓	✓	✓	✓	✓

in Münch (2018). Hybrid multigrid techniques in the context of HDG discretizations are considered, e.g., in Fabien et al. (2019).

There exist other techniques as well with the aim to overcome the complexity of matrix-based methods for high polynomial degrees. Preconditioners and multigrid methods applied to a low-order re-discretization of the operator on a mesh with vertices located on the nodes of the high-order discretization is a well-known technique originating from Deville and Mund (1985, 1990) and has been analyzed for example in Brown (2010), Fischer (1997), Heys et al. (2005), Lottes and Fischer (2005), Pazner (2020), Sundar et al. (2015). Such an approach is not considered here.

5.1.2 Novel contributions of the present work

The present work extends the work in Kronbichler and Wall (2018) towards hybrid multigrid techniques combining geometric (h), polynomial (p), and algebraic coarsening. The goal is to devise a multigrid solver applicable to engineering problems with complex geometry, where the nomenclature "complex" is used to describe problems characterized by coarse grids with many elements. As can be seen from Table 5.1, the individual components relevant for efficient hybrid

multigrid methods are covered by different works. However, none of these works fulfills all properties and it is the aim of the present work to fill this gap.

As a model problem, the constant-coefficient Poisson equation in three space dimensions is studied for the development of hybrid multigrid methods. With respect to the choice of multigrid smoothers, this work mainly considers Chebyshev-accelerated Jacobi smoothers which have the characteristic that convergence rates are independent of h and mildly dependent on p, see Kronbichler and Wall (2018), Sundar et al. (2015). Chebyshev smoothing is particularly attractive since it only requires application of the matrix-vector product and the inverse diagonal of the system matrix, i.e., the smoother benefits from fast matrix-free evaluation routines and is efficient in a parallel setting. Although more aggressive smoothers based on overlapping Schwarz methods resulting in lower iteration counts exist, comparative studies would need to be carried out to answer which approach is more efficient, see the initial investigation in Kronbichler et al. (2019). These aspects are, however, beyond the scope of this thesis. Chebyshev smoothing also works well for variable-coefficient problems with smoothly varying coefficient (Kronbichler and Wall 2018, Sundar et al. 2015), but it can be expected that more robust smoothers might be necessary for more challenging model equations (such as highly varying coefficients in porous media problems or convective terms in Navier–Stokes problems). These aspects are also not covered by the present work.

In case of discontinuous Galerkin discretizations, a transfer from discontinuous to continuous function spaces (denoted here as DG-to-FE transfer) should be considered in addition to h- and p-coarsening in order to further reduce the size of the coarse-level problem, see Rudi et al. (2015). For example, the problem size is reduced by a factor of 2^d for linear elements with $p = 1$. Moreover, this approach is also beneficial in order to reduce iteration counts for the coarse-level problem, due to the fact that existing AMG implementations and smoothers are often most effective on continuous function spaces. However, it is unclear whether to perform the DG-to-FE transfer on the high-order polynomial space p or for the coarse problem at $p = 1$, or likewise on the finest mesh or the coarsest mesh. It is a main finding of the present work that a DG-to-FE transfer at the fine level is beneficial, both in terms of iteration counts and overall computational costs. Furthermore, this approach results in a multigrid algorithm whose convergence rates are independent of the interior penalty factor. This leads to multigrid coarsening strategies denoted as chp- or cph-multigrid and illustrated in Figure 5.2, with a transfer to continuous (c) function spaces on the finest level followed by geometric (h) and polynomial (p) coarsening before the coarse-level solver (e.g., AMG) is invoked. This work discusses design choices in the context of hybrid multigrid algorithms, where computational costs act as the driving force for algorithmic selections. Other relevant factors which are not considered explicitly here are memory requirements or aspects of parallel scalability. For example, the use of an additional c-transfer or mixed-precision multigrid is justified by a significant increase in throughput and is assumed to outweigh the disadvantage of increased memory requirements.

5.2 A hybrid multigrid preconditioner

This section studies the constant coefficient Poisson equation as a model problem

$$-\nabla^2 u = f \text{ in } \Omega \subset \mathbb{R}^d \,,$$

with Dirichlet boundary condition $u = g$ on Γ^{D} and Neumann boundary conditions $\nabla u \cdot \boldsymbol{n} = h$ on Γ^{N}, where $\Gamma^{\mathrm{D}} \cup \Gamma^{\mathrm{N}} = \Gamma$ and $\Gamma^{\mathrm{D}} \cap \Gamma^{\mathrm{N}} = \emptyset$, as well as $\Gamma^{\mathrm{D}} \neq \emptyset$ for well-posedness. As introduced in Chapter 2, a mesh of hexahedral elements is considered and L^2-conforming function spaces are considered

$$\mathcal{V}_h = \left\{ u_h \in L^2(\Omega_h) \ : \ u_h \left(\boldsymbol{x}^e(\boldsymbol{\xi}) \right) |_{\Omega_e} = \tilde{u}_h^e(\boldsymbol{\xi})|_{\tilde{\Omega}_e} \in \mathcal{V}_{h,e} = \mathcal{Q}_p(\tilde{\Omega}_e) \ \forall e \right\} \ , \tag{5.1}$$

where the polynomial degree is denoted by p in this section for consistency with the term p-multigrid that has found widespread use in the multigrid context. The shape functions are tensor products based on the LGL nodes and a tensor product quadrature rule based on Gauss points with $p+1$ points per coordinate direction is used. A high-order polynomial mapping $\boldsymbol{x}^e(\boldsymbol{\xi})$ may be used. Discretization is based on the SIPG method, to obtain the weak formulation: Find $u_h \in \mathcal{V}_h$ such that

$$a_h^e \left(v_h, u_h \right) = \left(v_h, f \right)_{\Omega_e} \quad \forall v_h \in \mathcal{V}_{h,e} \ , e = 1, ..., N_{\mathrm{el}} \ , \tag{5.2}$$

where $a_h^e \left(v_h, u_h \right)$ is given by equation (2.124). In matrix-vector notation, the discrete problem can be written as the linear system of equations

$$\mathbf{A}\mathbf{u} = \mathbf{b} \ , \tag{5.3}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$, $\mathbf{u}, \mathbf{b} \in \mathbb{R}^N$ with the problem size (number of unknowns or degrees of freedom) denoted by $N = N_{\mathrm{el}}(p+1)^d$. Contributions from inhomogeneous boundary conditions are included in the right-hand side vector $\mathbf{b}$ and the matrix $\mathbf{A}$ only accounts for the homogeneous part $a_{h,\mathrm{hom}}^e \left(v_h, u_h \right)$. For the multigrid algorithm detailed below, coarser discretizations of the Laplace operator are required, which is realized by evaluating the operator for modified discretization parameters h and p (including the interior penalty parameter), i.e., on a coarser mesh or for a lower polynomial degree p. In the literature, this approach is sometimes denoted as re-discretization, as opposed to a Galerkin product. Further, a transfer from discontinuous to continuous function spaces is considered in the hybrid multigrid algorithm. The continuous finite element (FE) space is given as

$$\mathcal{V}_h^{\mathrm{FE}} = \left\{ u_h \in H^1(\Omega_h) \ : \ u_h \left(\boldsymbol{x}(\boldsymbol{\xi}) \right) |_{\Omega_e} = \hat{u}_h^e(\boldsymbol{\xi})|_{\tilde{\Omega}_e} \in \mathcal{V}_{h,e} = \mathcal{Q}_p(\tilde{\Omega}_e) \ , \ \forall e \right\} \ , \tag{5.4}$$

and the weak form of the (negative) Laplace operator simplifies to

$$a_{h,\mathrm{FE}}^e \left(v_h, u_h \right) = \left(\nabla v_h, \nabla u_h \right)_{\Omega_e} \ . \tag{5.5}$$

Dirichlet boundary conditions are imposed strongly for the FE discretization, but only the homogeneous operator is required inside the multigrid algorithm.

Iterative solvers and multigrid smoothers only require the action of $\mathbf{A}$ applied to a vector, so that an explicit assembly of the matrix $\mathbf{A}$ is avoided by the use of matrix-free operator evaluation (see Chapter 4). The operator only needs to be assembled on the coarsest multigrid level for the AMG coarse-level solver. When assembling the coarse-level matrix, constrained degrees of freedom are kept in the system with diagonal entries set to 1 to ensure that the matrix is invertible.

Algorithm 5.1 Preconditioned conjugate gradient algorithm (solves $\mathbf{A}\mathbf{x} = \mathbf{b}$ to given tolerance)

function SOLVERCG($\mathbf{A}, \mathbf{x}, \mathbf{b}$)
  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
  $\|\mathbf{r}_0\| = \|\mathbf{r}\| = \text{NORM}(\mathbf{r})$
  $\mathbf{v} = \mathbf{P}^{-1}\mathbf{r}$ ▷ e.g., MULTIGRIDVCYCLE($L, \mathbf{A}, \mathbf{0}, \mathbf{r}$)
  $\mathbf{p} = \mathbf{v}$
  $\delta = \mathbf{r}^\mathsf{T}\mathbf{v}$
  while $\|\mathbf{r}\|/\|\mathbf{r}_0\| > \varepsilon_{\text{rel}}$ **and** $\|\mathbf{r}\| > \varepsilon_{\text{abs}}$ **do**
    $\mathbf{v} = \mathbf{A}\mathbf{p}$
    $\omega = \delta/(\mathbf{p}^\mathsf{T}\mathbf{v})$
    $\mathbf{x} \leftarrow \mathbf{x} + \omega\mathbf{p}$
    $\mathbf{r} \leftarrow \mathbf{r} - \omega\mathbf{v}$
    $\|\mathbf{r}\| = \text{NORM}(\mathbf{r})$
    $\mathbf{v} = \mathbf{P}^{-1}\mathbf{r}$ ▷ e.g., MULTIGRIDVCYCLE($L, \mathbf{A}, \mathbf{0}, \mathbf{r}$)
    $\delta' = \delta$
    $\delta = \mathbf{r}^\mathsf{T}\mathbf{v}$
    $\beta = \delta/\delta'$
    $\mathbf{p} \leftarrow \mathbf{v} + \beta\mathbf{p}$
  end while
end function

Algorithm 5.2 Multigrid V-cycle (solves $\mathbf{A}\mathbf{x} = \mathbf{b}$ approximately)

function MULTIGRIDVCYCLE($l, \mathbf{A}^{(l)}, \mathbf{x}^{(l)}, \mathbf{b}^{(l)}$)
  if $l = 0$ **then**
    $\mathbf{x}^{(0)} \leftarrow \text{COARSELEVELSOLVER}(\mathbf{A}^{(0)}, \mathbf{x}^{(0)}, \mathbf{b}^{(0)})$ ▷ coarse-level solver, e.g., AMG
    return $\mathbf{x}^{(0)}$
  else
    $\mathbf{x}^{(l)} \leftarrow \text{SMOOTH}(\mathbf{A}^{(l)}, \mathbf{x}^{(l)}, \mathbf{b}^{(l)}, n_{\text{s}})$ ▷ pre-smoothing
    $\mathbf{r}^{(l)} = \mathbf{b}^{(l)} - \mathbf{A}^{(l)}\mathbf{x}^{(l)}$ ▷ calculate residual
    $\mathbf{b}^{(l-1)} = \mathbf{R}^{l-1,l}\mathbf{r}^{(l)}$ ▷ restriction
    $\mathbf{x}^{(l-1)} \leftarrow \text{MULTIGRIDVCYCLE}(l-1, \mathbf{A}^{(l-1)}, \mathbf{0}, \mathbf{b}^{(l-1)})$ ▷ coarse-level correction
    $\mathbf{x}^{(l)} \leftarrow \mathbf{x}^{(l)} + \mathbf{P}^{l,l-1}\mathbf{x}^{(l-1)}$ ▷ prolongation
    $\mathbf{x}^{(l)} \leftarrow \text{SMOOTH}(\mathbf{A}^{(l)}, \mathbf{x}^{(l)}, \mathbf{b}^{(l)}, n_{\text{s}})$ ▷ post-smoothing
    return $\mathbf{x}^{(l)}$
  end if
end function

5.2.1 Multigrid preconditioned Krylov solver

The basic multigrid idea is to tackle oscillatory errors by smoothing and to tackle smooth errors by coarse-grid corrections, which applies to all types of multigrid coarsening (geometric, polynomial, algebraic) discussed here. In this work, multigrid is used as a preconditioner inside a Krylov solver instead of using multigrid as a solver (see for example Benzi et al. (2005) for an introduction to and overview of Krylov solvers). This approach is sometimes also denoted

Algorithm 5.3 Chebyshev-accelerated Jacobi smoother (solves $\mathbf{Ax} = \mathbf{b}$ approximately)

1: **function** CHEBYSHEVSMOOTHER($\mathbf{A}, \mathbf{x}_0, \mathbf{b}, n_\mathrm{s}$)
2: **for** $j = 0, \ldots, n_\mathrm{s} - 1$ **do**
3: $\mathbf{x}_{j+1} = \mathbf{x}_j + \sigma_j \left(\mathbf{x}_j - \mathbf{x}_{j-1}\right) + \theta_j \mathbf{D}^{-1} \left(\mathbf{b} - \mathbf{A}\mathbf{x}_j\right)$
4: **end for**
5: **return** $\mathbf{x}_{n_\mathrm{s}}$
6: **end function**

as a Krylov-accelerated multigrid method. The combination of a multigrid cycle with a Krylov method can be expected to be more robust and to result in lower iteration counts in general as opposed to pure multigrid solvers, see Lottes and Fischer (2005), Shahbazi et al. (2009), Stiller (2016, 2017a,b), Sundar et al. (2015), especially for anisotropic problems. This strategy appears to be the most frequent use case in practice, and is therefore followed in the present work. Some performance numbers could be improved by alternative multigrid flavors, e.g., by considering full multigrid cycles (Kronbichler and Ljungkvist 2019), where only a single or only few iterations are needed on the finest level. Due to the symmetric positive definite nature of the model problem considered here, the conjugate gradient (CG) algorithm (Hestenes and Stiefel 1952, Saad 2003) is used as Krylov solver, which is detailed in Algorithm 5.1. The algorithmic components which are of main interest are the application of the discretized operator in line 8 and the application of the preconditioner in line 13. Other components are vector update operations and inner products (involving global communcation), but these parts of the algorithm are overall of subordinate importance since the computational costs are mainly determined by operator evaluation and the multigrid cycle called in the preconditioning step. However, it should be pointed out that the costs of all parts of the algorithm are explicitly taken into account by the experimental cost measures used in the present work, in the spirit of parallel textbook multigrid efficiency (Gmeiner et al. 2015b), see the performance metrics defined in Section 5.3.1.

In the preconditioning step of the conjugate gradient solver (preconditioner $\mathbf{P}$), the operator $\mathbf{A}$ is inverted approximately by performing one multigrid V-cycle according to Algorithm 5.2 with initial solution $\mathbf{x}^{(L)} = \mathbf{0}$, where L denotes the finest level. Pre- and post-smoothing are done in lines 6 and 11, respectively, and the residual evaluation in line 7. The same number of smoothing steps n_s is used for both pre- and post-smoothing and for all multigrid levels $0 < l \leq L$. These steps typically form the most expensive part of the multigrid algorithm as long as the workload in degrees of freedom per core is sufficiently large, i.e., away from the strong-scaling limit where latency effects become dominant. The coarse-level correction is called in line 9, recursively calling the multigrid V-cycle for the next coarser level $l - 1$ until the coarsest level $l = 0$ is reached, on which the coarse-level solver is called (line 3). Restriction (operator $\mathbf{R}^{l-1,l}$) and prolongation (operator $\mathbf{P}^{l,l-1}$) are done in lines 8 and 10, respectively.

5.2.2 Chebyshev-accelerated Jacobi smoother

In the context of matrix-free methods analyzed here, an attractive multigrid smoother is a Chebyshev-accelerated Jacobi smoother (Adams et al. 2003), which requires the diagonal $\mathbf{D}$ of the operator $\mathbf{A}$ as well as the application of the matrix-vector product $\mathbf{Au}$. Therefore, any fast implementation for the evaluation of the discretized operator can be applied inside the smoother

and parallel scalability is directly inherited from the operator evaluation. Algorithm 5.3 details the Chebyshev iteration with iteration index j and n_s smoothing steps, where the two additional scalar parameters σ_j and θ_j are calculated according to the theory of Chebyshev polynomials and require an estimation of the maximum eigenvalue λ_max of $\mathbf{A}$. The parameters are determined such that the Chebyshev smoother tackles eigenvalues in the range $[0.06\lambda_\mathrm{max}, 1.2\lambda_\mathrm{max}]$ on the current level, while smaller eigenvalues are damped by the coarse-grid correction. Since the maximum eigenvalue is only estimated, a safety factor of 1.2 is included to ensure robustness of the smoother. Note that the precise value used for the lower bound is not critical in terms of robustness and iteration counts. A Chebyshev iteration with n_s pre- and post-smoothing steps is denoted as Chebyshev($n_\mathrm{s}, n_\mathrm{s}$) in the following. Although iteration counts decrease continuously for increasing number of smoothing steps, the overall solve time is comparably insensitive w.r.t. the particular value of n_s as investigated, e.g., in Fehn et al. (2020). Typical values for which the smoother is most efficient are $n_\mathrm{s} = 4, \dots, 6$ for mixed-precision multigrid (Section 5.2.5), and a default parameter of $n_\mathrm{s} = 5$ is used in the present work. While the constant-coefficient Poisson case is considered in this section, it should be mentioned that Chebyshev smoothing has been reported to work well also for variable-coefficient problems with smoothly varying coefficient (Kronbichler and Wall 2018, Sundar et al. 2015).

The diagonal required by the Chebyshev smoother is calculated in the setup phase and has storage requirements comparable to the DoF vector $\mathbf{u}$. The maximum eigenvalue needed by the Chebyshev iteration is estimated by performing 20 conjugate gradient iterations. Compared to a single solution of the linear system, this cost is not negligible. However, for many large-scale time dependent problems where thousands to millions of time steps have to be solved, setup costs are amortized after a few time steps. The setup costs are proportional to the costs of a fine-level matrix-vector product, and therefore increase similarly under mesh refinement as the solution of the linear system of equations itself. Hence, these costs are not considered explicitly here, and the reader is referred to Kronbichler and Ljungkvist (2019) for details on setup costs.

5.2.3 Coarsening strategies and multigrid transfer operations

The multigrid level l introduced in Algorithm 5.2 is uniquely defined by the grid size h, the polynomial degree p, and the continuity parameter $c \in \{\mathrm{DG}, \mathrm{FE}\}$

$$l = f(h, p, c) \; .$$

From one multigrid level to the next, only one of the three parameters may change for the hybrid multigrid methods discussed in this work. For example, a transfer from DG space to FE space leads to two multigrid levels that coincide with respect to grid size h and polynomial degree p, i.e., a combined coarsening from high-order discontinuous space to low-order continuous space is not considered here. The approach is denoted as h-/p-multigrid if geometric/polynomial coarsening is employed only. Combined geometric-polynomial multigrid is denoted as hp- or ph-multigrid, depending on which coarsening is applied first, as illustrated in Figure 5.1. Following this notation and depending on whether the DG-to-FE transfer is performed at high degree or at low degree, the approach is denoted as cp-multigrid or pc-multigrid, respectively, or as ch-multigrid or hc-multigrid if geometric coarsening is involved. Applying all three possibilities for coarsening would for example result in a cph-multigrid strategy, with the c-coarsening performed

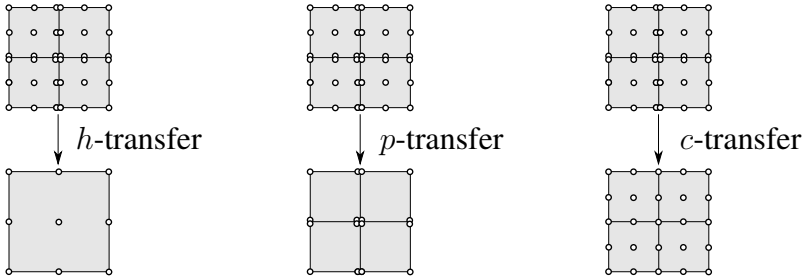


Figure 5.3: Illustration of elementary coarsening strategies for nodal high-order discontinuous Galerkin discretizations.

first, followed by p-coarsening and finally h-coarsening. The different types of coarsening are illustrated in Figure 5.3. In all cases, algebraic multigrid may be applied as a coarse-level solver.

To introduce a common notation for all types of transfers $t \in \{h, p, c\}$, the prolongation of the coarse-level correction from coarse to fine levels is generically written as

$$\mathbf{u}^{(l)} = \mathbf{P}^{l,l-1}\mathbf{u}^{(l-1)} = \sum_{e=1}^{N_{\text{el}}^{(l)}} \mathbf{S}_e^l \mathbf{P}_e^{l,l-1} \mathbf{G}_e^{l-1} \mathbf{u}^{(l-1)} \,, \tag{5.6}$$

where the global prolongation operator is expanded into the sum over all elements on the fine level with elementwise prolongation operator $\mathbf{P}_e^{l,l-1}$. The gather operator $\mathbf{G}_e^{l-1}$ extracts local degrees of freedom of a coarse-level element from the global DoF vector. The scatter operator $\mathbf{S}_e^l$ writes local degrees of freedom into the global fine-level DoF vector and additionally performs a weighting of degrees of freedom according to the multiplicity of a shared node in case of continuous function spaces. The elementwise prolongation operator is realized as L^2-orthogonal projection

$$\left(v_h^{(l)}, \hat{u}_h^{(l)}\right)_{\tilde{\Omega}_e^{(l)}} = \left(v_h^{(l)}, \hat{u}_h^{(l-1)}\right)_{\tilde{\Omega}_e^{(l)}} \rightarrow \mathbf{P}_e^{l,l-1} = \left(\mathbf{M}_e^l\right)^{-1} \mathbf{M}_e^{l,l-1} \,, \tag{5.7}$$

where $\mathbf{M}_e^l$ denotes the mass matrix and $\mathbf{M}_e^{l,l-1}$ the embedding from space $l-1$ into l. Note that the integral is performed in reference space over the fine-level element $\tilde{\Omega}_e^{(l)}$. Therefore, the operation is the same for all elements and is done only once in the setup phase where the 1D prolongation matrices are constructed. Prolongation in multiple dimensions is constructed as the tensor product of 1D operations, exploiting fast matrix-free evaluation techniques. The 1D prolongation matrices represent the interpolation of coarse-level basis functions into the nodes of the fine-level basis functions. In the case of h-coarsening and for general mappings from reference to physical space, however, the coarse-level space is no longer a subset of the fine-level space. Therefore, the chosen multigrid transfer operations implicitly introduce the approximation of nested function spaces as also mentioned, e.g., in Lu et al. (2014). In case of p-transfer and c-transfer, the function spaces are "strictly" nested. Restriction of the residual $\mathbf{r}$ onto coarser levels is defined as the transpose of prolongation,

$$\mathbf{r}^{(l-1)} = \mathbf{R}^{l-1,l}\mathbf{r}^{(l)} = \left(\mathbf{P}^{l,l-1}\right)^{\mathsf{T}} \mathbf{r}^{(l)} = \sum_{e=1}^{N_{\text{el}}^{(l)}} \left(\mathbf{G}_e^{l-1}\right)^{\mathsf{T}} \left(\mathbf{P}_e^{l,l-1}\right)^{\mathsf{T}} \left(\mathbf{S}_e^l\right)^{\mathsf{T}} \mathbf{r}^{(l)} \,. \tag{5.8}$$

5.2.3.1 h-coarsening

A hierarchy of h-levels is constructed based on the octree concept, see for example Burstedde et al. (2011), Sundar et al. (2012) for details on aspects of the chosen mesh topology. Therefore, coarser meshes in the multigrid context are not obtained by coarsening a fine mesh, but rather by starting from a coarse mesh that is refined uniformly several times to obtain the fine mesh. This coarse mesh also forms the coarse-grid problem in the multigrid algorithm. From this perspective, it is clear that pure h-multigrid based on the octree approach works well for cube-like domains of moderate geometrical complexity, but reaches limitations for complex geometries where only one or two refinement levels applied to the coarse mesh might be affordable in practice. In these cases, it is essential to further coarsen the problem by the use of p-multigrid and algebraic multigrid techniques described in more detail below. In the context of this thesis, meshes without hanging nodes are considered and each octree has the same number of mesh refinement levels. Multigrid methods for high-order discretizations on adaptively refined meshes are discussed in Janssen and Kanschat (2011), Kanschat (2004), Kronbichler and Wall (2018), Kronbichler and Ljungkvist (2019), Sundar et al. (2012) in a pure h-multigrid context.

5.2.3.2 p-coarsening

As opposed to h-multigrid, p-multigrid offers the possibility for various p-coarsening strategies. Reducing the polynomial degree by one, $p_{l-1} = p_l - 1$, is frequently applied in literature (Antonietti et al. 2015, Bassi et al. 2009, 2011, Bassi and Rebay 2003, Diosady and Darmofal 2009, Fabien et al. 2019, Fidkowski and Darmofal 2004, Fidkowski et al. 2005, Ghidoni et al. 2014, Nastase and Mavriplis 2006b, Shahbazi et al. 2009). An alternative is to reduce the polynomial degree by a factor of two, $p_{l-1} = p_l/2$ (with appropriate rounding operation), which has been used in Helenbrook et al. (2003), Helenbrook and Atkins (2008), Helenbrook and Mascarenhas (2016), Mascarenhas et al. (2009, 2010). This coarsening strategy has a close analogy to h-coarsening since the number of degrees of freedom is reduced by a factor of two in each coordinate direction from one level to the next. It is also not uncommon to immediately reduce the polynomial degree to its minimum, $p_{l-1} = 0$ or $p_{l-1} = 1$ for all p_l (two-level algorithm), see for example Bastian et al. (2019), Persson and Peraire (2008), Rasetarinera and Hussaini (2001). Elementwise constant shape functions with $p_{l=0} = 0$ are not considered in this work. On the one hand, the present DG discretization is not consistent for polynomial degree $p = 0$ on general meshes and the chosen definition of the penalty parameter, see Bastian et al. (2019) under which cirumstances the $p = 0$ DG discretization resembles a consistent finite volume discretization. On the other hand, as argued in Helenbrook and Atkins (2008), the small-wave-number modes that remain after smoothing are essentially continuous for diffusive problems and are, therefore, not well represented by a piecewise constant coarse space with $p = 0$. For the neutron diffusion problems studied in O'Malley et al. (2017), a continuous $p = 1$ coarse space has been found to be advantageous over a piecewise constant space. It has been observed in Persson and Peraire (2008), Shahbazi et al. (2009) by the example of the compressible Navier–Stokes equations involving diffusive terms that $p_{l=0} = 1$ performs better than $p_{l=0} = 0$. A piecewise constant space with $p_{l=0} = 0$ is typically used in the convection-dominated limit and the compressible Euler equations (Luo et al. 2006, Nastase and Mavriplis 2006b). In Bastian et al. (2019), $p_{l=0} = 0$ is also used for a Poisson problem with variable coefficients. These previous studies indicate that

the optimal coarse space depends on the model problem under investigation. For the constant-coefficient Poisson problem considered here, the analysis is therefore restricted to a specific choice $p_{l=0} = 1$ for the coarse space. Discussions and comparisons of different p-sequences can be found in Helenbrook and Atkins (2008) in the context of iteration counts and in Nastase and Mavriplis (2006b) in terms of iteration counts and computational costs. In that work, only a single polynomial degree of $p = 4$ is investigated. Here, a more rigorous investigation of the following p-coarsening strategies is fostered

- $p_{l-1} = p_l - 1$ (decrease by one),
- $p_{l-1} = \lfloor p_l/2 \rfloor$ (bisection),
- $p_{l-1} = 1 \, \forall p_l$ (two-level algorithm),

considering a wide range of polynomial degrees p and studying the impact on both iteration counts and computational costs. Generally, all p-levels are exploited in the proposed multigrid algorithm until $p = 1$ is reached.

5.2.3.3 c-coarsening (transfer from discontinuous to continuous space)

A transfer from the discontinuous space to a continuous space at the coarse degree $p = 1$ is used in Helenbrook and Atkins (2008), O'Malley et al. (2017), an idea that has already been described in Lasser and Toselli (2001) in the context of two-level overlapping preconditioners. A transfer at the highest degree p is suggested in Rudi et al. (2015) without justification and with results shown only for the lowest degree $p = 1$. This approach might be counter-intuitive at first sight since an additional multigrid level at high polynomial degree (and therefore with expensive smoothers) is introduced and the problem size is only marginally reduced for a DG-to-FE transfer at high degree, i.e., by a factor of $(1 + 1/p)^d$. It is interesting to note that a similar idea called conforming aggregation is used in Olson and Schroder (2011) in the context of smoothed aggregation algebraic multigrid techniques, where degrees of freedom at the same spatial location are aggregated on the finest level. For the two-level scheme proposed in Bastian et al. (2012, 2019), Dobrev et al. (2006), Siefert et al. (2014), the high-order DG space is directly reduced to a linear conforming space. This could be the reason for the strong increase in iteration counts observed in Bastian et al. (2019), Siefert et al. (2014) for increasing p (and for a similar two-level preconditioner used in Remacle et al. (2016)). As in O'Malley et al. (2017), an additional multigrid level is introduced in the present work for the DG-to-FE transfer, i.e., the transfer to the continuous FE space is done at constant degree p and mesh size h and it is found that this is important for optimal multigrid convergence rates. Two variants of the DG-to-FE transfer are investigated in this work, namely performing this transfer at highest degree or lowest degree $p = 1$ (and similarly on the finest mesh or coarsest mesh). Performing the transfer to continuous elements on the finest level has very attractive properties. It reduces the iteration counts considerably, and yields a multigrid solver for SIPG discretizations of the Poisson equation that is robust w.r.t. the penalty parameter of the SIPG method. Theoretical background for this behavior is provided in Antonietti et al. (2017), where this approach is motivated from the perspective of space splitting and auxiliary space methods. The important difference is that this splitting is here integrated into multigrid with the same smoother used on all levels.

The elementwise prolongation matrix is an identity matrix in the case of a DG-to-FE transfer since the continuous and discontinuous function spaces are the same from an elementwise perspective. Accordingly, the degrees of freedom shared by neighboring elements in the continuous case are simply injected into the degrees of freedom duplicated in the discontinuous case. With restriction being the transposed operation, the residual contributions of degrees of freedom of duplicated nodes in the discontinuous case are summed into the uniquely defined degree of freedom in the continuous case. The c-transfer can easily be extended to adaptively refined meshes with hanging nodes according to $2:1$ refinements (not covered in the present work), but requires more advanced techniques for generally non-conforming meshes, for which DG discretizations might be considered a more natural approach.

Remark 5.1 *The two-level approaches in Antonietti et al. (2017), Dobrev et al. (2006), Lasser and Toselli (2001) are also known or interpreted as auxiliary space preconditioning. This nomenclature is not used in the present work. Instead, these approaches are categorized as one type of multigrid coarsening in the generalized framework of hybrid multigrid algorithms. The multigrid methods in Bastian et al. (2012), Siefert et al. (2014) are introduced as algebraic multigrid methods that are "not fully algebraic". The present work fosters a fine-level point of view and categorizes these approaches as p-multigrid (potentially with additional c-coarsening) with algebraic multigrid applied as coarse-level solver; for good reasons, because the fine levels are those where the numerical method spends its time (assuming that the method is applied away from the strong-scaling limit) and are those that determine the computational efficiency of the approach.*

5.2.4 Coarse-level solver

The coarse-level problem (sometimes also denoted as coarse-grid problem) is usually small, and it therefore seems to be a reasonable choice to use an iterative Krylov solver with a simple preconditioner such as point-Jacobi as coarse-grid solver. However, the convergence of such an iterative scheme is often poor as soon as the coarse grid becomes more complex, and the large number of global communication steps due to inner products in the coarse-grid solver typically limits parallel scalability. A remedy to the latter aspect is to use the Chebyshev iteration as coarse grid solver with a fixed number of iterations. The number of iterations of the Chebyshev solver is determined a-priori according to the eigenvalue distribution of the coarse-level matrix, and is estimated in a way that the Chebyshev error estimator reaches a given tolerance (Varga 2009). This procedure has been used in Krank et al. (2017), Kronbichler and Wall (2018), but it turned out that this coarse-grid solver forms a bottleneck for more complex geometries, especially for large polynomial degrees in combination with a pure h-multigrid method.

The bottleneck of expensive coarse-grid solvers can be removed by the use of algebraic multigrid methods. Note that an AMG V-cycle does not necessarily converge at the same rate as the smoothers on the geometric levels of the multigrid hierarchy would allow to. However, the success of multigrid methods originates from the fact that the coarse-grid correction ensures mesh-independent convergence rates as well as low iteration counts and – at the same time – causes only low computational overhead as compared to the operations on the fine level. It is therefore important that the coarse-grid correction does not deteriorate the multigrid convergence rate, which should only be affected by the smoother on the fine level. For this reason, it is reasonable

to solve the coarse-level problem by an inner Krylov solver preconditioned by an AMG V-cycle to a specified tolerance, instead of using a single AMG V-cycle as coarse-grid solver. Note that applying a Krylov solver within the preconditioner does no longer guarantee that the preconditioner is a stationary operation, which might in general require the use of flexible solvers such as FGMRES (Saad 1993). Nevertheless, a basic CG iteration is used as outer Krylov solver for the numerical examples in this chapter since no convergence problems have been observed. Extending AMG solvers designed for continuous discretizations to the discontinuous case is not trivial without further measures as shown in Olson and Schroder (2011). Since the goal is to apply the AMG coarse-grid solver in a black-box fashion in its optimal regime, performance numbers are mainly shown for AMG applied to a continuous discretization of the coarse problem with lowest degree $p = 1$, see also Bastian et al. (2012), Siefert et al. (2014). In the present work, the AMG implementation provided by the `Trilinos ML` project (Gee et al. 2006) is used, applying one V-cycle with one smoothing step of an `ILU` smoother without fill-in and no overlap (i.e., in a block-Jacobi fashion over the MPI ranks) and an `Amesos-KLU` coarse solver unless specified otherwise. A comparative study of different AMG solver frameworks is beyond the scope of this work, and is for example shown in O'Malley et al. (2017) for the neutron diffusion equation, or in Offermans et al. (2019) in the context of computational fluid dynamics.

5.2.5 Mixed-precision multigrid

The matrix-free implementation outlined in Chapter 4 is perfectly suited for mixed-precision computations in the multigrid preconditioned Krylov solver, following the idea of Gropp et al. (2000). This is due to the fact that the amount of data transferred from main memory reduces by a factor of two in case of single precision (implying twice the throughput in terms of elements processed per time), and the vectorization strategy with explicit vectorization over elements/faces also allows twice the throughput in terms of arithmetics. The throughput of the matrix-vector product therefore increases by a factor of approximately 2 when reducing accuracy from double precision to single precision. To not spoil accuracy of the numerical solution and to ensure convergence of the outer Krylov solver, single precision is only used in the multigrid V-cycle. The outer Krylov solver operates in double precision. Larger round-off errors in the multigrid cycle can be tolerated since these high-frequency errors introduced by single-precision round-off errors are tackled by the multigrid smoothers (Kronbichler and Ljungkvist 2019) and since multigrid is only a preconditioner applied to the residual of the outer Krylov solver, see Algorithm 5.1. Since the `Trilinos ML` solver used here only supports double precision, the AMG coarse-grid preconditioner operates in double precision. The performance of mixed-precision is compared to pure double-precision computations in Section 5.3.

5.3 Numerical results for hybrid multigrid solver

This section shows numerical results for the hybrid multigrid solver. Relevant performance metrics used to evaluate the efficiency of multigrid solvers are introduced in Section 5.3.1. Information on the hardware under consideration is given in Section 5.3.2. The considered test cases are briefly summarized in Section 5.3.3, before numerical results are shown for each problem in the subsequent sections.

5.3.1 Performance metrics

Frequently used metrics are the average convergence rate ρ and the number of iterations n_{10} needed to reduce the residual by ten orders of magnitude ($\varepsilon_{10} = \|\mathbf{r}_{n_{10}}\|_2/\|\mathbf{r}_0\|_2 = 10^{-10}$)

$$\rho = \left(\frac{\|\mathbf{r}_n\|_2}{\|\mathbf{r}_0\|_2}\right)^{\frac{1}{n}} \ , \ n_{10} = \frac{\log_{10}\left(\|\mathbf{r}_{n_{10}}\|_2/\|\mathbf{r}_0\|_2\right)}{\log_{10}\rho} = \frac{-10}{\log_{10}\rho} \ ,$$

where $\mathbf{r}_n$ denotes the residual after n iterations. These quantities are well suited to demonstrate mesh-independent convergence rates, or to quantitatively investigate robustness of the multigrid method, i.e., the influence of certain parameters such as mesh anisotropies, variable coefficients, or the polynomial degree on the convergence behavior of the multigrid algorithm. However, they are not suited to quantify the effectiveness of smoothers in terms of computational efficiency. To measure computational costs, theoretical measures such as operation counts required for the matrix-vector product or matrix nonzeros are often considered (Lottes and Fischer 2005, Sundar et al. 2015). These quantities should be considered with some care because they inherently contain assumptions on the bottleneck (for example that the algorithm is compute-bound so that floating point operations are really a cost measure). However, this depends on many aspects such as the hardware under consideration (Flop-to-Byte ratio), the implementation strategy (matrix-based vs. matrix-free), and the optimization level of the implementation. Due to these uncertainties and model assumptions of theoretical cost measures, experimental cost measures determined from the actual performance of the multigrid solver are preferred here, in the spirit of Bastian et al. (2019), Kronbichler and Wall (2018). An effective number of fine-level matrix-vector products, denoted as $n_{10,\mathbf{Au}}$ in the following, is helpful to incorporate computational costs for the smoother and to compare different smoothers in the metric of computational costs instead of global iteration counts. For example, it is unclear whether more aggressive smoothers achieving lower iteration counts are also superior in terms of time-to-solution. The quantity $n_{10,\mathbf{Au}} = t_{\text{wall},\mathbf{u}=\mathbf{A}^{-1}\mathbf{b}(\varepsilon_{10})}/t_{\text{wall},\mathbf{Au}}$ is defined as the ratio of the wall time for one application of the linear solver with tolerance ε_{10} over the wall time for one operator evaluation. Since absolute wall times depend on the problem size, it is useful to express $n_{10,\mathbf{Au}}$ as a function of two normalized quantities. The first one is the efficiency $E_{\mathbf{Au}}$ of the matrix-free operator evaluation measured as the number of degrees of freedom N processed per second per core (also denoted as throughput as introduced in Chapter 4)

$$E_{\mathbf{Au}} = \frac{N}{t_{\text{wall},\mathbf{Au}} N_{\text{cores}}} \ . \tag{5.9}$$

The second one is the time t_{10} required by the multigrid solver to solve one degree of freedom per core with a residual reduction of ε_{10}

$$t_{10} = \frac{t_{\text{wall},\mathbf{u}=\mathbf{A}^{-1}\mathbf{b}(\varepsilon_{10})} N_{\text{cores}}}{N} \ , \tag{5.10}$$

or equivalently the throughput $E_{10} = 1/t_{10}$ of the linear solver in degrees of freedom solved per second per core. Then, the effective number of fine-level matix-vector products is determined experimentally as

$$n_{10,\mathbf{Au}} = t_{10} E_{\mathbf{Au}} = \frac{E_{\mathbf{Au}}}{E_{10}} \ . \tag{5.11}$$

Table 5.2: Performance metrics used to evaluate the efficiency of multigrid solvers.

Quantity	description
n_{10}	number of iterations to reduce the residual by $\varepsilon_{10} = 10^{-10}$
t_{10}	wall time in seconds to solve one unknown per core to reach $\varepsilon_{10} = 10^{-10}$
E_{10}	throughput of solver in unknowns solved per second per core ($= 1/t_{10}$)
$E_{\mathbf{Au}}$	throughput of matrix-free operator evaluation in DoFs per second per core
$n_{10,\mathbf{Au}}$	effective number of fine-level mat-vec products ($= E_{\mathbf{Au}}/E_{10}$) to reach ε_{10}

The aim of $n_{10,\mathbf{Au}}$ is to obtain a measure for the algorithmic complexity of the whole multigrid solver, but as independent of hardware and absolute performance numbers as possible. Table 5.2 summarizes the used performance metrics.

Remark 5.2 *The definition of $n_{10,\mathbf{Au}}$ has similarities with the parallel textbook efficiency factor defined in Gmeiner et al. (2015b), with the difference that one fine-level matrix-vector product is used as work unit here, instead of one fine-level smoothing step. The overall goal is optimizing time-to-solution and the operator evaluation* **Au** *is the only algorithmic component re-occurring for practically all iterative solvers and preconditioners. Choosing* **Au** *as work unit ensures that the algorithmic complexity of different smoothers is reflected in the values achieved for $n_{10,\mathbf{Au}}$. The performance advantage achieved by the use of mixed-precision multigrid is also naturally included in this metric.*

5.3.2 Hardware

The numerical experiments shown in this section are performed on an Intel Skylake architecture with AVX512 vectorization, see Table 4.2. The GNU compiler `g++` version 7.3 with optimization flags `-O3 -funroll-loops -std=c++17 -march=skylake-avx512` is used. All computations are done on thin nodes unless specified otherwise. The present analysis focuses mainly on the node-level performance because multigrid solvers are well-known to be scalable even to the largest supercomputers (Gholami et al. 2016, Kronbichler and Wall 2018). The multigrid communication is between nearest neighbors, both horizontally within the matrix-vector products and vertically between the multigrid levels with one round-trip per V-cycle through the coarse solver, assuming the latter is sufficiently cheap. This is backed up by performance projections to exascale machines where multigrid is expected to be primarily memory-limited within the nodes (Ibeid et al. 2020). Good parallel scalability up to high core counts on large supercomputers when using AMG coarse-grid solvers is shown in Offermans et al. (2016, 2019).

5.3.3 Test cases

The hybrid multigrid methods are investigated for a series of test cases with increasing complexity regarding the geometry and the number of elements on the coarse grid, as well as the

maximum aspect ratio defined as

$$\mathrm{AR} = \max_{e=1,\dots,N_{\mathrm{el}}} \left(\max_{\Omega_e} \frac{\sigma_1(\boldsymbol{J}^e)}{\sigma_d(\boldsymbol{J}^e)} \right) , \tag{5.12}$$

where σ_1 and σ_d are the largest and smallest singular values of the Jacobian matrix $\boldsymbol{J} = \partial \boldsymbol{x} / \partial \boldsymbol{\xi}$, which is evaluated at all quadrature points of the element. The following problems are considered, where a visualization of the geometries and the meshes of the different test cases is shown in Figure 5.4:

- *Cube*: the geometry is a unit cube with $\mathcal{O}(10^1) - \mathcal{O}(10^2)$ elements on the coarse grid. This geometry could also be discretized with a single element on the coarse grid, but configurations with $2^d, 3^d, 5^d$ elements on the coarse grid are considered to test all multigrid components and to make sure that the coarse-grid problem is non-trivial (but very small). This test case is well-suited to test the different multigrid ingredients, identify optimal multigrid coarsening strategies, perform parameter studies, study the impact of Cartesian and curved elements on iteration counts and throughput, and to compare the present implementation to the state-of-the-art (since data is mainly available for simple cube-like geometries in the literature).

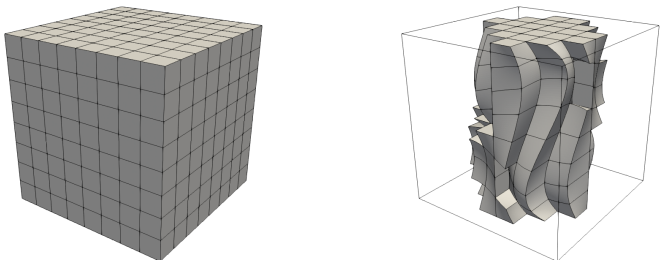
- *Nozzle*: the geometry of this test case is the nozzle geometry of the FDA benchmark, which has been designed to assess CFD solvers for the simulation of the flow through medical devices (Malinauskas et al. 2017). The geometry is a tube with gradual or sudden contractions/expansions of the cross section area, inducing separating flows and involving laminar, transitional, and turbulent flow regimes. A coarse-grid mesh with $\mathcal{O}(10^3)$ elements is used in the present work. The tube and cone geometries are known analytically and used for high-order representations of the geometry via manifolds (using a cubic mapping). The blood flow through this device can be modeled as an incompressible flow, and the pressure Poisson component of the related incompressible Navier–Stokes solver is investigated here. The mesh contains high-aspect-ratio elements with a moderate distortion, especially in the outflow part of the domain on the right. A comprehensive LES study of this flow configuration using the present high-order discontinuous Galerkin discretization approach has been performed in Fehn et al. (2019b).

- *Lung*: The most complex test case studied here is the geometry of the upper airway generations of the human lung, using a patient-specific geometry of a preterm infant, for which gas exchange mechanisms have been investigated recently in the literature (Roth et al. 2018). The geometry is discretized with a purely hexahedral mesh and the coarse-grid problem consists of $\mathcal{O}(10^4)$ elements for 8 airway generations. Simulating the flow of air through the human lung as a numerical solution of the incompressible Navier–Stokes equations again involves the solution of a pressure Poisson equation, the model problem studied in this section.

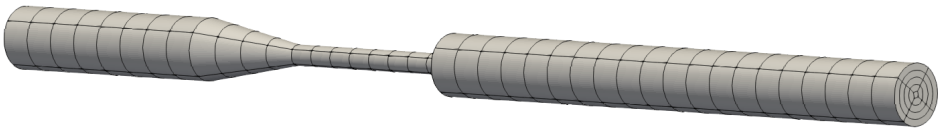
5.3.4 Cube

A simple analytical test case with solution

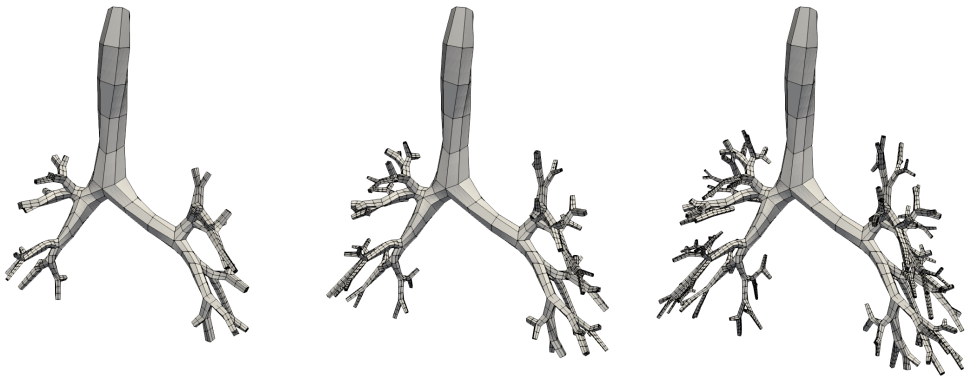
$$u(\boldsymbol{x}) = \sin(3\pi x_1)\sin(3\pi x_2)(3\pi x_3) \tag{5.13}$$



(a) Cube: Cartesian mesh (left) and section of curvilinear mesh (right) with 8^3 elements ($h/L = 1/8$) and aspect ratios of $\mathrm{AR} = 1.0$ and 2.9, respectively.



(b) Nozzle: coarse mesh h_0 of FDA nozzle geometry consisting of 440 elements ($\mathrm{AR} \approx 9.2$).



(c) Lung: coarse mesh h_0 of a patient-specific geometry of the human lung of a preterm infant for 6, 7, and 8 airway generations (from left to right) with 1968, 4236, and 9396 elements, where the mesh with 8 generations has an aspect ratio of approximately $\mathrm{AR} = 67$.

Figure 5.4: Visualization of geometries and meshes investigated for the hybrid multigrid solver. The size of the coarse-grid problem ranges from $\mathcal{O}(10^1)$ to $\mathcal{O}(10^4)$ elements.

is considered on a cube geometry in 3D, $\Omega = [-1, 1]^3$. Dirichlet boundary conditions are prescribed on the domain boundary using the known analytical solution. The right-hand side is determined according to the method of manufactured solutions

$$f(\boldsymbol{x}) = -\nabla^2 u(\boldsymbol{x}) = 27\pi^2 \sin(3\pi x_1) \sin(3\pi x_2)(3\pi x_3) \ . \tag{5.14}$$

Two types of meshes are analyzed, a Cartesian mesh and a curvilinear mesh with deformation

$$d(\boldsymbol{x}) = a \sin(2\pi(x_1 + 1)/2) \sin(2\pi(x_2 + 1)/2)(2\pi(x_3 + 1)/2) \tag{5.15}$$

in each coordinate direction. An amplitude of $a = 0.15$ is used, leading to elements that are deformed significantly, see Figure 5.4. For the curvilinear mesh with deformation manifold,

Table 5.3: Cube test case: iteration count n_{10} as a function of polynomial degree p for various multigrid coarsening strategies for 3D *Cartesian* mesh.

(a) h-multigrid (with $p_{l-1} = \lfloor p_l/2 \rfloor$ if p-transfer is involved)

MG type	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h	14.8	12.5	12.4	12.2	14.4	14.8	17.2	17.5	19.5	19.1	22.3	22.4	24.3	24.5	26.2
hpc	14.8	12.5	12.4	12.2	14.4	14.9	17.2	17.5	19.6	19.1	22.3	22.4	24.2	24.6	26.1
chp	7.5	5.5	5.2	5.1	5.2	5.1	5.5	5.6	6.6	6.7	7.8	7.8	8.8	8.9	9.8

(b) p-multigrid ($p_{l-1} = p_l - 1$)

MG type	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	3.3	12.4	11.2	11.2	11.3	10.5	11.0	11.4	11.8	11.5	12.5	12.6	12.8	13.3	13.5
phc	14.8	12.5	11.3	11.3	11.3	10.7	10.9	11.5	11.8	11.6	12.5	12.8	13.3	13.6	13.8
cph	7.5	5.5	5.1	4.9	4.8	5.0	4.7	4.7	4.6	4.7	4.6	4.7	4.6	4.8	4.9

(c) p-multigrid ($p_{l-1} = \lfloor p_l/2 \rfloor$)

MG type	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	3.3	12.4	13.0	11.9	14.2	14.1	15.9	15.4	17.9	16.9	20.1	19.4	21.3	22.3	24.3
phc	14.8	12.5	13.9	12.0	14.3	14.2	16.0	15.3	17.8	16.9	20.2	19.8	21.6	21.9	23.9
cph	7.5	5.5	5.1	4.9	5.1	4.8	5.0	5.1	5.6	5.4	6.3	6.3	7.1	7.0	7.8

(d) p-multigrid ($p_{l-1} = 1 \, \forall p_l$)

MG type	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	3.3	12.4	13.0	16.6	20.3	24.6	29.4	35.6	41.6	44.9	55.8	64.6	74.0	86.9	96.8
phc	14.8	12.5	13.9	16.8	20.7	24.7	29.8	35.6	41.6	44.8	54.7	65.2	75.2	88.6	97.0
cph	7.5	5.5	5.1	5.2	6.6	8.7	10.7	12.9	15.5	17.4	19.9	22.5	24.6	26.9	29.7

element mappings of polynomial degree 3 are used independently of the polynomial degree of the shape functions. The smoother used for all experiments is Chebyshev(5,5) and the coarse-grid problem is solved iteratively to a relative tolerance of 10^{-3} by the conjugate gradient method with AMG V-cycle as preconditioner.

5.3.4.1 Robustness with respect to p-refinement

In a first numerical experiment, the number of iterations is investigated as a function of the polynomial degree p for various multigrid coarsening strategies. Table 5.3 lists the results obtained for the Cartesian mesh and Table 5.4 the results obtained for the curvilinear mesh. The tables distinguish between h-like MG approaches where additional p-coarsening is done on the coarsest h-level (hp-MG), and p-like MG approaches where additional h-coarsening is done at lowest degree $p = 1$ (ph-MG). While the three different p-coarsening strategies from Section 5.2.3.2 are considered for the p-like approaches, the h-like approaches exclusively use the p-coarsening

Table 5.4: Cube test case: iteration count n_{10} as a function of polynomial degree p for various multigrid coarsening strategies for 3D *curvilinear* mesh.

(a) h-multigrid (with $p_{l-1} = \lfloor p_l/2 \rfloor$ if p-transfer is involved)

MG type	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h	17.7	13.8	14.0	15.0	17.8	19.4	22.6	22.6	25.4	26.4	28.9	29.8	32.3	33.3	35.8
hpc	17.7	13.8	14.0	15.0	17.8	19.4	22.6	23.1	25.3	26.3	28.8	29.7	32.3	32.9	35.5
chp	8.5	5.9	5.5	5.5	5.9	6.6	7.9	8.7	10.1	10.7	12.1	12.6	13.9	14.4	15.7

(b) p-multigrid ($p_{l-1} = p_l - 1$)

MG type	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	3.4	13.7	12.8	13.0	13.8	15.0	15.5	16.3	16.6	16.7	17.7	17.9	18.7	18.8	19.6
phc	17.7	14.2	13.2	13.1	13.9	15.1	15.5	16.3	16.7	16.8	17.8	18.0	18.7	18.7	19.7
cph	8.5	5.9	5.4	5.3	5.3	5.2	5.2	5.3	5.8	5.9	6.5	6.7	7.4	7.6	7.9

(c) p-multigrid ($p_{l-1} = \lfloor p_l/2 \rfloor$)

MG type	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	3.4	13.7	15.3	14.0	18.5	19.5	22.2	21.7	25.1	25.9	29.1	29.2	33.0	33.8	36.5
phc	17.7	14.2	16.4	14.1	18.6	19.5	22.3	21.9	25.1	25.9	29.1	29.3	32.8	33.7	36.6
cph	8.5	5.9	5.9	5.5	6.5	6.3	7.8	7.8	9.5	9.7	10.9	11.3	12.6	12.7	13.9

(d) p-multigrid ($p_{l-1} = 1 \, \forall p_l$)

MG type	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	3.4	13.7	15.3	20.3	27.0	34.7	39.2	45.9	54.0	63.7	76.4	94.9	115	137	157
phc	17.7	14.2	16.4	21.0	27.0	34.4	39.1	45.9	53.4	62.9	75.8	93.6	113	135	156
cph	8.5	5.9	5.9	8.4	11.7	15.2	18.9	23.0	27.0	31.4	35.9	40.3	44.9	49.5	54.2

denoted as bisection that approximately halves the number of unknowns per direction from one multigrid level to the next. A fixed number of elements of 8^3 is used and the polynomial degree varies between $1 \leq p \leq 15$. If h-multigrid is involved, the coarse-grid problem consists of 2^3 elements. For brevity, only a subset of all possible combinations of multigrid coarsening strategies are listed in Tables 5.3 and 5.4. Additional results are shown in the publication related to this chapter (Fehn et al. 2020), and comments on additional results can also be found in the text. The results can be summarized as follows:

- Extending the pure h- or p-multigrid methods towards hybrid multigrid methods with additional p- or h-coarsening, respectively, on coarser levels does not change the multigrid convergence rates. The multigrid convergence rates are also not altered if additional c-coarsening is performed at the coarsest level before the coarse-grid solver is invoked (hc-, hpc- and pc-, phc-approaches) or at an intermediate level between h- and p-coarsening (hcp- and pch-approaches).

- A different convergence behavior with much lower iteration counts is observed when performing the c-transfer on the finest level before h- or p-coarsening is invoked. For all multigrid approaches and for both Cartesian and curvilinear meshes, iteration counts are reduced by a factor of 2 to 3 compared to c-coarsening performed on coarser levels. Performing the c-transfer introduces additional costs as quantified in Section 5.3.4.3.

- With respect to p-robustness, the h-like approaches on the one hand and the p-like approaches with $p_{l-1} = \lfloor p_l/2 \rfloor$ coarsening on the other hand show a similar relation between polynomial degree and iteration counts. This is not unexpected, as both approaches reduce the degrees of freedom in factors of two per direction per level. In combination with the Chebyshev smoother considered here, these approaches show a slight increase in iteration counts for large p.

- The p-multigrid methods with $p_{l-1} = 1 \, \forall p_l$ coarsening show a much stronger increase in iteration counts for large p. The results shown here also shed light on previous results in Bastian et al. (2019), Remacle et al. (2016), Siefert et al. (2014), where two-level approaches with an immediate transfer from highest to lowest polynomial degree have been used. As shown in the following, this coarsening strategy is not only performing worst in terms of iteration counts, but also in terms of computational costs.

- The p-multigrid methods with $p_{l-1} = p_l - 1$ coarsening show the best behavior in terms of p-robustness w.r.t. iteration counts. On the Cartesian mesh, the iteration counts are completely independent of p for the cp- and cph-approaches, and the number of iterations increases only slightly for increasing p on the curvilinear mesh. However, this type of p-coarsening is also the most complex one introducing the largest numbers of multigrid levels. Hence, from the results shown in Tables 5.3 and 5.4, it is unclear whether this strategy pays off in terms of computational costs, an aspect investigated in detail in Section 5.3.4.3.

Additional h-robustness tests performed in Fehn et al. (2020) demonstrate mesh-independent convergence rates for the 3D Cartesian and curvilinear meshes and the different multigrid coarsening strategies. Iteration counts under mesh refinement are also shown below for the nozzle and lung test cases.

5.3.4.2 Robustness with respect to interior penalty parameter

The coarsening strategies with c-transfer on the finest level (such as cph-, chp-coarsening) have the interesting property that the resulting multigrid algorithm exhibits convergence rates that are independent of the penalty factor of the interior penalty method. This property is demonstrated in Table 5.5, where the cph-multigrid method is compared to combined hp- and ph-multigrid methods without c-transfer (pure p- and pure h-multigrid methods would show a qualitatively similar behavior). As expected, for standard hp- and ph-coarsening, the iteration counts degrade significantly when increasing the interior penalty factor, while the cph-multigrid method shows constant iteration counts when scaling the penalty factor, equation (2.122), by $\zeta_{\text{IP}} = 10^0, 10^1, 10^2, 10^3$. The chp-multigrid approach also shows robustness with respect to the interior penalty parameter τ, and is not shown explicitly in Table 5.5. Qualitatively, the same results are

Table 5.5: Cube test case: Robustness of multigrid algorithm w.r.t. interior penalty parameter on a 3D Cartesian mesh with 8^3 elements. The table lists the iteration count n_{10}.

(a) hp-multigrid ($p_{l-1} = \lfloor p_l/2 \rfloor$)

IP factor ζ_{IP}	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10^0	14.8	12.5	12.4	12.2	14.4	14.9	17.2	17.5	19.6	19.1	22.3	22.4	24.2	24.6	26.1
10^1	25.4	32.6	39.9	39.7	46.8	45.4	52.6	51.8	55.8	56.5	62.0	62.3	67.9	68.5	73.0
10^2	38.5	53.8	79.9	83.7	109	104	128	117	134	133	147	146	157	166	176
10^3	45.0	73.3	113	123	172	162	205	190	223	194	221	219	243	249	278

(b) ph-multigrid ($p_{l-1} = \lfloor p_l/2 \rfloor$)

IP factor ζ_{IP}	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10^0	14.8	12.5	13.9	12.0	14.3	14.2	16.0	15.3	17.8	16.9	20.2	19.8	21.6	21.9	23.9
10^1	25.4	29.8	33.8	33.2	38.2	38.4	43.2	42.6	48.7	46.4	52.3	52.2	56.7	57.7	61.3
10^2	38.5	45.2	49.6	52.2	61.7	63.4	67.5	70.7	80.0	77.8	85.7	85.5	96.4	93.3	102
10^3	45.0	59.1	66.0	70.9	79.4	80.1	89.0	94.8	108	105	116	119	126	125	138

(c) cph-multigrid ($p_{l-1} = \lfloor p_l/2 \rfloor$)

IP factor ζ_{IP}	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10^0	7.5	5.5	5.1	4.9	5.1	4.8	5.0	5.1	5.6	5.4	6.3	6.3	7.1	7.0	7.8
10^1	7.7	5.4	5.3	5.3	5.4	5.2	5.3	5.6	5.7	5.7	6.4	6.5	7.4	7.2	8.0
10^2	7.7	5.4	5.3	5.4	5.5	5.4	5.4	5.7	5.8	5.8	6.5	6.5	7.2	7.2	8.1
10^3	7.7	5.4	5.4	5.4	5.5	5.4	5.4	5.7	5.9	5.9	6.9	6.8	7.6	7.8	8.8

obtained when repeating this experiment for the 3D curvilinear mesh. An explanation for this τ-robustness might be that the continuous finite element space covers the DG solution in the limit of large penalty factors, thereby balancing the deteriorating conditioning of the DG operator, see also the theory in Antonietti et al. (2017). In other words, the interior penalty parameter does not only impact the conditioning, but also the approximation properties of the DG solution in relation to the continuous FE space. This behavior is appealing as it allows to avoid the need to optimize the IP parameter in order to obtain iteration counts as low as possible and, at the same time, ensure coercivity of the SIPG discretization.

5.3.4.3 Identification of optimal multigrid sequence maximizing throughput

The results in Section 5.3.4.1 revealed that using more p-levels in the multigrid hierarchy reduces the number of iterations, at the costs of increased computational load per iteration. However, it remains unclear which type of p-coarsening is the most efficient one. Likewise, it needs to be investigated whether a c-transfer at the finest level (with an additional expensive smoothing step performed on the finest level as opposed to a cheap c-transfer at an intermediate level or at the coarsest level) reduces overall computational costs, i.e., algorithmic selections should be driven by the time-to-solution metric.

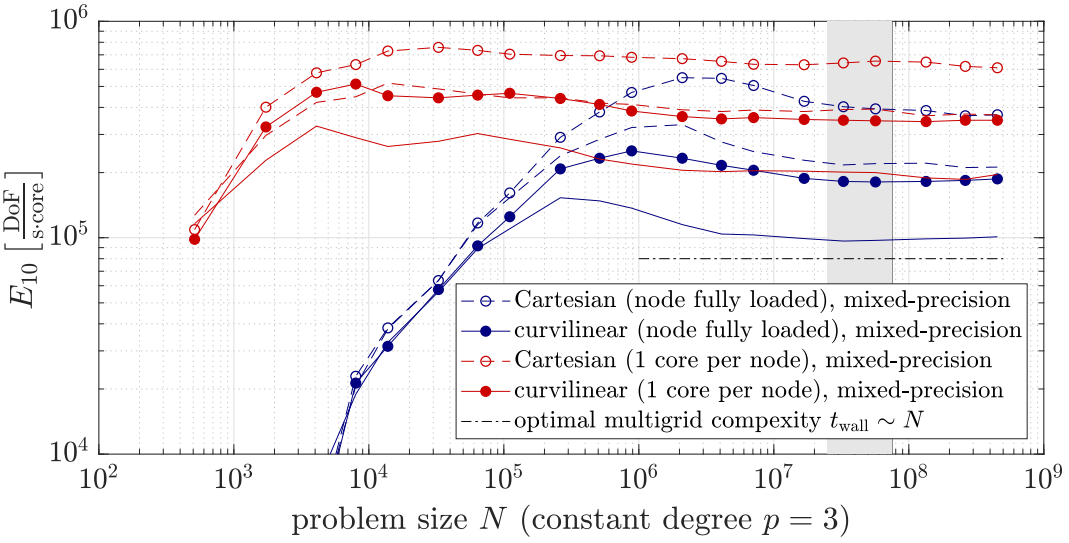
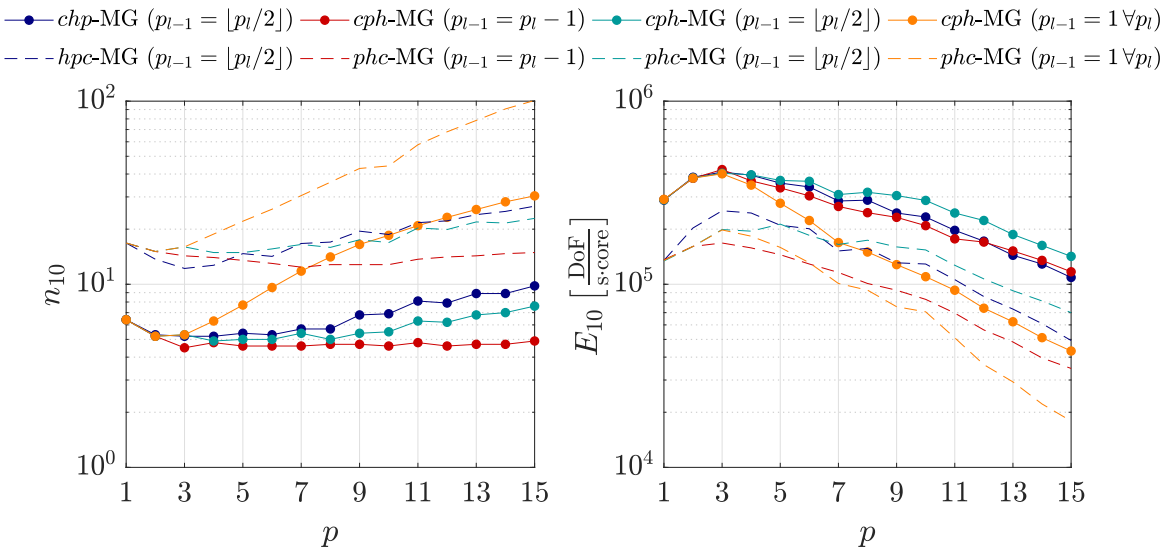


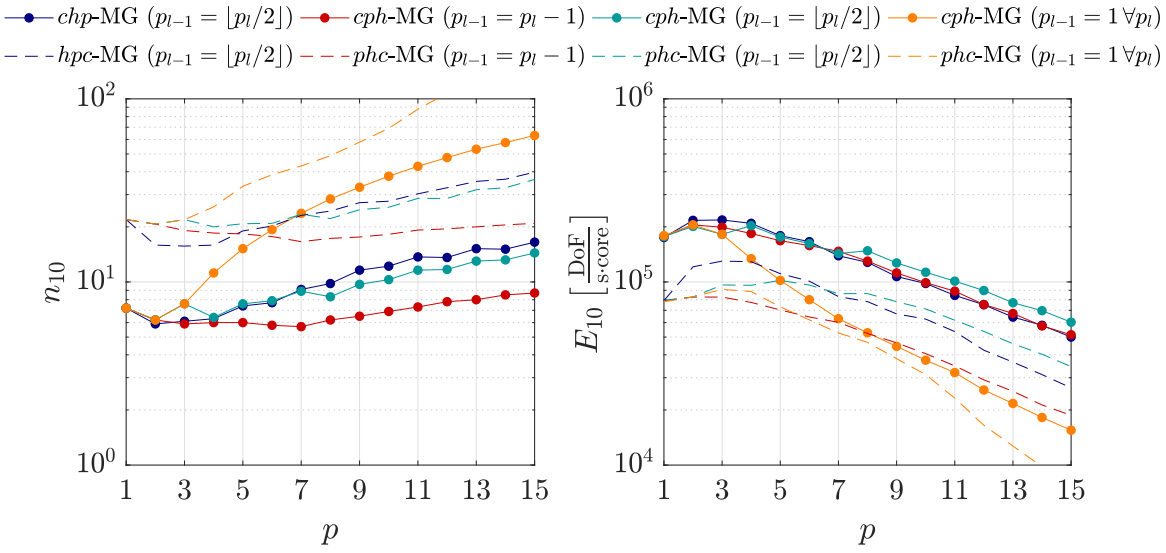
Figure 5.5: Throughput E_{10} versus problem size for polynomial degree $p = 3$ and cube test case with Cartesian mesh (dashed lines) and curvilinear mesh (solid lines). A *cph*-multigrid coarsening strategy is used with $p_{l-1} = \lfloor p_l/2 \rfloor$. Standard mixed-precision multigrid results are shown as lines with markers, and additional computations performed in double precision only are shown as lines without markers. A fat memory node is used here in order to investigate a wide range of problem sizes. The gray band indicates the range of problem sizes used for the throughput measurements in Figure 5.6, for which a fully loaded node (blue curves) is considered with the problem size large enough to saturate caches.

For throughput measurements, it is important to fully utilize all cores of one compute node since certain resources are shared by the cores of a node, i.e., the performance reported in degrees of freedom solved per second per core would otherwise be extraordinarily high. This is demonstrated in Figure 5.5, where the throughput is significantly larger if only a single core is utilized per node instead of a fully loaded node. Figure 5.5 also shows the speed-up that can be achieved by the use of mixed-precision multigrid, which is around a factor of 1.8 for large problem sizes. Towards very small problem sizes (strong-scaling limit), the performance breaks down since performance is limited by latency and the available parallelism instead of arithmetic throughput or memory throughput, and the performance advantage of mixed-precision multigrid therefore vanishes in such a scenario. For the computations on a fully-loaded node, an elevation of the throughput can be observed for problem sizes around 1 MDoF due to the fact that data fits (partly) into caches, which have a higher bandwidth than main memory. Throughput measurements are therefore run in a saturated regime of sufficiently high workload per core so that the data does no longer fit into caches. In Figure 5.5, the throughput is shown as a function of the problem size to highlight these cache effects. The range of problem sizes (25 MDoF to 75 MDoF) used below for benchmarking the present solver is indicated by a gray band. While it is good practice to run the solver in a saturated regime for benchmarking, it is of course beneficial to

explicitly exploit caching effects for practical computations. Note that a constant throughput for large problem sizes implies optimal complexity of the solver in terms of $t_{\text{wall}} \sim N$.



(a) cube test case on 3D Cartesian mesh



(b) cube test case on 3D curvilinear mesh

Figure 5.6: Iterations n_{10} versus throughput E_{10} for different multigrid coarsening strategies on Cartesian mesh and curvilinear mesh. The problem size is between 25 MDoF and 75 MDoF for all polynomial degrees $1 \leq p \leq 15$.

Figure 5.6 details the performance in terms of iteration counts as well as computational efficiency for different hybrid multigrid algorithms, each of them exploiting all levels of h-, p-, and c-coarsening (in different orders). For the p-like approaches, the three different types of p-coarsening are again investigated. The results for n_{10} in the left panels of the figures visualize results similar to those shown in Tables 5.3 and 5.4. The cph- and phc-methods with $p_{l-1} = p_l - 1$

Table 5.6: Iteration count n_{10} and effective number of fine-level matrix-vector products $n_{10,\mathbf{Au}}$ for Cartesian mesh versus curvilinear mesh in 3D. The cph-multigrid method with $p_{l-1} = \lfloor p_l/2 \rfloor$ is used. The problem size is between 25 MDoF and 75 MDoF for all polynomial degrees $1 \leq p \leq 15$. The efficiency measures $E_{\mathbf{Au}}$ and E_{10} are specified in $\frac{\text{MDoF}}{\text{s·core}}$.

(a) 3D Cartesian mesh

	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_{10}	6.4	5.2	5.3	4.9	5.0	5.0	5.4	5.0	5.4	5.5	6.3	6.2	6.8	7.0	7.6
$n_{10,\mathbf{Au}}$	90	102	97	95	106	94	99	99	92	93	100	85	91	87	93
$E_{\mathbf{Au}}$	26.0	39.0	39.0	37.7	39.0	34.3	30.5	31.4	28.1	26.7	24.5	18.9	17.0	14.2	13.2
E_{10}	0.29	0.38	0.40	0.40	0.37	0.37	0.31	0.32	0.31	0.29	0.25	0.22	0.19	0.16	0.14

(b) 3D curvilinear mesh

	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_{10}	7.2	6.2	7.6	6.4	7.6	7.9	8.9	8.3	9.7	10.3	11.6	11.7	13.0	13.2	14.4
$n_{10,\mathbf{Au}}$	94	99	109	98	119	122	127	122	133	144	152	140	147	143	158
$E_{\mathbf{Au}}$	16.7	19.8	19.9	20.0	21.0	19.7	18.2	18.1	16.9	16.3	15.3	12.5	11.3	10.0	9.5
E_{10}	0.18	0.20	0.18	0.20	0.18	0.16	0.14	0.15	0.13	0.11	0.10	0.09	0.08	0.07	0.06

coarsening exhibit a constant number of iterations for large p on the Cartesian mesh, and a slight increase in the number of iterations for the curvilinear mesh. The results in Figure 5.6 highlight that performing the c-transfer on the finest level is not only beneficial in terms of iteration counts, but also in terms of computational costs. The chp- and cph-multigrid methods clearly outperform the hpc- and phc-multigrid methods on the Cartesian mesh as well as on the curvilinear mesh in this experiment. In terms of p-coarsening, the $p_{l-1} = 1 \, \forall p_l$ strategy performs worst both in terms of iteration counts and computational costs. The $p_{l-1} = \lfloor p_l/2 \rfloor$ strategy performs best in terms of computational costs. The differences between hp- versus ph-multigrid methods are very small, and small differences in the number of iterations determine which approach is more efficient overall. Despite exhibiting the lowest number of iterations, the $p_{l-1} = p_l - 1$ strategy is not competitive if the c-transfer is done at the coarse level. However, it is interesting to realize that the $p_{l-1} = p_l - 1$ strategy can keep up with the $p_{l-1} = \lfloor p_l/2 \rfloor$ coarsening strategy if the c-transfer is performed on the fine level. Going through all polynomial degrees in the multigrid hierarchy introduces less overhead in this case since the operator evaluation is significantly faster for the continuous FE space (e.g., no face integrals) compared to the DG space (Kronbichler and Wall 2018). Although not explicitly shown here, it should be mentioned that the increased number of multigrid levels for the $p_{l-1} = p_l - 1$ strategy is disadvantageous due to increased memory requirements, and also in the strong-scaling limit where overall costs are dominated by the latency of matrix-vector products (and hence the number of multigrid levels). For these reasons, the chp- and cph-multigrid methods with $p_{l-1} = \lfloor p_l/2 \rfloor$ coarsening strategy are considered most promising. These multigrid methods are investigated below for the more challenging test cases.

In terms of absolute numbers, a maximum throughput of up to $E_{10} = 0.41 \, \frac{\text{MDoF}}{\text{s·core}}$ or, equivalently, a minimum solve time of $t_{10} = 2.4 \, \frac{\mu\text{s·core}}{\text{DoF}}$ at degree $p = 3$ is achieved for the Carte-

sian mesh. The performance is reduced for the curvilinear mesh, with a maximum throughput of $E_{10} = 0.22\ \frac{\text{MDoF}}{\text{s·core}}$ and a minimum solve time of $t_{10} = 4.6\ \frac{\mu\text{s·core}}{\text{DoF}}$ at degree $p = 3$. The reduced performance for the curvilinear mesh compared to the Cartesian mesh can be explained by an increase in iteration counts on the one hand, and a reduced throughput of the matrix-free operator evaluation on the other hand, as summarized in Table 5.6. In addition to previous results, Table 5.6 lists the effective number of fine-level matrix-vector products $n_{10,\textbf{Au}}$. For the Cartesian mesh, $n_{10,\textbf{Au}} \approx 100$ is obtained, i.e., solving the linear system of equations to a relative tolerance of $\varepsilon_{10} = 10^{-10}$ corresponds to the costs of 100 fine-level matrix-vector products. For the curvilinear mesh, $n_{10,\textbf{Au}} \approx 100 - 150$ is obtained with the effective number of matrix-vector products increasing for higher p. To put these numbers into perspective, the cost per iteration is equivalent to $10 - 20$ fine-level matrix-vector products, while the iterative scheme performs one double-precision matrix-vector product in the CG solver, 10 single-precision matrix-vector products in the fine-level smoother in the DG space and the same number in continuous space, plus additional work on coarser levels as well as vector operations.

5.3.4.4 Comparison to state-of-the-art

To give a perspective on the achieved performance, the present hybrid multigrid solver is compared to state-of-the-art implementations from the literature in the metric t_{10}. Note that the different results vary by the degree of optimization and specialization in the algorithms and implementations, so the primary intent is to provide a point of reference rather than a benchmark:

- In Stiller (2017b), the Poisson equation is solved using an interior penalty DG discretization with collocation approach on a 3D Cartesian mesh using overlapping Schwarz smoothers. A solve time of $t_{10} \approx 7\frac{\mu\text{s}}{\text{DoF}}$ is achieved for $p = 4$ run on a 3.1 GHz Intel Core i7-5557U CPU (one core used). Including the difference in throughput between partially loaded and fully loaded nodes according to Figure 5.5, the present approach can be considered significantly faster.

- In Huismann et al. (2019), the Poisson equation is solved on a 3D Cartesian mesh using a collocation variant of the continuous spectral element method. A solve time of $t_{10} \approx 10\frac{\mu\text{s·core}}{\text{DoF}}$ for $p = 3$ and $t_{10} \approx 5\frac{\mu\text{s·core}}{\text{DoF}}$ for $p = 4$ is specified in that work, where simulations have been run on a single core of a node composed of two Intel Xeon E5-2590-v3 with 12 cores each. A parallel efficiency for a fully loaded node between 52% and 65% is specified in (Huismann et al. 2019, Table 2) for a Krylov-accelerated MG solver. This aspect needs to be taken into account and increases solve times roughly by a factor of two, see also the results in Figure 5.5. For moderately high polynomial degrees $p \leq 5$, the present approach with $t_{10} \approx 2.5 - 3\ \frac{\mu\text{s·core}}{\text{DoF}}$ is therefore significantly more efficient, despite the fact that the implementation in Huismann et al. (2019) uses optimizations that are restricted to Cartesian meshes and the fact that a computationally cheaper continuous finite element discretization is used. Somewhat orthogonally, the approach in Huismann et al. (2019) is clearly faster for very large polynomial degrees such as $p = 16$, for which solve times as low as $t_{10} \approx 1\frac{\mu\text{s·core}}{\text{DoF}}$ when using a single core are specified in that work.

- In Bastian et al. (2019), an interior penalty DG discretization is considered for the constant coefficient Poisson problem on a 3D Cartesian geometry using block-Jacobi smoothers. A

maximum performance of $t_8 \geq 1.33 \frac{\mu s}{\text{DoF}}$ is achieved at degree $p = 2$ on 16 cores of an Intel Xeon E5-2698v3 node, corresponding to $t_{10} = 26.6 \frac{\mu s \cdot \text{core}}{\text{DoF}}$. Compared to the performance numbers specified above, the present approach is approximately one order of magnitude faster. It should be emphasized in this context that the smoothers used in Bastian et al. (2019) are more complex and designed for variable-coefficient problems. At the same time, these results demonstrate that a conservative selection of smoothers with a focus on robustness for potentially more complex PDEs is clearly non-optimal.

- In Kronbichler and Wall (2018), a constant-coefficient Poisson problem is solved on a 3D Cartesian geometry for an interior penalty DG discretization using a pure h-multigrid approach with Chebyshev smoother of degree 2. A minimal solve time of $t_9 = 2.1 \frac{\mu s \cdot \text{core}}{\text{DoF}}$, or equivalently $t_{10} = 2.33 \frac{\mu s \cdot \text{core}}{\text{DoF}}$, is achieved at degree $p = 4$, comparable to what is achieved in the present work, albeit on older hardware but using matrix-free kernels that are further optimized compared to the present study. In Kronbichler et al. (2019), an optimized code-version of the h-multigrid method from Kronbichler and Wall (2018) using so called element-wise face integrals and merged vector operations achieves a solve time as low as $t_9 \geq 1.1 \frac{\mu s \cdot \text{core}}{\text{DoF}}$ (or equivalently $t_{10} \geq 1.25 \frac{\mu s \cdot \text{core}}{\text{DoF}}$) on a hardware comparable to the present study. These optimizations have not been included in the present study since they have not been available in the `deal.II` library by the time of writing, but indicate further performance improvements of the present hybrid multigrid methods once these optimizations are integrated.

- Finally, the present DG solver with matrix-free evaluation and sum-factorization is compared to matrix-based hybridizable DG solvers that appear attractive since the HDG approach reduces the global matrix size considerably by eliminating interior degrees of freedom and solving a linear system of equations for the trace variable living on the element boundaries only. In Yakovlev et al. (2016), a Helmholtz-like equation with constant coefficients is solved on a unit cube with 9^3 uniform hexahedral elements of degree $p = 1, \ldots, 7$ and overall costs including mesh generation and setup are reported in that work using a single core on an Intel Xeon E7-4870 processor. A direct solver is used in that work and the authors argue that such an approach is effective in serial and for the small problem sizes considered. The wall times reported in Yakovlev et al. (2016) range from 5.0 s for $p = 1$, 170 s for $p = 3$, to $16.2 \cdot 10^3$ s for $p = 7$. Here, the constant coefficient Poisson equation is solved on the same mesh, which is at least as difficult to solve as a Helmholtz-like equation when using iterative solvers. Wall times of 0.59 s for $p = 1$, 0.86 s for $p = 3$, and 4.5 s for $p = 7$ are obtained for the present solver for the whole application (including setup and postprocessing) when running the code on a single core, achieving a speed-up by a factor of 8.5 for $p = 1$, 200 for $p = 3$, and 3600 for $p = 7$ over the HDG results shown in Yakovlev et al. (2016). These results point in a similar direction as the study by Kronbichler and Wall (2018), which provides a thorough comparative study of matrix-free DG versus matrix-based HDG methods using the same hardware for both approaches.

Table 5.7: Nozzle test case: robustness and performance of hybrid multigrid solver for cph-multigrid method with $p_{l-1} = \lfloor p_l/2 \rfloor$ coarsening.

(a) iteration count n_{10}

h	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h_0	5.5	8.4	11.4	8.6	10.7	11.8	12.2	10.5	11.7	11.8	12.7	12.0	12.9	13.7	14.5
$h_0/2$	8.3	8.8	12.3	9.6	11.8	12.5	13.4	11.5	12.7	12.0	12.9	12.8	13.7	13.8	14.3
$h_0/4$	8.8	9.8	13.5	9.8	11.7	13.5	14.2	11.7	12.6	12.5	12.9	13.7	13.8	14.6	14.8

(b) throughput E_{10} in $\frac{\text{kDoF}}{\text{s·core}}$

h	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h_0	276	338	294	357	306	265	221	251	227	225	203	210	190	177	140
$h_0/2$	3.30	36.1	115	166	129	136	110	114	95.1	90.9	78.8	73.6	66.6	62.2	56.4
$h_0/4$	9.28	82.7	126	159	142	112	92.8	106	94.2	89.4	84.4	74.9	71.0	63.8	59.7

(c) relative share of AMG coarse-grid solver in % of wall time ('-' means less than 0.1%)

h	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h_0	13.3	4.1	2.0	1.0	0.7	0.4	0.3	0.2	0.2	0.1	-	-	-	-	-
$h_0/2$	4.1	6.7	11.6	6.8	3.7	2.6	1.5	0.9	0.6	0.4	0.3	0.2	0.2	0.1	0.1
$h_0/4$	11.4	6.1	5.1	2.3	1.3	0.8	0.5	0.3	0.2	0.1	0.1	-	-	-	-

(d) speed-up of cph-coarsening over phc-coarsening

h	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h_0	0.97	2.05	1.48	1.70	1.64	1.47	1.49	1.57	1.53	1.60	1.43	1.47	1.44	1.35	1.42
$h_0/2$	1.33	1.42	2.02	2.04	1.87	2.04	1.66	1.71	1.71	1.62	1.55	1.57	1.55	1.49	1.50
$h_0/4$	1.24	2.19	1.53	1.87	1.71	1.48	1.41	1.62	1.55	1.54	1.56	1.47	1.52	1.46	1.47

5.3.5 Nozzle

This section presents results for the nozzle test case. To mimic the incompressible flow case for the nozzle problem, a Dirichlet boundary condition with a constant value of 1 is prescribed at the inflow boundary on the left, and a constant value of 0 at the outflow boundary on the right. On the walls of the nozzle geometry, homogeneous Neumann boundary conditions are prescribed. To generate a coarse grid, the nozzle domain is meshed with a minimum number of elements. The coarse grid shown in Figure 5.4 consists of 440 elements and additional information on the mesh generation can be found in Fehn et al. (2019b). The coarse grid is identified as the h_0 mesh in the following, and meshes are considered that are refined once ($h_0/2$) and twice ($h_0/4$) via uniform mesh refinements of the coarse mesh. A cubic mapping is used for all computations for a high-order representation of the geometry which is described via manifold descriptions. For polynomial degrees from $p = 1, \dots, 15$, the problem size ranges from $3.5 \cdot 10^3 - 1.8 \cdot 10^6$ unknowns for mesh h_0, $2.8 \cdot 10^4 - 1.4 \cdot 10^7$ unknowns for $h_0/2$, and $2.3 \cdot 10^5 - 1.2 \cdot 10^8$ unknowns

for $h_0/4$. Computations on mesh h_0 are performed on one core due to the small problem size, on mesh $h_0/2$ on one node (48 cores), and on mesh $h_0/4$ on two nodes (96 cores). The smoother used for all experiments is Chebyshev(5,5) and the coarse-grid problem is solved iteratively to a relative tolerance of 10^{-3} by the conjugate gradient method with AMG V-cycle as preconditioner.

Table 5.7 summarizes the numerical results for the nozzle geometry of the FDA benchmark with a focus on the cph-multigrid method. In terms of iteration counts, mesh-independent convergence is observed, and a slight increase in the number of iterations for large p in agreement with previous results. Compared to the curvilinear mesh for the cube problem, the number of iterations is larger, explaining the reduced throughput E_{10} compared to the results on the curvilinear mesh for the cube geometry in Table 5.6. An increased throughput is measured on the coarse mesh h_0, since the computations are performed on a single core, see also Figure 5.5. On the finer meshes, the throughput is small for low polynomial degrees. This is due to the fact that the problem size covers a broad range from a very low to high workload per core when going from $p = 1$ to $p = 15$ for a fixed number of elements (in contrast, the number of elements has been adapted for the cube test case to obtain a similar problem size for all p).

Table 5.7 also lists the relative share of the AMG coarse-grid solver in % of the overall wall time required by the linear solver. The coarse-grid solver accounts for up to 13% of the computational costs for linear shape functions ($p = 1$), and becomes negligible in terms of computational costs for increasing polynomial degree and finer meshes. By the use of hybrid multigrid methods, the overall computational efficiency of the method is determined by the fast matrix-free operator evaluation on the finest levels as intended.

The computationally efficient cph- and chp-coarsening strategies show a similar performance for the nozzle test case both in terms of iteration counts and computational costs, so that no significant advantage of one over the other method could be identified. As shown in Table 5.7, the cph-multigrid method is more efficient than the phc-method for all polynomial degrees and meshes considered for the nozzle problem. Robustness w.r.t. the interior penalty factor is obtained for the cph-multigrid method (and similarly for chp-coarsening), while a strong increase in iteration counts is observed in case of phc-coarsening, see Fehn et al. (2020).

5.3.6 Lung

The complex lung geometry shown in Figure 5.4 is meshed with purely hexahedral elements by the use of a specialized mesh generator as well as facilities of the `deal.II` library (Arndt et al. 2020a). The patient-specific geometry of the first three generations is obtained from a segmentation of MRI scans, while higher airway generations are constructed using a recursive tree growing algorithm that mimics the true anatomy of the preterm infant and respects anatomical length and diameter ratios of airways reported for the preterm infant (Roth et al. 2018). In a first step, a 3D cylinder tree is created, which is subsequently deformed according to the patient-specific geometry of upper airway generations obtained from magnetic resonance imaging and described via B-splines. When refining the mesh, new nodes are placed correctly on the patient-specific geometry. A tri-linear mapping of the geometry is used in the present work. Also for the lung test case, the application in mind is the solution of the pressure Poisson equation as part of an incompressible Navier–Stokes solver. Therefore, a Dirichlet boundary value of 1 is prescribed at the upper boundary and homogeneous Dirichlet boundary conditions at all outlets where the

Table 5.8: Lung test case: robustness and performance of hybrid multigrid solver for cph-multigrid method with $p_{l-1} = \lfloor p_l/2 \rfloor$ coarsening.

(a) iteration count n_{10}

h	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h_0	12.3	19.3	27.6	19.9	26.8	29.4	30.7	27.9	32.6	32.9	36.5	36.4	39.5	39.4	41.9
$h_0/2$	17.6	20.0	28.9	22.6	29.6	30.8	36.8	33.7	38.6	38.0	42.5	40.7	43.0	43.9	45.7
$h_0/4$	18.5	21.0	30.9	25.9	32.6	34.6	39.7	35.9	40.7	40.5	44.4	43.7	47.9	47.4	51.5

(b) throughput E_{10} in $\frac{\text{kDoF}}{\text{s·core}}$

h	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h_0	5.47	36.8	53.2	80.9	64.7	57.3	49.0	49.8	40.9	37.8	33.2	31.3	27.9	26.5	23.5
$h_0/2$	39.8	90.4	68.3	76.3	57.4	52.1	42.0	44.2	38.0	37.2	32.3	31.2	28.2	25.3	22.6
$h_0/4$	24.2	77.8	58.1	61.8	48.2	43.6	37.3	40.1	35.6	34.2	30.6	29.3	25.2	23.5	20.4

(c) relative share of AMG coarse-grid solver in % of wall time ('-' means less than 0.1%)

h	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h_0	15.0	21.7	18.5	8.5	5.6	3.4	2.1	1.4	0.9	0.7	0.5	0.4	0.3	0.2	0.2
$h_0/2$	19.7	6.3	2.7	1.2	0.8	0.4	0.3	0.2	0.1	-	-	-	-	-	-
$h_0/4$	18.1	9.8	3.9	1.9	1.1	0.7	0.4	0.3	0.2	0.1	0.1	-	-	-	-

(d) speed-up of cph-coarsening over phc-coarsening

h	Polynomial degree p														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
h_0	0.72	1.80	1.51	1.88	1.62	1.36	1.47	1.59	1.55	1.47	1.51	1.47	1.48	1.47	1.51
$h_0/2$	1.83	2.18	1.51	1.60	1.34	1.32	1.27	1.35	1.35	1.35	1.36	1.37	1.48	1.35	1.39
$h_0/4$	1.77	2.22	1.52	1.67	1.37	1.27	1.29	1.40	1.41	1.30	1.34	1.36	1.38	1.38	1.38

airways that are resolved by this lung model end. Homogeneous Neumann boundary conditions are prescribed on all airway walls.

The problem is solved on the coarse mesh labeled h_0, and on two finer meshes $h_0/2$ and $h_0/4$ obtained via uniform mesh refinements. In the following, results are shown for the mesh resolving 8 generations of the lung with a coarse mesh consisting of 9396 elements, see Figure 5.4. The lung mesh contains bad-aspect-ratio elements so that this test case represents more practical, difficult problems. For polynomial degrees from $p = 1, \dots, 15$, the problem size ranges from $7.5 \cdot 10^4 - 3.8 \cdot 10^7$ unknowns for mesh h_0, $6.0 \cdot 10^5 - 3.1 \cdot 10^8$ unknowns for $h_0/2$, and $4.8 \cdot 10^6 - 2.5 \cdot 10^9$ unknowns for $h_0/4$. Computations on meshes h_0 and $h_0/2$ are done on one fat compute node (48 cores) and computations on mesh $h_0/4$ on 8 fat nodes (384 cores). The smoother used for all experiments is Chebyshev(5,5) and the coarse-grid problem is solved iteratively to a relative tolerance of 10^{-1} by the conjugate gradient method preconditioned by an AMG V-cycle with Chebyshev(3,3) smoother. For the lung test case, the AMG coarse-grid

preconditioner with ILU smoother lacks robustness with respect to the number of cores, so that a Chebyshev smoother is used for the AMG coarse-grid preconditioner for this problem.

Table 5.8 lists the results for the lung test case mainly focusing on the cph-multigrid method. The number of iterations n_{10} increase slightly on finer meshes, and more strongly for increasing p. The number of iterations is highest for the lung test case explaining the reduction in throughput E_{10} compared to the results for the cube geometry with curvilinear mesh. The cph-multigrid method is faster than the phc-multigrid method for all polynomial degrees due to a significant reduction in iteration counts. Regarding the interior penalty parameter, robustness is obtained for cph- and chp-coarsening, and a strong increase in iterations counts is observed, e.g., in case of phc- and hpc-coarsening. The cph- and chp-methods (and similarly the phc- and hpc-methods) perform similarly for the lung problem with a small advantage for ph-type approaches due to slightly smaller iteration counts in agreement with the results in Figure 5.6 for the cube problem. The costs of the AMG coarse-grid solver are negligible for high-order methods and the coarse-grid solver does also not form a bottleneck for the lowest polynomial degrees, demonstrating a proper design of the present multigrid algorithms by the use of hybrid coarsening strategies. Additional results are shown in Fehn et al. (2020) for pure h- and p-multigrid methods. Due to the rather complex coarse-grid problem for the lung test case, most of the time is spent in the coarse-grid solver unless the polynomial degree is very high in case of p-multigrid, or the mesh refinement level is very high in case of h-multigrid. A substantial speed-up by more than one order of magnitude is achieved by the use of hybrid multigrid techniques, demonstrating that the approach developed here becomes mandatory in order to obtain a versatile PDE solver that is efficient for a wide range of problems and spatial resolution parameters h and p.

5.4 Block preconditioners for indefinite saddle point problem

This section discusses preconditioners for incompressible flow solvers. When using a monolithic or coupled solution approach, the linear(ized) system of equations for the velocity and pressure unknowns arising from the discretization of the incompressible Navier–Stokes equations in space and time, equations (2.213) and (2.214), is an indefinite saddle point problem of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\mathrm{m}} \\ \mathbf{b}_{\mathrm{c}} \end{bmatrix} , \tag{5.16}$$

where $\mathbf{A}$ is the matrix of the velocity block and $\mathbf{B}$ is given as

$$\mathbf{B} = -\mathbf{D} = \mathbf{G}^{\mathsf{T}} . \tag{5.17}$$

As discussed in Section 2.4.2.2, the above relation is, in general, strictly valid only under the assumption of exact integration, or for certain formulations of the velocity divergence term and pressure gradient term also in case of inexact integration. Depending on the temporal treatment of the convective term (explicit or implicit), the system of equations can be further characterized as symmetric or non-symmetric. In case of an implicit treatment of the convective term, the matrix $\mathbf{A}$ is

$$\mathbf{A} = \frac{\gamma_0^n}{\Delta t_n}\mathbf{M} + \mathbf{C}_{\mathsf{lin}}\left(\mathbf{u}^{(k)}\right) + \mathbf{V} , \tag{5.18}$$

and in case of an explicit treatment of the convective term (or when considering the generalized Stokes problem)

$$\mathbf{A} = \frac{\gamma_0^n}{\Delta t_n}\mathbf{M} + \mathbf{V} \,. \tag{5.19}$$

When dealing with the numerical solution of linear systems of equations, the term Stokes problem is therefore also used even when solving the Navier–Stokes equations with an explicit treatment of the convective term. As in Section 2.5.7, this section discusses the case where the divergence and continuity penalty terms are applied in a postprocessing step and, therefore, do not occur explicitly in the saddle point problem. The idea behind is to be able to apply well-established preconditioners for the incompressible Navier–Stokes equations, originally developed for other types of discretization in space. If added to the monolithic system, the penalty terms would appear as additional contributions in the matrix $\mathbf{A}$. A block-preconditioner explicitly taking into account the grad–div stabilization term for continuous finite element discretizations has been developed in Heister and Rapin (2013).

The saddle point structure makes the numerical solution more complicated as compared to projection-type solution methods for incompressible flows (see Sections 2.5.7.3 and 2.5.7.4). An overview of the numerical solution of saddle point problems is given in Benzi et al. (2005) and literature mentioned therein. The saddle point problem is solved by the generalized minimum residual (GMRES) method (Saad and Schultz 1986) with right preconditioning in this work. The development of efficient preconditioners is the key requirement in order to obtain robust and efficient solvers. A well established approach is based on so-called block-preconditioners (Benzi et al. 2005, Elman et al. 2014). The derivation of these block preconditioners makes use of the Schur complement factorization of the saddle point matrix. The idea of block-preconditioning is to develop optimal (spectrally-equivalent) preconditioners for the velocity and pressure Schur complement blocks separately. The preconditioner for the coupled system is then obtained by combining the respective preconditioners for the velocity block and Schur complement block in a block-diagonal or block-triangular manner. This technique allows a modular implementation since the problem of constructing efficient preconditioners for the coupled system is reduced to the problem of designing efficient preconditioners for the velocity block and the Schur complement block. Moreover, projection-type and monolithic Navier–Stokes solvers can make use of the same type of preconditioners for the velocity and pressure blocks. Block preconditioners are widely used for large-scale geodynamic problems such as earth mantle convection simulations (Kronbichler et al. 2012, May et al. 2014, Rudi et al. 2015).

The starting point for the derivation of block preconditioners is to consider a block factorization of the system matrix

$$\begin{aligned}
\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{B} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} ,
\end{aligned} \tag{5.20}$$

where the Schur complement $\mathbf{S}$ is defined as

$$\mathbf{S} = -\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^{\mathsf{T}} \,. \tag{5.21}$$

Based on the above block factorization, different types of block preconditioners can be derived by neglecting parts of that factorization. In the following, three types of block preconditioners of varying complexity are considered: a block diagonal preconditioner, a block triangular preconditioner, and a preconditioner based on an exact block triangular factorization, see Benzi et al. (2005) for details. For all of them, the main challenge lies in developing efficient preconditioners $\hat{\mathbf{A}}$ and $\hat{\mathbf{S}}$ for the velocity block $\mathbf{A}$ and the Schur complement block $\mathbf{S}$, respectively.

5.4.1 Block diagonal preconditioner

The simplest block preconditioner is obtained by only considering the diagonal blocks $\mathbf{A}$ and $\mathbf{S}$, see first row of equation (5.20)

$$\mathbf{P} = \begin{bmatrix} \hat{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & -\hat{\mathbf{S}} \end{bmatrix} \rightsquigarrow \mathbf{P}^{-1} = \begin{bmatrix} \hat{\mathbf{A}}^{-1} & \mathbf{0} \\ \mathbf{0} & -\hat{\mathbf{S}}^{-1} \end{bmatrix} \,, \tag{5.22}$$

Hence, this preconditioner consists of one application of $\hat{\mathbf{A}}^{-1}$ and $-\hat{\mathbf{S}}^{-1}$, respectively, where the minus sign in front of the Schur complement preconditioner will become clear from the following considerations.

5.4.2 Block triangular preconditioner

The block triangular preconditioner also takes into account the matrix of the discrete pressure gradient (right preconditioning)

$$\mathbf{P} = \begin{bmatrix} \hat{\mathbf{A}} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \hat{\mathbf{S}} \end{bmatrix} \rightsquigarrow \mathbf{P}^{-1} = \begin{bmatrix} \hat{\mathbf{A}}^{-1} & -\hat{\mathbf{A}}^{-1}\mathbf{B}^{\mathsf{T}}\hat{\mathbf{S}}^{-1} \\ \mathbf{0} & \hat{\mathbf{S}}^{-1} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{A}}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\hat{\mathbf{S}}^{-1} \end{bmatrix} \,, \tag{5.23}$$

In numerical experiments, this preconditioner typically leads to a significantly lower number of iterations compared to the block diagonal preconditioner. In terms of computational costs, the block triangular preconditioner is comparable to the block diagonal preconditioner since the computational costs for applying the discrete pressure gradient $\mathbf{B}$ are low as compared to approximately inverting the velocity block $\hat{\mathbf{A}}^{-1}$ and the Schur complement block $\hat{\mathbf{S}}^{-1}$, typically involving more expensive operations such as a multigrid V-cycle. Assuming optimal preconditioners for the velocity block, $\hat{\mathbf{A}}^{-1} = \mathbf{A}^{-1}$, and the Schur complement block, $\hat{\mathbf{S}}^{-1} = \mathbf{S}^{-1}$, the preconditioned system is (right preconditioning)

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \mathbf{P}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \,, \tag{5.24}$$

which follows immediately from the second row of equation (5.20).

5.4.3 Preconditioner based on block triangular factorization

Considering all terms in the block factorization (5.20) results in the following preconditioner

$$\mathbf{P} = \begin{bmatrix} \hat{\mathbf{A}} & \mathbf{0} \\ \mathbf{B} & \hat{\mathbf{S}} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \hat{\mathbf{A}}^{-1}\mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{5.25}$$

and its inverse

$$\begin{aligned} \mathbf{P}^{-1} &= \begin{bmatrix} \mathbf{I} & \hat{\mathbf{A}}^{-1}\mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} \hat{\mathbf{A}} & \mathbf{0} \\ \mathbf{B} & \hat{\mathbf{S}} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{I} & -\hat{\mathbf{A}}^{-1}\mathbf{B}^{\mathsf{T}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\hat{\mathbf{S}}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{A}}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} . \end{aligned} \tag{5.26}$$

Assuming optimal preconditioners for the velocity block, $\hat{\mathbf{A}}^{-1} = \mathbf{A}^{-1}$, and the Schur complement block, $\hat{\mathbf{S}}^{-1} = \mathbf{S}^{-1}$, equation (5.20) shows that this preconditioner approximates the system matrix exactly (implying convergence after one iteration)

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^{\mathsf{T}} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \mathbf{P}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} . \tag{5.27}$$

Compared to the block diagonal and block triangular preconditioners, this preconditioner is more expensive since applying $\mathbf{P}^{-1}$ to a vector involves two applications of the velocity preconditioner $\hat{\mathbf{A}}^{-1}$ and one application of the pressure Schur complement preconditioner $\hat{\mathbf{S}}^{-1}$. In case that $\hat{\mathbf{A}}^{-1}$ forms the most expensive part, this preconditioner increases costs by up to a factor of two compared to the other block preconditioners described above. In numerical experiments, a typical observation is that this preconditioner reduces the number of iterations only moderately or slightly compared to the block triangular preconditioner.

Remark 5.3 *Note that the above block preconditioners can be realized in a matrix-free way provided that a matrix-free representation can be found for $\hat{\mathbf{A}}^{-1}$ and $\hat{\mathbf{S}}^{-1}$. The following sections discuss such preconditioners.*

5.4.4 Preconditioners for the velocity block

According to the modular design of block preconditioners, the efficiency of this approach essentially depends on the availability of efficient preconditioners for the velocity block and the Schur complement block. This section discusses preconditioners $\hat{\mathbf{A}}^{-1}$ for the unsteady convection–diffusion operator $\mathbf{A}$ acting on the velocity unknowns.

- Inverse mass matrix preconditioner $\hat{\mathbf{A}}^{-1} = \mathbf{M}^{-1}$: This preconditioner can be derived by neglecting both the convective term and the viscous term in $\mathbf{A}$. As suggested in Shahbazi et al. (2007), it can be expected that this preconditioner is efficient if the mass matrix term is dominant, i.e., for small time step sizes Δt, and that the performance degrades if the

convective term or the viscous term become important, i.e., for large time steps sizes. This preconditioner is particularly efficient in the context of discontinuous Galerkin methods where the mass matrix is block diagonal. Consequently, applying the inverse mass matrix is a local, element-by-element operation. However, assembling and factorizing or inverting the diagonal blocks would still be an expensive operation especially for very large polynomial degrees, with an overall complexity of at least $\mathcal{O}(k^{2d})$. However, as first presented in Kronbichler et al. (2016), a matrix-free representation is available for the inverse mass matrix, which exploits the tensor-product structure and uses sum-factorization with optimal complexity of $\mathcal{O}(k^{d+1})$. This algorithm applies the inverse of the 1D interpolation from nodes to quadrature points and inverse quadrature weights. As explained in Krank et al. (2017), an alternative interpretation is the transformation from a Gauss–Lobatto basis to a Gauss basis, application of the inverse mass matrix, which is diagonal in the Gauss basis, and a subsequent transformation back to the Gauss–Lobatto basis. From a performance perspective, the inverse mass matrix is then as cheap as the forward application of the mass matrix, and can be characterized as a memory-bound operation for moderately large polynomial degrees on modern hardware. The speed of this operation is then determined by the speed at which the vectors can be streamed from memory, while arithmetic operations can be hidden behind the memory transfer.

- Jacobi and Gauss–Seidel preconditioning: In order to incorporate the convective and viscous terms, Jacobi-type preconditioners can be used. In the context of high-order discontinuous Galerkin discretizations it is typically found that point-Jacobi preconditioners are ineffective and that block-Jacobi techniques are required for robustness at high polynomial degrees, where block refers to the degrees of freedom of one element. As explained in the introduction of this chapter, such block-Jacobi approaches have so far most often been realized in a matrix-based fashion (see Section 5.1 for references to the literature). It is often argued that storing only the block-diagonal is acceptable as it is only a fraction of the whole matrix. However, these matrix-based approaches come along with significantly increased complexity for large k, $\mathcal{O}(k^{2d+1})$ for assembly, $\mathcal{O}(k^{3d})$ for factorizations, and $\mathcal{O}(k^{2d})$ for matrix-vector products. Recently, progress towards matrix-free block-type preconditioners applicable to general operators and non-Cartesian meshes has been made in Bastian et al. (2019), Pazner and Persson (2018). In Bastian et al. (2019), inner Krylov solvers are used to solve the local block-Jacobi problems. In the course of this thesis, vectorized versions of local CG and GMRES solvers have been implemented in order to exploit the favorite vectorization strategy of the matrix-free implementation in `deal.II` which vectorizes over several elements. The Krylov solver is run simultaneously for several elements and convergence is achieved once all elements of the vectorization batch report convergence. This preconditioner relies entirely on matrix-free algorithms with sum-factorization and each iteration has complexity $\mathcal{O}(k^{d+1})$. The overall efficiency of this preconditioner then essentially depends on how fast the local Krylov solver converges, where it was found in Bastian et al. (2019) that coarse tolerances can be used for the local problem without impacting convergence of the outer global solver. In this context, it should be taken into account that the data can reside in the cache for moderately large k due to the locality of the problem. For non-symmetric and convection-dominated problems, a Gauss–Seidel

iteration strategy is also popular for improved robustness and faster convergence.[1] ILU preconditioners might also be effective as long as the polynomial degree k does not become too large (matrix-based approach), see for example Kronbichler et al. (2018a).

- Multigrid: A more robust but also more expensive preconditioner is to use a multigrid cycle with appropriate smoother as a means to invert the operator. For Navier–Stokes problems with an explicit treatment of the convective term or for Stokes problems, the operator is symmetric and the favorite multigrid smoother used in this work is a fast Chebyshev iteration around the point-Jacobi method. For problems involving convection the operator becomes non-symmetric. In this case, one can either restrict the multigrid preconditioner to the (symmetric) Helmholtz-like part of the operator which still works well as long as the convective term does not become dominant, or apply the multigrid preconditioner to the whole operator with suitable smoothers for non-symmetric problems. In the latter case, the Jacobi and Gauss–Seidel preconditioners discussed above are often used as multigrid smoothers, requiring a relaxation in general for robustness and optimal convergence. Sometimes, a fixed number of GMRES iterations with Jacobi or Gauss–Seidel preconditioner is also used as multigrid smoother. The reader is also referred to Section 5.2 for a discussion of efficient multigrid algorithms in the context of high-order discontinuous Galerkin discretizations.

5.4.5 Preconditioners for the Schur complement block

In order to derive preconditioners for the Schur complement, a well-known approach is based on the theory of pseudo differential operators, see Benzi et al. (2005) and references therein. This means that the matrices contained in $\mathbf{S}$ are replaced by their respective spatial derivative operators. For this analysis, the convective term is neglected. Moreover, the Laplace formulation of the viscous term is used, while the preconditioner derived by this analysis is then also used when considering the divergence formulation of the viscous term. This derivation therefore includes the implicit assumption that the formulation of the viscous term plays a minor role in terms of constructing preconditioners that approximate the Schur complement in a spectrally equivalent way. Using pseudo differential operators, the Schur complement is written as

$$\mathbf{S} \mathrel{\widehat{=}} -\left(-\nabla\cdot\right)\left(\frac{\gamma_0^n}{\Delta t_n}\mathrm{Id} - \nu\Delta\right)^{-1}\nabla\,. \tag{5.28}$$

Then, efficient Schur complement preconditioners can be derived for two limiting cases:

- Limit of large time steps or large viscosities: Neglecting the term originating from the discrete time derivative and assuming commutativity of spatial derivative operators results in the following preconditioner

$$\mathbf{S} \mathrel{\widehat{=}} -\left(-\nabla\cdot\right)\left(-\nu\Delta\right)^{-1}\nabla \approx -\frac{1}{\nu}\mathrm{Id} \;\rightsquigarrow\; -\hat{\mathbf{S}}^{-1} = \nu\mathbf{M}_p^{-1}\,, \tag{5.29}$$

  where $\mathbf{M}_p$ denotes the pressure mass matrix. For the steady Stokes problem this is an optimal preconditioner for the Schur complement showing mesh-independent convergence.

[1] A Gauss–Seidel type iteration strategy is currently not supported by the matrix-free infrastructure available in `deal.II`.

- Limit of small time steps or small viscosities: Under these assumptions the viscous term can be neglected, leading to

$$\mathbf{S} \mathrel{\widehat{=}} -\left(-\nabla\cdot\right)\left(\frac{\gamma_0^n}{\Delta t_n}\mathrm{Id}\right)^{-1}\nabla \approx +\left(\frac{\gamma_0^n}{\Delta t_n}\right)^{-1}\Delta \;\rightsquigarrow\; -\hat{\mathbf{S}}^{-1} = \frac{\gamma_0^n}{\Delta t_n}\mathbf{L}^{-1}\,, \tag{5.30}$$

  where $\mathbf{L}$ is the matrix of the discrete negative Laplace operator using the same DG discretization (SIPG method) as for the discretization of the pressure Poisson equation in case of projection-type solution methods. To approximately invert the negative Laplace operator in a spectrally equivalent way, one multigrid V-cycle can be performed, leading to the approximation $\hat{\mathbf{L}}^{-1}$. An exact inversion of the negative Laplace operator, e.g., by using an iterative solver with multigrid preconditioner, should be avoided since such a nested application of iterative solvers becomes very expensive computationally, but typically shows only minor improvements in iteration counts.

By combining both approaches, a robust preconditioner has been developed in Cahouet and Chabard (1988) for the generalized Stokes problem

$$-\hat{\mathbf{S}}^{-1} = \nu\mathbf{M}_p^{-1} + \frac{\gamma_0^n}{\Delta t_n}\mathbf{L}^{-1}\,, \tag{5.31}$$

where an approximation $\hat{\mathbf{L}}^{-1}$ should again be used for computational efficiency. This preconditioner forms a spectrally equivalent approximation for the generalized Stokes problem and results in mesh-independent convergence rates for the whole range of time step sizes and viscosities. Hence, this preconditioner allows to obtain optimal computational complexity with costs proportional to the problem size. The additional application of the inverse pressure mass matrix comes at almost no extra costs compared to a multigrid V-cycle used for the inversion of the Laplace operator.

The convective term has been neglected in the above derivations. Extending these approaches to the Navier–Stokes equations (Elman and Silvester 1996, Kay et al. 2002) shows that these preconditioners allow to obtain mesh-independent convergence but the number of iterations increases roughly as ν^{-1} for decreasing viscosity for steady Navier–Stokes problems. To address small viscosities and to develop Schur complement preconditioners that incorporate the convective operator, the goal is to find a formulation that does not need to invert the convective operator but only involves a forward application of this operator. By assuming commutativity of spatial derivative operators, the inversion of the velocity convection–diffusion operator in the Schur complement preconditioner can be avoided (Elman et al. 2006)

$$\mathbf{A}\mathbf{B}^\mathsf{T} = \mathbf{B}^\mathsf{T}\mathbf{A}_p \rightsquigarrow \mathbf{A}^{-1}\mathbf{B}^\mathsf{T} = \mathbf{B}^\mathsf{T}\mathbf{A}_p^{-1} \rightsquigarrow -\mathbf{S}^{-1} = \left(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\mathsf{T}\right)^{-1} = \mathbf{A}_p\left(\mathbf{B}\mathbf{B}^\mathsf{T}\right)^{-1}\,. \tag{5.32}$$

Herein, the matrix $\mathbf{A}_p$ represents a pressure convection–diffusion operator in analogy to the velocity convection–diffusion operator $\mathbf{A}$. Accordingly, the inverse Schur complement can be applied to a vector without the need to invert the velocity convection–diffusion system $\mathbf{A}$. The operator $\mathbf{B}\mathbf{B}^\mathsf{T}$ is a discrete Laplace operator which can be inverted efficiently by using multigrid. This idea leads to the so-called pressure convection–diffusion preconditioner proposed and analyzed in Elman et al. (2008, 2002a,b), Kay et al. (2002), Silvester et al. (2001). The pressure

convection–diffuion preconditioner yields mesh-independent convergence rates and for decreasing viscosity the number of iterations increases approximately with $\nu^{-1/3}$ for steady Navier–Stokes problems. In Kay et al. (2002) this preconditioner is derived by considering the fundamental solution tensor for the Oseen operator. In Elman et al. (2002b), Silvester et al. (2001) this preconditioner is motivated by assuming commutativity of differential operators (including scalings by the inverse mass matrix)

$$\left(\mathbf{M}^{-1}\mathbf{A}\right)\left(\mathbf{M}^{-1}\mathbf{B}^{\mathsf{T}}\right) \approx \left(\mathbf{M}^{-1}\mathbf{B}^{\mathsf{T}}\right)\left(\mathbf{M}_p^{-1}\mathbf{A}_p\right) \ , \tag{5.33}$$

or equivalently

$$\mathbf{S} = -\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^{\mathsf{T}} \approx -\left(\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathsf{T}}\right)\mathbf{A}_p^{-1}\mathbf{M}_p \rightsquigarrow -\hat{\mathbf{S}}^{-1} = \mathbf{M}_p^{-1}\mathbf{A}_p\left(\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathsf{T}}\right)^{-1} \ . \tag{5.34}$$

The operator $\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathsf{T}}$ is known as compatible discretization of the Laplace operator compared to the classical discretization $\mathbf{L}$, see for example Cahouet and Chabard (1988). This operator is impractical since each application of the operator involves three operator evaluations and since a computation of the diagonal of this operator (required for example by multigrid smoothers) is expensive and difficult to realize in a matrix-free context. However, the compatible discretization of the Laplace operator can be replaced by the classical discretization $\mathbf{L}$

$$-\hat{\mathbf{S}}^{-1} = \mathbf{M}_p^{-1}\mathbf{A}_p\mathbf{L}^{-1} \ , \tag{5.35}$$

where a spectrally equivalent approximation $\hat{\mathbf{L}}^{-1}$ that is cheap to compute is used in practice. The operator $\mathbf{A}_p$ denotes the discretization of an unsteady convection–diffusion operator for the pressure

$$\frac{\gamma_0^n}{\Delta t_n}p_h + \nabla\cdot\left(\boldsymbol{u}_{\mathrm{lin}}p_h\right) - \nu\Delta p_h \ , \tag{5.36}$$

where the diffusivity is ν and the advection velocity is the current solution $\boldsymbol{u}_{\mathrm{lin}}$ of the Newton solver. DG discretizations of such a scalar transport equation have been derived in Chapter 3, where the convective term can be used in both the conservative formulation and convective formulation (the former being used as the default setup unless specified otherwise). On domain boundaries, Dirichlet boundary conditions are prescribed on Γ_h^{N} and Neumann boundary conditions on Γ_h^{D}. The inverse negative Laplace operator is the most expensive step of this preconditioner, since only a forward application of the pressure convection–diffusion operator is needed and due to the explicit character of the inverse mass matrix in DG.

Remark 5.4 *The need to implement a scalar convection–diffusion operator for a scalar quantity (in this case the pressure) is often seen as a disadvantage of this preconditioner since this operator is initially not needed for the discretization of the Navier–Stokes equations. However, the scalar convection–diffusion equation is often already implemented in computational fluid dynamics solvers, so that this preconditioner does not require additional implementations.*

Remark 5.5 *The pressure convection–diffusion operator can also be used for problems that do not involve convection. Inserting* $\mathbf{A}_p = \frac{\gamma_0^n}{\Delta t_n}\mathbf{M}_p + \nu\mathbf{L}$ *into equation* (5.35) *and assuming* $\hat{\mathbf{L}}^{-1} =$

L^{-1} *shows that the pressure convection–diffusion preconditioner is equivalent to the preconditioner* (5.31) *for the generalized Stokes problem*

$$-\hat{\mathsf{S}}^{-1} = \mathsf{M}_p^{-1}\mathsf{A}_p\mathsf{L}^{-1} = \mathsf{M}_p^{-1}\left(\frac{\gamma_0^n}{\Delta t_n}\mathsf{M}_p + \nu\mathsf{L}\right)\mathsf{L}^{-1} = \nu\mathsf{M}_p^{-1} + \frac{\gamma_0^n}{\Delta t_n}\mathsf{L}^{-1} \ . \tag{5.37}$$

Remark 5.6 *Another Schur-complement preconditioner that takes into account the convective term is the BFBt preconditioner developed in Elman et al. (2006), Elman (1999). However, the BFBt preconditioner is less robust than the pressure convection–diffusion preconditioner. The dependence of the number of iterations on the viscosity is approximately* $\nu^{-1/2}$ *for a steady Navier–Stokes problem, and this preconditioner does also not yield mesh-independent convergence rates (iterations increase roughly with* $h^{-1/2}$ *as mentioned in Kay et al. (2002)). For this reason, this preconditioner is not discussed in detail here.*

5.5 A unifying perspective

The above discussion of block-preconditioners for the coupled Navier–Stokes problem revealed that the construction of efficient preconditioners for the coupled system can be broken down into developing preconditioners for the velocity convection–diffusion system

$$\left(\frac{\gamma_0^n}{\Delta t_n}\mathsf{M} + \mathsf{C}_{\mathsf{lin}}\left(\mathbf{u}^{(k)}\right) + \mathsf{V}\right)\mathbf{u} = \mathbf{b}_u \ , \tag{5.38}$$

and the pressure Poisson problem

$$\mathsf{L}\mathbf{p} = \mathbf{b}_p \ . \tag{5.39}$$

The same types of problem have to be solved for the dual splitting scheme, equations (2.221) and (2.219), and for the pressure-correction scheme, equations (2.224) and (2.226). Hence, it is a general observation that various types of incompressible Navier–Stokes solution strategies require the same basic ingredients in terms of preconditioning in order to obtain a fast solver.

Remark 5.7 *Other solution approaches which are not discussed here, such as the algebraic splitting scheme used in Shahbazi et al. (2007) or the SIMPLE-based solution strategy used in Klein et al. (2015, 2013) in combination with DG discretizations, contain essentially the same ingredients. In this context, it is interesting to realize that algebraic approaches such as the Schur-complement preconditioners discussed above, the algebraic splitting scheme in Shahbazi et al. (2007), or the SIMPLE-based approach in Klein et al. (2015) make use of an analogy to operator splitting techniques and resort to the classical discretization* L *of the Laplace operator for ease of implementation or reasons of computational efficiency instead of the compatible version* $\mathsf{B}\mathsf{M}^{-1}\mathsf{B}^{\mathsf{T}}$ *that naturally arises from the algebraic system of equations. An interpretation of the SIMPLE algorithm in terms of block factorizations or block preconditioners for the coupled system is given in Elman et al. (2008, 2014).*

In the context of stabilized DG discretizations for the incompressible Navier–Stokes equations discussed in the present thesis, an additional velocity system needs to be solved consisting of the mass matrix and additional divergence and continuity penalty terms

$$\left(\mathsf{M} + \Delta t_n\mathsf{A}_{\mathsf{D}} + \Delta t_n\mathsf{A}_{\mathsf{C}}\right)\mathbf{u} = \mathbf{b}_u \ , \tag{5.40}$$

see equation (2.217) for the coupled solution approach, equation (2.222) for the dual splitting scheme, and equation (2.229) for the pressure-correction scheme. This system of equations is symmetric positive-definite. Similar to the momentum equation (5.38), an efficient preconditioner for this system of equations is the inverse mass matrix preconditioner. Results shown in Fehn et al. (2018a) indicate that the inverse mass matrix preconditioner can yield robust iteration counts for equations (5.38) and (5.40) for turbulent flow examples in case that the time step size is restricted according to the CFL condition. This is due to the scaling of the mass matrix by the inverse of the time step size. For increasing spatial resolution, the time step size has to be reduced according to equations (2.209) and (2.210), and the system of equations remains well-conditioned. Alternative preconditioners for equation (5.40) are a point-Jacobi or block-Jacobi preconditioner, or multigrid (with Chebyshev or Jacobi smoother) for faster and robust convergence, similar to the preconditioners discussed in Section 5.4.4.

Remark 5.8 *In general, the choice of optimal preconditioners depends on the problem as well as the physical and discretization parameters. Unless the problem size is very small, a multigrid preconditioner appears to be the most efficient option for the pressure Poisson problem with wide consensus throughout the literature. For the velocity convection–diffusion system, a multigrid preconditioner is likely the optimal choice for a steady Stokes problem, while the inverse mass matrix might result in the most efficient solution algorithm in case of a high-Reynolds-number problem with explicit convective term and the time step size restricted according to the CFL condition. However, providing a general answer to the question of optimal preconditioners is beyond the scope of this work. In particular, the question regarding the efficiency of explicit versus implicit formulations of the convective term (and, related to this, the use of matrix-free versus matrix-based preconditioners) appears to be open, see also Section 6.3. A point that can be made in a very general context is that the metric to be optimized should be computational costs and not iteration counts.*

5.6 Numerical results for Navier–Stokes solvers

The implementation of block preconditioners discussed in Section 5.4 has been validated in detail, and the main conclusions are briefly reported below. Since block-preconditioners are state-of-the-art techniques that are well-documented in the literature, no detailed results for all the different variants are shown here for reasons of brevity.

The block-triangular preconditioner typically results in a more efficient preconditioner compared to the block-diagonal version since the number of iterations is reduced significantly, but the costs to apply the preconditioner increase only marginally. Using a block-triangular factorization typically reduces the number of iterations only moderately compared to the block-triangular preconditioner, but a complete block-triangular factorization is significantly more expensive to apply since the velocity block has to be inverted twice. Therefore, the block-triangular preconditioner proves very efficient overall and often serves as a reasonable default choice. For the steady Stokes equations, multigrid for the velocity block and an inverse mass matrix preconditioner for the pressure block ensure robust convergence. For the unsteady Stokes equations, a Cahouet–Chabard preconditioner for the pressure block ensures mesh-independent convergence as well, in agreement with theory (Cahouet and Chabard 1988). For the incompressible Navier–

Table 5.9: Taylor–Green vortex problem: iteration counts for coupled solution approach considering a wide range of refinement levels and polynomial degrees (adaptive time stepping is used with a Courant number of $\mathrm{Cr} = 0.4$ for all simulations).

(a) coupled system (momentum equation and continuity equation)

l	Polynomial degree k					
	2	3	5	7	11	15
0	–	3.0	3.0	3.5	4.5	4.5
1	2.0	3.9	4.0	4.2	4.8	5.4
2	4.4	5.3	5.7	5.8	6.4	7.4
3	5.5	5.7	5.8	6.1	7.3	8.7
4	5.9	5.8	6.1	7.0	8.9	10.2
5	5.9	6.0	4.5	8.5	9.4	11.3
6	6.4	7.0	8.2	8.3	–	–
7	7.2	8.1	8.0	–	–	–
8	8.3	–	–	–	–	–

(b) penalty step (including divergence and continuity penalty terms)

l	Polynomial degree k					
	2	3	5	7	11	15
0	–	5.0	7.5	7.7	8.3	9.3
1	3.0	6.8	8.4	9.0	10.2	11.6
2	7.8	8.1	8.7	9.4	11.1	12.6
3	8.8	9.1	9.7	10.7	12.8	14.9
4	9.1	9.4	10.5	11.6	13.9	15.6
5	9.5	9.9	11.4	12.4	14.2	15.5
6	9.9	10.6	11.7	12.3	–	–
7	10.0	11.0	11.3	–	–	–
8	9.9	–	–	–	–	–

Stokes equations involving the convective term, it was found that the pressure convection–diffusion preconditioner ensures mesh-independent convergence by the example of the driven cavity problem considering the steady Navier–Stokes equations. There is currently no block-preconditioner available that also ensures robustness w.r.t. the viscosity. Note, however, that flows become turbulent (and therefore inherently unsteady in nature) for small viscosities. Then, the presence of a time derivative term in the equations improves conditioning and the above block-preconditioners still enable an efficient iterative solution of low-viscosity, high-Reynolds-number turbulent flows.

To demonstrate the robustness of preconditioners for the unsteady incompressible Navier–Stokes equations, results are reported for the three-dimensional Taylor–Green vortex problem as a representative of transitional and turbulent flows. Similar results as those shown here have already been published in Fehn et al. (2018a) for the dual splitting scheme. The mesh is a uniform Cartesian grid on a periodic box where the coarse grid consists of only one element. An explicit formulation of the convective term is used along with adaptive time stepping where the time step sizes is calculated according to the CFL condition, and solver tolerances of $\varepsilon_{\mathrm{abs}} = 10^{-12}$ and $\varepsilon_{\mathrm{rel}} = 10^{-3}$ are used here. The parameters for this flow problem can be found in Section 2.6.7.2. For the coupled solution approach, the inverse mass matrix preconditioner is used for the velocity block, and the Cahouet–Chabard preconditioner for the pressure block with a cph-MG V-cycle with Chebyshev(5) smoother and Chebyshev coarse-grid solver to approximately invert the Laplace operator in the Schur-complement preconditioner. The dual splitting and pressure-correction schemes use the same multigrid preconditioner for the pressure Poisson equation, and an inverse mass matrix preconditioner for the viscous step or momentum equation. For all solvers, the penalty step is preconditioned by the inverse mass matrix operator. The saddle-point problem is solved by the GMRES method, while symmetric positive definite

problems are solved by the CG algorithm. Results are summarized in Table 5.9 for the coupled solution approach and in Table 5.10 for the dual splitting scheme. For the pressure-correction scheme, iteration counts are almost identical to the dual splitting scheme and are therefore not shown explicitly here. This behavior is expected given that both projection methods are algorithmically very similar.

Table 5.10: Taylor–Green vortex problem: iteration counts for dual splitting scheme considering a wide range of refinement levels and polynomial degrees (adaptive time stepping is used with a Courant number of $\mathrm{Cr} = 0.4$ for all simulations).

(a) pressure step (Poisson equation)

l	Polynomial degree k					
	2	3	5	7	11	15
0	–	2.0	3.0	2.9	3.8	4.5
1	0.4	2.9	3.8	3.6	4.1	4.5
2	2.8	3.5	3.4	3.9	4.3	5.4
3	3.0	3.2	3.8	4.2	4.9	6.3
4	3.0	3.1	4.1	4.6	5.9	8.6
5	2.8	3.3	4.5	5.3	7.4	11.1
6	2.8	3.6	5.2	6.7	–	–
7	2.7	4.2	6.6	–	–	–
8	2.4	–	–	–	–	–

(b) viscous step (Helmholtz-like equation)

l	Polynomial degree k					
	2	3	5	7	11	15
0	–	2.0	2.0	2.0	2.0	2.2
1	2.0	2.0	2.0	2.0	2.4	3.0
2	2.0	2.2	2.5	2.8	3.4	4.4
3	2.4	2.7	3.0	3.2	4.7	5.5
4	2.9	3.0	3.4	4.4	5.7	6.9
5	3.2	3.5	4.6	5.6	6.6	8.2
6	4.0	4.6	5.8	6.3	–	–
7	5.1	6.0	6.4	–	–	–
8	6.6	–	–	–	–	–

(c) penalty step (including divergence and continuity penalty terms)

l	Polynomial degree k					
	2	3	5	7	11	15
0	–	5.0	7.5	7.7	8.3	9.4
1	3.0	6.8	8.4	8.8	9.8	11.5
2	7.8	8.3	8.9	9.3	11.1	12.4
3	9.0	9.2	9.7	10.6	12.6	14.2
4	9.3	9.6	10.6	11.6	13.2	14.3
5	9.7	10.2	11.4	12.0	13.4	14.9
6	10.1	10.6	11.6	12.0	–	–
7	10.4	11.1	11.2	–	–	–
8	10.1	–	–	–	–	–

A slight increase in iteration counts is observed for increasing spatial resolution, i.e, increasing mesh refinement level l and increasing polynomial degree k. Nevertheless, the number of iterations is very small, which is remarkable given that a simple inverse mass matrix preconditioner is used for the velocity system. Note that this is also related to the fact that the time step size decreases for increasing spatial resolution according to the CFL condition. Since the mass matrix term becomes more dominant with decreasing time step size (and decreasing viscosity),

Table 5.11: Taylor–Green vortex problem: throughput per time step for polynomial degrees $k = 2, 3, 5, 7, 11, 15$. The time interval is $0 \leq t \leq T = 20$. Adaptive time stepping with a Courant number of $\mathrm{Cr} = 0.4$ is used for all computations.

k	throughput per time step [MDoF/s/core]		
	coupled solver	dual splitting scheme	pressure-correction scheme
2	$0.43 \ldots 0.77$	$0.75 \ldots 1.26$	$0.74 \ldots 1.27$
3	$0.45 \ldots 0.61$	$0.69 \ldots 0.92$	$0.66 \ldots 0.90$
5	$0.40 \ldots 0.44$	$0.60 \ldots 0.70$	$0.61 \ldots 0.72$
7	$0.36 \ldots 0.36$	$0.53 \ldots 0.58$	$0.56 \ldots 0.61$
11	$0.20 \ldots 0.20$	$0.32 \ldots 0.36$	$0.34 \ldots 0.36$
15	$0.11 \ldots 0.12$	$0.18 \ldots 0.20$	$0.19 \ldots 0.22$

this strategy allows robust preconditioning for high-Reynolds-number turbulent flows. Due to a small number of iterations already for the simple inverse mass matrix preconditioner, multigrid preconditioners for the velocity block involve a computational overhead that often does not pay off in terms of overall computational costs despite a further decrease in iteration counts. For this reason, the setup chosen here proved highly efficient for the class of scale-resolving turbulent flow simulations that are of primary interest in the present work. For projection-type methods, it is typically found that the computational costs are well-balanced between the different substeps of the projection schemes, see also the results shown in Fehn et al. (2018a). In particular, the pressure Poisson equation does not form a bottleneck due to the use of efficient matrix-free multigrid methods. According to Section 5.3, this can be expected to also hold for complex geometries with larger coarse-grid problems due to the use of hybrid multigrid techniques. In terms of wall time, the relative share of computational costs for evaluating the convective term is typically only a few percent despite using exact integration for the nonlinear convective term with an increased number of quadrature points. Therefore, the convective term is negligible in terms of overall computational costs for incompressible Navier–Stokes solvers that treat this term explicitly. In contrast, exact integration used for the convective term can be expected to significantly deteriorate the overall performance for fully-implicit incompressible Navier–Stokes solvers, see Figure 4.10.

Table 5.11 lists the average throughput per time step in degrees of freedom solved per second of wall time and per core. The simulations have been performed on the hardware specified in Table 4.2. Only those simulations from Tables 5.9 and 5.10 are listed for which the problem size is large enough to saturate at least one compute node. The dual splitting scheme and the pressure-correction scheme achieve almost the same throughput. Compared to the coupled solution approach, the throughput is a factor of 1.5 to 1.8 larger for the projection-type methods, which results in a reduction in absolute run time by the same factor. This result is very interesting, given that such comparative studies are rarely available in the literature, especially not for solvers using the same implementation routines such that a fair comparison is possible. Of course, it should be noted that the observed speed-up of projection methods over the coupled solution approach is problem dependent. However, one might intuitively expect that more complex

problems, e.g. involving anisotropies or convective terms, result in a further performance advantage of projection methods, since the effect converging most slowly affects the convergence rate of the whole solver for the coupled solution approach, while it only affects a single sub-step in case of projection methods. In terms of absolute numbers, a throughput of up to one million degrees of freedom solved per time step per core per second of wall time is achieved. In Arndt et al. (2020b), a throughput of 1.05 MDoF/s/core per time step has been reported for the present DG solver at degree $k = 3$ for an inviscid TGV simulation with 10^{11} degrees of freedom run on 150k cores. The author is currently not aware of other high-order DG solvers for the incompressible Navier–Stokes equations achieving this level of performance. Towards high polynomial degrees, the throughput reduces continuously. On the one hand, this is due to a decrease in efficiency of the matrix-free operator evaluation for large polynomial degrees in three space dimensions according to Figure 4.10. On the other hand, this decrease in throughput is due to an increase in iteration counts for increasing polynomial degrees according to Tables 5.9 and 5.10. Note that this aspect is critical for the success of high-order discretizations applied in a setting where the solution is under-resolved such as high-Reynolds-number turbulent flows and where high-order discretizations do not show high-order accuracy. Chapter 6 investigates this aspect in detail.

5.7 Conclusion and outlook

This chapter has addressed the efficient iterative solution of algebraic systems of equations. The main novelity is the development of hybrid multigrid techniques for high-order DG discretizations, i.e., multigrid coarsening strategies that exploit all levels of geometric, polynomial, and algebraic coarsening with an additional transfer from discontinuous to continuous finite element spaces. This chapter has discussed the relevant design choices in the context of hybrid multigrid methods and has conducted performance comparisons for various multigrid methods and different types of p-coarsening in the metric of computational costs. Optimal-complexity matrix-free operator evaluation is exploited on all multigrid levels, smoothers, and transfer operators except for the coarse-grid solver. The performance is further improved by the use of mixed-precision multigrid. The results of these developments can be summarized as follows:

(i) A $p_{l-1} = \lfloor p_l/2 \rfloor$ coarsening strategy that reduces the number of unknowns roughly in factors of 2^d from one level to the next performs better than other p-coarsening types that reduce the polynomial degree by one until the lowest degree is reached, or directly from high-order to the lowest polynomial degree within one level.

(ii) Performing the c-transfer from discontinuous to continuous space at the fine level is superior to an alternative c-transfer performed at the coarse level before the coarse-grid solver is invoked. Moreover, this approach yields a multigrid algorithm with iteration counts that are independent of the penalty factor of the interior penalty method, which is an important result of the present thesis. The cph- and chp-multigrid methods are identified as most promising (and could also be interesting in a full multigrid context).

(iii) By the development of hybrid multigrid methods that exploit all possibilities of h-, p-, and c-coarsening, the bottleneck of expensive coarse-grid solvers, which would otherwise dominate overall computational costs, is significantly relaxed. An extension of the hybrid

multigrid methods proposed here towards hp-adaptivity (preferably h-adaptivity) is also expected to improve overall efficiency of the solver by making optimal use of degrees of freedom.

This chapter has also discussed preconditioning strategies for different incompressible Navier–Stokes solvers, including block preconditioners for saddle-point problems that are required in case of a coupled solution approach. The main ingredients for all types of Navier–Stokes solvers are preconditioners for a reaction–(convection–)diffusion problem for the velocity unknowns, and a preconditioner for the pressure Poisson operator. Hence, the incompressible Navier–Stokes solvers can make use of the same hybrid multigrid techniques developed in this chapter. The fact that the inverse mass matrix can be applied very efficiently in a matrix-free way renders this preconditioner highly efficient in the context of high-Reynolds-number turbulent flow problems. The effectiveness of these preconditioning techniques has been demonstrated by the example of the three-dimensional Taylor–Green vortex problem. A direct comparison of coupled and splitting-type Navier–Stokes solvers gives insights into their relative performance and the increase in costs due to the solution of a saddle-point problem compared to projection solvers. Such numbers are often difficult to extract from the literature since the use of different implementation frameworks typically excludes a one-to-one comparison.

For the solution of fully-implicit problems involving non-symmetric convective terms, matrix-based block-Jacobi or block-Gauss–Seidel techniques are currently the state-of-the-art. As briefly described in this chapter, these techniques can be used as preconditioners or as smoothers in a multigrid context. A partially matrix-based implementation of these preconditioners as well as a matrix-free implementation with an iterative solution of the local block-Jacobi problems as first presented in Bastian et al. (2019) has been realized in the course of this thesis, but using a vectorized implementation that solves the local problems simultaneously for several elements. Parts of this implementation are currently restricted to the serial case. Furthermore, only a Jacobi-like iteration over the elements of the mesh is currently possible, while Gauss–Seidel-type iteration strategies are currently not supported by the matrix-free infrastructure (and would require a special treatment such as multiple colors in parallel). The partially matrix-based variant can be expected to be efficient in two space dimensions, but to become prohibitively expensive in three space dimensions for high polynomial degrees. For the fully matrix-free variant, the development of efficient matrix-free preconditioners for the elementwise block-Jacobi problems appears to be crucial, where the inverse mass matrix is currently mainly used as preconditioner. Here, techniques based on the fast diagonalization method could prove efficient, even though this field is scarcely explored so far in the field of DG methods and for general differential operators, see also Pazner and Persson (2018). According to the author's opinion, fully matrix-free block-Jacobi preconditioners are a key ingredient to render fully implicit solvers computationally efficient for high-order methods in three space dimensions. One could also imagine matrix-based block-Jacobi preconditioners with some sparsification, see for example the tri-diagonal preconditioner used for the iterative solution of local block-Jacobi problems in Bastian et al. (2019). Hence, the highest potential for further performance improvements is expected to lie in the development of fast multigrid smoothers that are robust for anisotropic problems (stretched meshes such as boundary layers in turbulent flows or variable coefficients) and problems involving convection, and that can be realized in an entirely matrix-free way.

6 Efficiency of incompressible flow solvers

This chapter addresses the practically relevant aspect of simulating application problems efficiently. Generally speaking, the goal is to achieve a certain level of accuracy with a minimal amount of computational costs, e.g., in a minimum of time. Likewise, a numerical method can be considered more efficient if it produces more accurate results for a given amount of computational costs. This leads to the following generic definition of efficiency (Wang et al. 2013)

$$\text{efficiency} = \frac{\text{accuracy}}{\text{computational costs}} \,. \tag{6.1}$$

The metric accuracy is typically a problem-specific quantity of interest, such as the lift or drag coefficient, the reattachment length, or the kinetic energy dissipation rate in the context of fluid dynamical problems considered here. A definition of computational costs is more involved. Consider the example that a simulation run on a workstation produces the same level of accuracy within the same amount of time as another numerical method simulated on a compute cluster or even supercomputer. Clearly, the former method is more efficient since it requires significantly less resources. From this perspective, it is obvious that computational costs must also include the amount of utilized resources in order to obtain an objective cost metric in a general setting. Various definitions are conceivable and will be discussed in more detail in this chapter.

Investigations of type error-vs-costs are highly interdisciplinary and include various aspects, from properties related to the spatial discretization scheme, the time integration strategy, the solution of nonlinear and linear systems of equations, preconditioners, the type of algorithms used to implement these methods in relation to the hardware under consideration, the parallelization strategy, to aspects of parallel scalability. Each of these aspects allows different design choices with different characteristics in terms of robustness, generality, and efficiency. What renders this topic challenging is the fact that a numerical approach optimal w.r.t. the metric (6.1) can not be found by optimizing each category and its parameters separately with black-box interfaces to other aspects. Instead, these aspects are strongly inter-connected with changes made regarding one aspect affecting others. This chapter presents a methodology to analyze these aspects systematically. Due to the complexities mentioned above, this work does not reclaim completeness, but instead proposes one particular approach, however, with the goal to motivate this approach and its design choices with a holistic view on various aspects. Parts of this chapter are based on work that has already been published in Fehn et al. (2018a). Results of a parallel scalability study shown in this chapter have already been published in Arndt et al. (2020b).

This chapter assembles many ingredients from previous chapters. The accuracy and robustness of high-order DG discretizations (Chapter 2), optimal-complexity linear solvers and preconditioners (Chapter 5), and the fast matrix-free evaluation of discretized operators (Chapter 4) form the main building blocks determining overall efficiency, as discussed at length in this chapter.

Section 6.1 gives an introduction with a definition of important metrics, Section 6.2 discusses efficiency models, and Section 6.3 presents numerical results in terms of the error-vs-cost metric for different benchmark problems in computational fluid dynamics. A conclusion and outlook is given in Section 6.4.

6.1 Introduction

Due to the interdisciplinary nature of the topics discussed here (and CFD in general as emphasized in (Löhner 2008, Chapter 1)), a critical assessment of the state-of-the-art appears to be imperative. Although computational science and engineering is an established discipline, computational efficiency is a topic that is often treated shabbily in the literature on high-order discretization methods. Benchmarking can be understood as putting errors and computational costs of a numerical approach into relation. The present work wants to foster a view that considers benchmarking and computational efficiency a necessary ingredient in order to make progress, and therefore puts a main emphasis onto these topics. In this context, great efforts have been made in the comparative study by Wang et al. (2013) in setting standards of how to evaluate the efficiency of high-order methods in an objective manner. This section highlights the need for computationally efficient flow solvers by recalling well-known estimates describing the computational complexity as a function of the Reynolds number. Then, the state-of-the-art is discussed in terms of the efficiency of high-order discretizations, the relevance of parallel scalability, and suitable metrics to measure computational costs.

6.1.1 Influence of Reynolds number on computational complexity

As a background for subsequent discussions, this section briefly summarizes and discusses the usual estimates of how computational complexity grows with the Reynolds number for direct numerical simulations of turbulent flows, see for example Ferziger and Peric (2008), Pope (2001). A direct numerical simulation of a turbulent flow resolves all scales down to the Kolmogorov scale $\eta = (\nu^3/\varepsilon)^{1/4}$, where the dissipation rate is estimated as $\varepsilon \sim U^3/L$ with a characteristic velocity U describing the large scales of the flow and an integral length scale L. Considering a numerical method of fixed polynomial degree k of the shape functions, the number of elements required to resolve the flow increases with the Reynolds number as follows

$$N_{\mathrm{el}} \sim \left(\frac{L}{h}\right)^3 \sim \left(\frac{L}{\eta}\right)^3 \sim \left(\frac{U\,L}{\nu}\right)^{(3/4)\cdot 3} = \mathrm{Re}_L^{9/4} \,. \tag{6.2}$$

Assuming that a fixed number of eddy turnover times or flow-through times L/U need to be simulated and that the time step size is restricted according to the CFL condition, $\Delta t \sim h/U$, the total number of time steps increases with the Reynolds number as follows

$$N_{\Delta t} \sim \frac{L/U}{\Delta t} \sim \frac{L}{h} \sim \frac{L}{\eta} \sim \mathrm{Re}_L^{3/4} \,. \tag{6.3}$$

From these estimates it is clear that computational complexity or costs increase with the Reynolds number, independently of how a definition of computational costs looks like exactly. Put differently, on a given hardware, the maximum Reynolds number for which a scale-resolving simulation of a turbulent flow can be computed is limited for different reasons. It is the content of

subsequent sections to discuss these limits. Note that estimates leading to other exponents for the Reynolds number, e.g. for boundary layer flows, are available as well, see for example Löhner (2008, Section 4.6) and Jiménez (2003).

6.1.2 Efficiency of high-order discretizations

A fundamental question accompanying the development of high-order discretization schemes is whether the use of high-order methods pays off in terms of overall computational efficiency. In other words, the question is which combination of the discretization parameters h and k yields the best efficiency. The term "efficiency" is used here in the sense of equation (6.1) with a vague definition of computational costs, given that the works cited in this section use different cost metrics. Then, this question can be addressed by theoretical/analytical investigations and by numerical/experimental investigations:

- theoretical approach: Both computational costs and the discretization error are estimated by analytical models as a function of the parameters h, k, d. Inserting both relations into each other yields the error-vs-cost relation with the polynomial degree k as the main parameter, providing insight which polynomial degree is most efficient depending on the other parameters.

- numerical approach: A specific PDE model problem is considered and solved numerically for a specific geometry and boundary conditions for a set of parameters h, k, d. Measuring the computational costs and the discretization error (requires analytical solution or other accurate reference data) allows to identify optimal discretization parameters, that might then be generalized to other problems, e.g., flow configurations with similar characteristics in terms of the Reynolds number.

Examples of the theoretical approach can be found in Huerta et al. (2013), Löhner (2011, 2013). These works assume optimal convergence behavior according to $\varepsilon \sim h^{k+1}$ and naive matrix-based implementations of high-order continuous or discontinuous Galerkin discretizations, where the number of non-zeros of the matrix, $\mathcal{O}((k+1)^{2d})$, or the number of floating point operations, $\mathcal{O}((k+1)^{2d})$ for the matrix-vector product and $\mathcal{O}((k+1)^{3d})$ or $\mathcal{O}((k+1)^{2d+1})$ for assembly, are used as cost metric. The works by Löhner (2011, 2013) conclude that linear elements are most efficient for a level of accuracy that is of engineering interest (error of approximately 1%) and claim an efficiency advantage of several orders of magnitude for finite difference schemes (with matrix-free implementation) over the quadrature-based finite element discretizations with matrix-based implementation. The work by Huerta et al. (2013) concludes that methods exploiting static condensation (including the HDG approach) are superior compared to methods without static condensation, and that high-order methods may be more efficient than linear elements if this technique is exploited. In both cases, conclusions drawn in these works should be taken with some care due to the assumption of matrix-based implementations, for example: (i) another study by Kronbichler and Allalen (2018) has demonstrated that efficient implementations of high-order discontinuous Galerkin discretizations are indeed able to approach finite-difference performance where the performance penalty is a small integer factor rather than orders of magnitude as suspected in Löhner (2013); (ii) another study by Kronbichler and Wall (2018) has shown that high-order (dis-)continuous Galerkin discretizations

Table 6.1: Selected publications from the literature numerically addressing the question of optimal discretization parameters for high-order discretization methods, with a characterization in terms of smooth versus non-smooth solutions and matrix-based versus matrix-free implementations.

	smooth solution, asymptotic regime	**non-smooth solution, pre-asymptotic regime**
matrix-based implementation	Fidkowski and Darmofal (2004) Nastase and Mavriplis (2006a,b) Vos et al. (2010) Chang et al. (2018)	Vos et al. (2010)
matrix-free implementation	Fischer et al. (1988) Kronbichler and Wall (2018) Vermeire et al. (2017) Rojas et al. (2021) Krank et al. (2017) Fehn et al. (2018a)	Kronbichler and Wall (2018) Vermeire et al. (2017) Fehn et al. (2018a, 2019c)

with matrix-free implementation of optimal complexity (using the sum-factorization technique as discussed in Chapter 4) outperform matrix-based approaches based on the static condensation idea (including HDG) significantly in three space dimensions on modern computer hardware. Other works by Fehn et al. (2018a), Fidkowski and Darmofal (2004) also briefly discuss the efficiency of high-order discretizations from a theoretical perspective in terms of error and cost estimates, mainly with a focus on the asymptotic behavior of high-order methods.

The above discussion reveals that inappropriate assumptions do not allow to identify optimal discretizations. The theoretical approach has severe limitations, most importantly the definition of computational costs, assumptions about the implementation strategy, and the assumption of an optimal convergence behavior of $\varepsilon \sim h^{k+1}$. The latter behavior can not be observed for most practical problems, calling for a numerical investigation of this question. Table 6.1 lists contributions from the literature evaluating the efficiency of high-order discretization methods in a (rigorous) error-versus-cost metric. Four categories are distinguished, depending on whether a matrix-based or matrix-free implementation is used and whether smooth or non-smooth problems are considered. Instead of considering smoothness of the solution in an absolute sense, characterizing the resolution of the discretization scheme as operating in the regime of asymptotic or pre-asymtotic convergence leads to a similar classification. By definition, LES and DNS of turbulent flows can be categorized as operating in the pre-asymptotic regime. The comparative study by Wang et al. (2013) covers different panels, but the missing categorization in terms of implementation strategies and algorithms renders an interpretation of the results difficult.

For "smooth" solutions, the picture appears to be rather clear in the literature and high-order methods have a clear advantage in particular if a solution of high-accuracy is required, see also the conclusions drawn in Wang et al. (2013). This is even the case if matrix-based implementations of sub-optimal complexity w.r.t. the polynomial degree k are considered. This holds for

simple Poisson, Helmholtz, or convection–diffusion model problems (Chang et al. 2018, Fischer et al. 1988, Kronbichler and Wall 2018, Vos et al. 2010), but also for Euler and Navier–Stokes problems with smooth, laminar solutions (Fehn et al. 2018a, Fidkowski and Darmofal 2004, Krank et al. 2017, Nastase and Mavriplis 2006a,b, Rojas et al. 2021, Vermeire et al. 2017). This behavior is also confirmed by theoretical estimates (Huerta et al. 2013, Löhner 2011, 2013) and can be explained as follows. The exponential convergence $\varepsilon \sim h^{k+1}$ causes the error to decrease faster for increasing k than the computational costs increase with k, since the costs increase only algebraically with k even if the implementation has sub-optimal complexity like $\mathcal{O}((k+1)^{2d})$ or worse.

If the solution is non-smooth (or the discretization operates in the pre-asymptotic regime) *and* a matrix-based implementation is used, it appears to be difficult to achieve a performance advantage by the use of high-order discretizations, since high-order methods are only as accurate as or slightly more accurate than low-order discretizations[1] in this regime, but significantly more expensive per degree of freedom due to a matrix-based implementation. The rareness of literature demonstrating the opposite might indicate an intrinsic difficulty. As argued in Brown (2010), for high-order methods to be competitive in the pre-asymptotic regime, they must have comparable cost per DoF as low-order methods, which can not be realized by matrix-based implementations. The example with singularity considered in Vos et al. (2010) appears to be an exception and favors a high polynomial degree of $k \approx 5$ for a matrix-based implementation (using a direct solver), but is a two-dimensional problem for which the increased complexity of matrix-based methods is typically less severe. The results shown in Kronbichler and Wall (2018) for a problem with singularity in three space dimensions confirm that the use of optimal-complexity matrix-free implementations is essential in order to render high-order discretizations more efficient, where optimal efficiency is achieved around $k = 5$.

While many works consider simple Poisson or Helmholtz-like model problems with smooth analytical solution, the present work is particularly interested in application problems in computational fluid dynamics and the simulation of turbulent flow problems. In this context, the most interesting regime is therefore the lower right panel in Table 6.1. On the one hand, this is due to the fact that problems operating in the pre-asymptotic regime such as under-resolved turbulent flows or flows involving singularities and shocks have a high practical relevance in computational fluid dynamics. The accuracy of high-order DG discretizations per degree of freedom is for example investigated in Beck et al. (2014), Gassner and Beck (2013) for the compressible Navier–Stokes equations, and in Fehn et al. (2017, 2018b, 2019a) for the incompressible Navier–Stokes equations, where a focus is put on under-resolved turbulent flows (see Section 2.4 for further references to the literature). These results motivate to investigate the efficiency of high-order methods in more detail in terms of error-vs-costs, see for example Vermeire et al. (2017) and Fehn et al. (2019c) for high-order flux-reconstruction or DG discretizations of the compressible Navier–Stokes equations, and Fehn et al. (2018a) for DG discretizations of the incompressible Navier–Stokes equations. On the other hand, the lower right panel in Table 6.1 is so important because fast matrix-free implementations of high-order methods can be considered a necessary ingredient to achieve an efficiency advantage for high-order discretizations. By the example of the three-dimensional viscous Taylor–Green vortex problem at $\mathrm{Re} = 1600$, the

[1] A common argument is that high-order discretizations have smaller error constants and better dispersive properties that might render the scheme more accurate in the absence of optimal rates of convergence.

works by Fehn et al. (2018a, 2019c) highlight the importance of using iterative solvers, preconditioners, and matrix-free implementation techniques of optimal computational complexity[2] in order to render high-order methods more efficient for such turbulent flow problems. In the field of high-order DG discretizations of the *incompressible* Navier–Stokes equations (which is the primary target of the present thesis), it has been difficult to find works of other authors rigorously quantifying efficiency in an error-vs-costs metric. The comparative study by Wang et al. (2013) mainly considers *compressible* Navier–Stokes solvers based on high-order DG methods. A shortcoming of the study by Wang et al. (2013) is that it did not identify the aspect of matrix-based versus matrix-free implementations as a main performance-relevant factor for high-order methods (only the high memory requirements of implicit time stepping schemes are mentioned vaguely). This is exactly where the present thesis contributes to the state-of-the-art.

Remark 6.1 *Section 6.1.1 only considers h as discretization parameter instead of the pair h, k. The underlying assumption is that LES and DNS are operating in the pre-asymptotic regime. High-order methods can not be expected to converge with optimal rates of convergence in such a setting. Instead, the use of high-order methods of degree k is advantageous in the sense of reducing the number of unknowns required to reach a certain level of accuracy by a constant factor, i.e., how many points per wavelength does a disretization scheme of degree k require to obtain an acceptable accuracy. This aspect is discussed in more detail in Section 6.2.4.2. Then, the relation $h \sim \eta$ still holds but with a proportionality constant depending on the polynomial degree k (and a similar proportionality constant accounting for the polynomial degree in the CFL condition), so that the estimates from Section 6.1.1 appear to be appropriate also for the high-order case.*

6.1.3 Parallel scalability

Parallel scalability describes the property that a problem of fixed size can be solved faster by using more processors (strong scaling), or that larger problems can be solved within the same amount of time when increasing the number of processors proportionally to the problem size (weak scaling). Assuming optimal scalability, a program that runs over a wall-clock time of $t_{\text{wall},1}$ on $P = 1$ processor will need a wall-clock time of $t_{\text{wall},P} = t_{\text{wall},1}/P$ when run on P processors (strictly speaking, processor is here a node or multiple nodes in case of classical CPU compute clusters and not a single core due to the heterogeneity of modern multicore CPUs with respect to memory access). However, this parallel wall-time is typically not reached due to communication overhead or non parallelizable work. Hence, it makes sense to define the parallel speedup S and parallel efficiency η

$$S = \frac{t_{\text{wall},1}}{t_{\text{wall},P}} \quad , \quad \eta = \frac{t_{\text{wall},1}}{t_{\text{wall},P} \cdot P} \, . \tag{6.4}$$

The parallel efficiency is typically $\eta \leq 1$, but might also be larger than one due to cache effects, see Section 4.5, e.g., the program runs in a saturated regime for $P = 1$ and reaches the regime where data fits into caches for large P. The reader is referred to (Hager and Wellein 2010, Chapter 5) for more detailed models on parallel scalability.

[2] Optimal computational complexity is used here in the sense of using algorithms and implementations minimizing time-to-solution.

Parallel scalability and in particular parallel speedup (strong scaling) should not be misunderstood as a stand-alone metric, as has been the case in the early times of the Gordon Bell prize where the first year's prize was awarded to the group demonstrating the highest speedup (Bell et al. 2017), and still is the case nowadays to some extent, see for example the ACM Gordon Bell prize winner paper by Rudi et al. (2015), from which it is difficult to extract the code's node-level performance. Instead, parallel scalability should be inherently linked to the time-to-solution metric as the metric that reports on the true speed of an implementation, see for example the works by Fischer (2015), Offermans et al. (2016). As emphasized in Hager and Wellein (2010), parallel scalability should be addressed once the single-core or node-level performance has been sufficiently optimized (time-to-solution). Despite efforts made to establish standards in the field of scientific computing (Bailey 2009), it is still common practice to only show normalized speedup factors against a baseline simulation run on a smaller number of cores, see for example the recent works by Altmann et al. (2013), Cantwell et al. (2015), Hindenlang et al. (2012), Loppi et al. (2018), Witherden et al. (2014), Yakovlev et al. (2016) in the field of high-order continuous and discontinuous Galerkin discretization methods, the works by Heinecke et al. (2014), Rudi et al. (2015) with a focus on HPC, and other contributions by Houba et al. (2019). Normalizing the computation time against a baseline is problematic since this creates wrong incentives in the sense that using a slower code (a code with worse serial performance) yields "better" speed-up factors. For this reason, practitioners remind the CFD community of what the actual interest is, namely completing the simulation of unsteady turbulent flow problems with millions of time steps in a reasonable amount of time (Löhner 2019, Löhner et al. 2020). The work by Löhner (2019) describes the circumstance that LES of industrial problems often requires wall-times of weeks rather than hours no matter how many processors are used as "LES crisis" (and the fact that this circumstance has not improved dramatically over the last two decades as compared to other progress made in computer hardware). This circumstance is attributed to the large number of time steps (typically millions of time steps) required to gather turbulent statistics and it is argued that the strong-scaling limit (the minimal wall-time per time step for an arbitrarly large number of processors) is the relevant metric to overcome the LES crisis. The term "LES crisis" considers equation (6.3) as the limiting factor, thereby restricting the Reynolds number for which turbulent flow simulations are affordable. A practical example is given in the work by Löhner et al. (2020) which targets overnight LES of external car aerodynamics. The work by Müller et al. (2019) addressing PDE solvers for numerical weather prediction also reflects this attitude in the sense of how many days of forecast are possible within one wall-clock day.

Following this time-to-solution credo, parallel scalability in the field of high-order discretization methods has been investigated on the level of matrix-free operator evaluation in Kronbichler and Kormann (2012), Müthing et al. (2017), on the application level of geometric multigrid solvers with matrix-free smoothers on uniformly refined meshes in Arndt et al. (2020b), Gholami et al. (2016), Kronbichler and Wall (2018), and on adaptively refined meshes in Clevenger et al. (2020). Parallel scalability for incompressible Navier–Stokes solvers is shown in Arndt et al. (2020b), Krank et al. (2017), Offermans et al. (2016, 2019). A domain-decomposition-based evaluation of discretized PDE operators involving communication with nearest neighbors can be considered to be a scalable algorithm, where the present work essentially builds upon parallelization and implementation concepts developed in Bangerth et al. (2012). The reason for this scalability is that in a weak scaling setup the computation-to-communication ratio (volume-to-surface ratio) stays constant. The minimum wall-time in the strong-scaling limit is typically

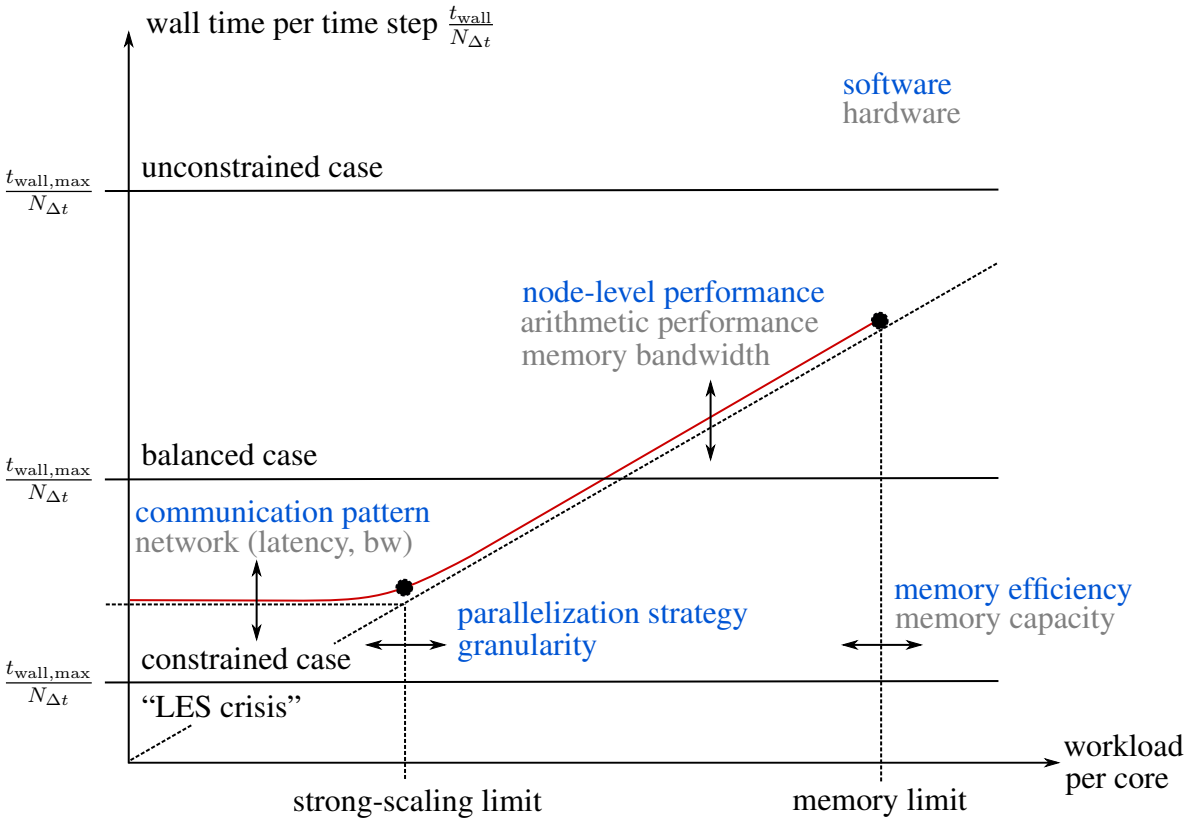


Figure 6.1: Illustration of various limits for PDE solvers in terms of workload per core and wall-time per time step. The impact of main hardware and software quantities is also highlighted. The red scaling curve corresponds to the unconstrained case.

dominated by internode latency for this type of operation (Fischer 2015), for moderately large polynomial degrees. Strong scalability of such PDE solvers is not described well by Amdahl's law with a certain (constant) proportion of non-parallelizable work, since the non-parallelizable part of such a PDE solver depends on the problem size. The share of non-parallelizable work decreases for increasing problem size and allows scalability to an increasing number of cores if the problem becomes larger. While explicit time stepping solvers only require this form of nearest-neighbor communication, scalability for implicit solvers is potentially limited by global communications (all-reduce) of complexity $\mathcal{O}(\log P)$ involved in Krylov solvers and multigrid preconditioners. However, theoretical models and performance extrapolations to exa-scale machines (Fischer 2015, Ibeid et al. 2020) are optimistic in the sense that this effect is not dominating for the largest supercomputers currently available and also for next-generation's exa-scale supercomputers.

There are also works that put a main emphasis onto the other end of strong scalability, namely the maximum problem size that can be solved on a given supercomputer, see for example the work by Bauer et al. (2020) dealing with problems of earth mantle convection. This metric favors algorithms optimized for memory efficiency. From such a perspective, the maximum Reynolds

number would be limited by equation (6.2). The question whether this memory limit is practically relevant depends on whether the large number of time steps required by the physical problem and the application at hand, see equation (6.3), can be simulated for the largest possible problem sizes in such a setting, since the high workload per core will imply a very large wall-time already for a single time step. These different points of view are summarized visually in Figure 6.1. In terms of the workload per core, one can distinguish between the strong-scaling limit and the memory limit (maximum problem size). The relevant limit for a PDE solver is then determined by the maximum wall-time (e.g. the wall-time limit on a cluster or supercomputer) divided by the number of time steps $N_{\Delta t}$ depending on the application (PDE model, resolution, Reynolds number, etc.). In this context, three cases can be distinguished: The first one is the *unconstrained case* implicitly assumed in the work by Bauer et al. (2020), where the underlying assumption is that one time step can be solved fast enough to not pose a constraint in terms of overall runtime of the simulation. The limiting resource is then the maximum problem size fitting into memory. The other limit is the *constrained case*, sententiously described as "LES crisis" in Löhner (2019). The underlying assumption of this limit is that the number of time steps required by the application is too large to complete the simulation with a desired spatial resolution within the wall-time limit. Apart from optimizing the code, the only solution to this constraint is to solve a smaller problem (with fewer time steps). As problem size and number of time steps increase monotonously with the Reynolds number for scale-resolving simulations according to equations (6.2) and (6.3), this puts an upper bound on the maximum Reynolds number that can be simulated. In between these two cases it the *balanced case*, i.e., the workload per core (the number of processors for a given problem size) is chosen such that the desired wall-time limit is reached, without constraints in terms of strong scalability or the memory capacity.

Remark 6.2 *Note that the assessment of whether the memory limit is practically relevant (unconstrained case in Figure 6.1) essentially depends on the ratio of memory size to memory bandwidth (or peak performance) of current supercomputers, which is difficult to answer once and for all. However, the work by Bauer et al. (2020), which is pioneering in exploring the largest possible problem sizes, appears to be primarily oriented to stationary problems given that the solution of one system of equations requires a wall-clock time of more than 700 s for the largest problem size of* $1.1 \cdot 10^{13}$ *unknowns, which in turn would allow to simulate only around 100 time steps in wall-clock times of a day. In contrast, unsteady simulations of a simplified earth mantle convection model run in Gmeiner et al. (2015a) already require* $\mathcal{O}(10^4)$ *time steps for a much smaller problem size of less than* 10^{10} *unknowns. This indicates that the problem size fitting into memory is not the limiting resource for time-dependent problems, since the memory bandwidth (or peak performance) of the hardware is too low to complete such unsteady simulations in reasonable wall-times. As a consequence, the problem size appears to be constrained more severely on this hardware by the required number of time steps large enough for the simulation to make sufficient progress in physical time (within a given wall-time limit).*

Remark 6.3 *Understanding high-performance computing as the iterative process of removing the most pressing bottlenecks, it is clear that one must first understand the limits of a PDE solver and its intended range of applications prior to optimizing the algorithm and implementation. The use of matrix-free methods as opposed to matrix-based methods is an important measure to relax the memory limit and to allow large problem sizes (Bauer et al. 2020). The present work demonstrates the range of problem sizes in form of a strong-scaling study, but does not explicitly address*

the memory limit in terms of improved, memory-lean algorithms, for reasons indicating a limited practical relevance on current supercomputers for the type of problems (scale-resolving simulations of unsteady flow problems) and the type of matrix-free implementation addressed here, see also Remark 6.2. An observation typically made for application runs is that the code operates either in the balanced regime or the strong-scaling regime. The quantities to look at during code optimization are then node-level performance and strong scalability, see for example Fischer et al. (2020b). To give an example, the largest problem solved in this work (Chapter 7) with more than 10^{11} unknowns and 10^5 time steps for the inviscid Taylor–Green problem (which shares the characteristics of homogeneous isotropic turbulence) operates in the balanced regime, i.e., the number of processors has been selected to complete the simulation in an acceptable wall-time. On the one hand, larger problem sizes exhausting the supercomputer's memory would have exceeded wall-time limits of several weeks, and on the other hand, the number of cores available was too small to reach the strong-scaling limit for the simulated maximum problem size. For very high polynomial degrees such as $k = 15$, the minimum wall-time per time step is typically found to be a pressing issue for the considered matrix-free implementation. The transport velocity in other canonical turbulent flows (free stream jets, channel flows, boundary layer flows) in streamwise direction causes significantly smaller time step sizes according to the CFL condition (and more time steps are required in case of long time intervals for statistical averaging), such that the point of operation can be expected to be shifted towards the strong-scaling limit compared to the case of homogeneous isotropic turbulence. For these reasons, the present work focuses on aspects of node-level performance and strong scalability.

Remark 6.4 *The new ARM-based A64FX chip by Fujitsu (32 GB on-chip memory with a bandwidth of 1 TB/s) exhibits a significantly higher memory bandwidth relative to the memory capacity than the Intel Skylake architecture listed in Table 4.2. Future hardware might be characterized by on-chip memory providing increased bandwidth (high bandwidth memory) but reduced capacity. For future HPC systems developing in this direction, it can be expected that the memory limit in Figure 6.1 moves closer to the strong-scaling limit.*

In order to highlight that parallel scalability should be inherently linked to time-to-solution, this chapter addresses the topic of parallel scalability (or the problem termed "LES crisis") in the form of the following question: What is the largest problem size for which the three-dimensional viscous Taylor–Green benchmark problem can be solved in real-time? Likewise, what is the minimum wall-time in which grid-converged results can be obtained for this Taylor–Green vortex problem?

6.1.4 Definitions of computational costs

Various definitions of "computational costs" are conceivable. For CPU hardware, the metric computational costs is typically defined as the product of the wall-clock time times the number of cores utilized, i.e.,

$$\text{computational costs } c = \text{wall-time } t_{\text{wall}} \times \text{cores } N_{\text{cores}} \ \ [\text{CPUh}] \ , \tag{6.5}$$

which is the typical currency of supercomputing facilities. For GPU hardware, "core" has a different meaning and node hours might be more approriate for heterogeneous architectures. The

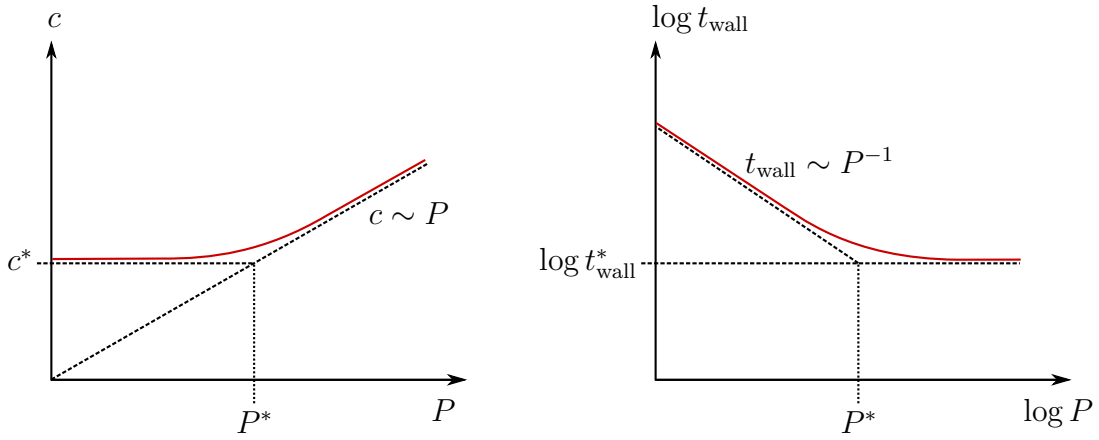


Figure 6.2: Illustration of computational costs $c = t_{\text{wall}} \cdot N_{\text{cores}}$ (left) and wall-clock time t_{wall} (right) as a function of the number of processors P. Optimal scalability is assumed until the strong-scaling limit (indicated by $*$) is reached, and $t_{\text{wall}} = t^*_{\text{wall}}$ for $P \geq P^*$.

main advantage of this metric is that the amount of computational costs required by a simulation is independent of the number of processors used to simulate the problem as long as one is operating away from the strong-scaling limit. This is somewhat orthogonal to defining computational costs as the wall-clock time (time-to-solution metric), which scales inversely proportional to the number of processors away from the strong-scaling limit and becomes minimal in the strong-scaling limit. This is illustrated in Figure 6.2. The suitability of both metrics depends on the constraints that render a simulation "expensive". CPUh as cost metric is appropriate if aquisition and operating costs dominate, which is the typical use case for research groups applying for a certain amount of compute resources on a supercomputer. Furthermore, the CPUh metric assumes that computational resources are limited and need to be distributed between users. As argued in Löhner (2019), in an industrial context the amount of time after which simulation results are available (with "overnight" as the relevant time scale) might be more critical. Companies might be willing to invest into large compute clusters to achieve this goal, since this might be less expensive than a delay in production or exceeding project deadlines. The time-to-solution metric therefore assumes infinite compute resources. The metric CPUh also has the advantage that it is a suitable metric for simulations performed on the node-level, allowing to compare the computational efficiency of different codes that are not optimized for parallel scalability. The present work mainly uses CPUh as cost metric, but results are also discussed for the alternative wall-time metric in order to highlight the impact of the chosen cost metric on the decision of finding the optimal polynomial degree that provides the best overall efficiency.

The comparative study by Wang et al. (2013) normalizes computational costs (in CPUh as defined above) in order to obtain non-dimensional work units. The `TauBench` code is run on the same hardware for a well-defined problem size and number of time steps, which defines the cost of one work unit. This procedure has both advantages and disadvantages. On the one hand, hardware progress is extracted from the cost metric (under the assumption that the `TauBench` reference code and the code of interest profit similarly from progress made in hardware). If reduced costs can be reported in terms of work units, this indicates that improvements have been

made in numerical algorithms rather than faster hardware. On the other hand, one might consider hardware progress as one factor among others enabling more efficient simulations of CFD problems and, therefore, waive a normalization of costs. Further, the dependency on another software required to define one work unit appears impractical. Instead, the same effect is achieved if the memory bandwidth of the hardware (the accumulated bandwidth of all nodes occupied by a simulation) was used for normalization of costs, a quantity that is easily accessible for both CPU and GPU hardware. This leads to a cost metric of unit Byte, the amount of data that can be transferred to/from main memory during the time a simulation has occupied a certain amount of resources. This is in line with the observation that memory-bandwidth is often the main bottleneck for PDE solvers on current hardware instead of arithmetic throughput (Flops), see also the discussion in Chapter 4.

A disadvantage of the metric CPUh is that comparisons to other hardware such as graphics processing units is difficult. For this purpose, energy consumption appears to be a more appropriate metric to allow a fair comparison between CPU and GPU implementations. Since the heterogeneity in hardware can be expected to grow in the future and Flops can be expected to become a less relevant metric on future hardware for many applications in scientific computing (Ibeid et al. 2020), energy consumption could establish itself as a unifying metric. Yet another metric is suggested in Vermeire et al. (2017), where resource utilization, defined as the purchase price of a hardware times the wall-clock time this hardware is occupied for a simulation, is used as cost metric. This metric also aims at allowing fair comparisons between CPU and GPU implementations. The downside is, however, that GPUs can typically not be used standalone but need a host as well as a network in a massively parallel context (with certain additional purchase costs), that the share of the hardware used for scientific computing is rather small in terms of the overall market in order to stimulate pricing, and that the cost-performance relationship for different models provided by a hardware manufacturer is rather nonlinear and further difficult to compare between companies. From such a perspective, this cost metric appears to be rather suitable to guide supercomputing facilities in deciding which hardware to buy, than to address the question of scientific interest of identifying the software and implementation that makes most efficient use of a certain class of hardware.

Remark 6.5 *Combining equations* (6.2) *and* (6.3) *leads to the conclusion that computational costs defined as the product of wall-time and number of cores increases at least as fast as* Re^3 *for increasing Reynolds number. Note, however, that it would be a premature conclusion that the larger the supercomputer the higher the maximum Reynolds number that can be simulated, even though the promise or hope that exponential growth of the peak performance of supercomputers according to Moore's law will solve the problem is common textbook knowledge (Ferziger and Peric 2008, Jiménez 2003, Pope 2001, 2004). Such a conclusion is essentially based on the assumption that parallelization can be exploited equally in space and time. State-of-the-art turbulent flow solvers use a method-of-lines approach, solving systems of equations of size* N_{DoFs} *in each time step with a domain-decomposition approach for parallelization in space. For this "standard" approach, the promise that larger computers will continuously extend the range of DNS-realizable Reynolds numbers assumes a situation termed "unconstrained case" in Figure 6.1. As discussed above, the large number of time steps to be completed in a certain wall-time limit might, however, be the limiting factor (Löhner 2019). The challenge in extending the range of applicability towards higher Reynolds numbers then lies in being able to solve larger and*

larger systems of equations in lower and lower wall-times when operating in the strong-scaling limit (with the relevant limiting hardware characteristics differing from those for the unconstrained case, see Figure 6.1). Extrapolations such as "Even if we take the conservative estimate of an increase in computer speed by a factor of 100 every decade, we will be able to run our ultimate LES [refers to LES with 10000^3 *grid points] in 2 days by 2020, in 30 min by 2030, and in 20 s by 2040." taken from Jiménez (2003) assume that the increase in peak performance is accompanied by a decrease in network latency and increase in network bandwidth by the same factor, which is an unrealistic scenario according to (Hager and Wellein 2010, Figure 3.1). In order to highlight the different origins of limiting resources, equations* (6.2) *and* (6.3) *are kept separated in the present work.*

6.2 Efficiency models

The methodology presented in this section has already been published in Fehn et al. (2018a). The time-accuracy-size spectrum analysis published later in Chang et al. (2018) exhibits many parallels to the efficiency model discussed here. The present modeling approach can be considered more general since it naturally includes time-dependent problems and uses computational costs instead of wall-time.

6.2.1 A general efficiency model for PDE solvers

This section discusses a general efficiency model for unsteady PDE solvers that are based on time stepping techniques according to a method-of-lines discretization approach separating discretization in space and time. A further assumption is that the system of linear or nonlinear equations to be solved within each time step is addressed by iterative solution techniques. Expanding equation (6.1) allows to identify three main contributions to the efficiency of PDE solvers under the given assumptions

$$\text{efficiency} = \underbrace{\frac{\text{accuracy}}{\text{DoFs} \cdot \text{timesteps}}}_{\text{discretization}} \cdot \underbrace{\frac{1}{\text{iterations}}}_{\text{solvers/preconditioners}} \cdot \underbrace{\frac{\text{DoFs} \cdot \text{timesteps} \cdot \text{iterations}}{\text{computational costs}}}_{\text{implementation}} , \tag{6.6}$$

or in a more mathematical notation

$$E(h,k,\Delta t) = E_{h,k,\Delta t}(h,k,\Delta t) \cdot E_{\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}}(h,k,\Delta t) \cdot E_{\mathbf{Ax}}(h,k) \; . \tag{6.7}$$

According to this model, the three main factors for an efficient numerical method are aspects of discretization in space and time (Chapter 2), efficient iterative solvers and preconditioners (Chapter 5), and the efficient evaluation of discretized PDE operators (Chapter 4).

The efficiency of the spatial and temporal discretization is denoted as $E_{h,k,\Delta t}$ and depends on the characteristic element length h, the polynomial degree k of the shape functions, and the time step size Δt. The order J of the time integration scheme does not appear explicitly as a parameter since it is considered to be constant; typically $J = 2, 3$ is used for the type of incompressible Navier–Stokes solvers discussed in this work. Apart from aspects of robustness and stability, a discretization is considered efficient if it requires few degrees of freedom or

few time steps to reach a desired accuracy. The selection of discretization schemes is often one of the first decisions to be made when developing new PDE solvers. Quantifying accuracy is straightforward if an analytical solution is available, but might be challenging in the absence of exact reference solutions. This is particularly important when dealing with large-eddy simulation of turbulent flows, since the accuracy of a particular method is difficult to evaluate in case that only experimental results or numerical reference solutions of the same level of accuracy are available, and since statistical errors are also present for this type of problems.

The efficiency of solvers and preconditioners is denoted as $E_{\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}}(h, k, \Delta t)$, where state-of-the-art iterative solution techniques are considered in the present work to solve linear or nonlinear system of equations, aiming at optimal-complexity algorithms with costs scaling linearly with the total number of unknowns (e.g., Krylov solvers with multigrid preconditioners). This selection is justified by the fact that other approaches such as direct solvers with sub-optimal complexity would render the solution of large algebraic systems of equations, e.g. arising from numerical models of LES or DNS-type for turbulent flows, prohibitively expensive. The efficiency $E_{\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}}$ mainly depends on the type of preconditioner used to solve these equations. Note that iterations should not be understood as the number of outer iterations required to solve the linear system of equations, but rather as an effective number of matrix–vector products (operator evaluations) applied during the whole iterative solution procedure including preconditioning operations, see for example the quantity $n_{10,\mathbf{Ax}}$ defined in Section 5.3.1. Hence, this term also includes other operations such as level 1 BLAS vector operations (addition, scaling, inner product) which can account for a significant part of the overall computational costs once operator evaluation is sufficiently optimized (Kronbichler and Allalen 2018). This quantity is not defined explicitly here since this aspect is typically difficult to investigate theoretically by cost models, but should be investigated numerically. The condition number and, hence, the efficiency of solvers and preconditioners $E_{\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}} = E_{\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}}(h, k, \Delta t)$ depends on the parameters of the spatial and temporal discretization, which also implies that different preconditioners are most efficient for different equations or parameters.

The efficiency of the implementation $E_{\mathbf{Ax}}$ is defined as the number of degrees of freedom per time step and per operator evaluation that can be processed within a given amount of computational costs. In Chapter 4, this metric has also been introduced as throughput. The fact that the efficiency of solvers/preconditioners and the efficiency of the implementation are written as separate factors in equation (6.7) is based on the assumption that the application of discretized operators is the basic ingredient of iterative solution techniques such as Krylov solvers and multigrid preconditioners. To efficiently evaluate discretized finite element operators for high polynomial degrees, a high-performance implementation based on matrix-free operator evaluation is used (Chapter 4). Equation (6.7) naturally demonstrates that the metric Flop/s often used to measure the efficiency of implementations is less relevant, i.e., performance optimizations should always target an increase in throughput rather than floating point operations. There is no direct relation between the number of floating point operations performed per second and the number of degrees of freedom processed per second because the algorithm selection is not fixed and a variety of possible implementations is available. The above efficiency model also reveals that all the components of solvers and preconditioners should be implemented in a matrix-free way in order to obtain a method that is efficient as a whole. In general, the efficiency of the implementation $E_{\mathbf{Ax}} = E_{\mathbf{Ax}}(h, k)$ depends on the parameters of the spatial discretization, where

the throughput would be independent of the mesh resolution and the polynomial degree in case of an optimal implementation.

While the individual factors have been introduced and discussed separately in previous chapters, it is clear that these are highly inter-connected. The type of temporal discretization approach essentially defines the type of algebraic systems of equations to be solved, e.g., nonlinear systems of equations for implicit formulations, linear systems of equations for mixed implicit-explicit formulations, and no algebraic systems of equations for fully explicit formulations; or a saddle-point problem for a monolithic solution of the incompressible Navier–Stokes equations versus a set of symmetric positive definite systems of equations for splitting-type approaches. This does not only result in different algebraic systems of equations, but also in different time step limitations and preconditioners of different complexity most suitable to solve these equations, which in turn have a direct impact on the computational efficiency of the overall approach. The choice of the spatial discretization method (DG versus HDG) affects the resulting system of equations and the type of preconditioning most suitable to solve these equations. The implementation strategy (matrix-free versus matrix-based) directly affects the efficiency of evaluating discretized operators, but also how efficiently algebraic systems of equations can be solved since operator evaluation is a main ingredient in iterative solvers. Deciding for an HDG approach, which is typically inherently linked to the idea of a matrix-based solution of systems of equations, should be based on much more than degrees of freedom or matrix size. As argued in Chapter 4, a matrix-based approach is difficult to realize efficiently for high-order discretizations on modern hardware. In fact, the matrix-free implementation used in this work is the central design choice in this context. At the same time, it prevails the use of certain preconditioning strategies (such as algebraic multigrid) or renders the design of robust preconditioners for non-symmetric problems more challenging than in a matrix-based environment with direct access to the matrix entries. Having said this, it might be possible that partially matrix-based preconditioners outperform a matrix-free variant in the range of intermediate degrees $k = 2, 3, 4$. Finally, note that the type of interpolation basis and quadrature rule chosen for a high-order discretization approach affects the conditioning of systems of equations and impacts the number of iterations required by iterative solvers, but also the speed at which discretized operators can be evaluated.

Remark 6.6 *The above efficiency model does not explicitly include the aspect of parallel scalability. Ideally, the overall efficiency does not depend on the parallelization of the numerical algorithm, i.e., the computational costs are constant in case of ideal parallel scalability of the code according to $t_{\text{wall}} \sim 1/P$. Formally, parallel scalability could be understood as part of the efficiency of the implementation, reducing throughput in case of sub-optimal parallel efficiency, e.g., due to ghost layer communication. Alternatively, the above efficiency model could be understood as a model for the node-level performance that needs to be extended by an additional factor $\eta(h, k, \Delta t, P)$ for the parallel efficiency, which depends – apart from the discretization parameters – on the number of processors P used for parallel computations.*

Remark 6.7 *The above efficiency model contains other hidden parameters. An important factor on efficiency is for example the use of* `float` *vs.* `double` *precision computations, which can be seen as an additional parameter of the implementation. In the present work, this is exploited by mixed-precision multigrid preconditioners in order to improve efficiency, see Chapter 5. The present matrix-free implementation also distinguishes between affine and deformed elements for*

optimal efficiency, see Chapter 4. Another important parameter of iterative solvers and preconditioners are solver tolerances ε. These should be chosen small enough to not affect accuracy, but for efficiency reasons algebraic system of equations should not be over-solved. These parameters are not highlighted explicitly since they are typically chosen once and for all and since the primary interest is in the $\{h, k, \Delta t\}$ parameter space.

Remark 6.8 *Too strong an encapsulation of the three aspects in equation* (6.7) *often hinders the development of efficient numerical methods when focusing on only a single aspect with black-box interfaces to the other aspects. Many examples can be given in this context, which are discussed controversially in the literature and where conclusions are far from obvious. The items listed below explain how an efficiency model as the one proposed here can help to avoid certain pitfalls if another perspective is taken:*

- *Discretization: Explicit time integration schemes are often motivated due to better parallel scalability compared to implicit solvers that involve global communication. However, optimal time integration schemes can not be designed with a look at the maximum possible time step size only (favoring implicit schemes, see Franciolini et al. (2017)) or by considering the wall-time per time step only (favoring explicit schemes, see Löhner (2019)). The wall-time per characteristic time of the flow problem would be much more suitable to compare methods with different characteristics in terms of explicit time step restrictions. Optimality of spatial discretization schemes is often evaluated with a view on optimal convergence behavior for smooth problems only, or by evaluating the costs of discretization schemes by counting unknowns, the non-zeros of a matrix, or operation counts (Flops). For example, this does not give a realistic estimate of how methods with adaptive mesh refinement perform relative to methods with uniform mesh refinement, or how matrix-based methods perform relative to matrix-free methods.*

- *Solvers: Developing iterative solvers and preconditioners with a view on iteration counts or multigrid convergence rates only does not necessarily lead to fast solvers. Adhering to black-box matrix-based interfaces for iterative solvers and preconditioners will not allow to take advantage of fast matrix-free methods. Motivation for HDG often falls short by concentrating on the reduction in problem size by static condensation instead of comparisons to fast matrix-free techniques operating on the full problem size.*

- *Implementation: Parallel scalability is used to motivate DG methods in the literature due a communication pattern with less neighbors compared to continuous Galerkin discretizations (Fischer 2015). At the same time, face integrals introduce additional costs with non-negligible communication costs on the node level (Kronbichler and Wall 2018, Kronbichler et al. 2019).*

6.2.2 Optimality assumptions

The efficiency model (6.7) and its dependency on parameters can be simplified by certain optimality assumptions regarding the implementation and regarding iterative solvers and preconditioners. As discussed in Chapter 4, the efficiency of the implementation does not depend on

the mesh size h (or problem size) for fixed polynomial degree k as long as the workload is sufficiently high. Furthermore, numerical results for the present matrix-free implementation have shown that the throughput depends only mildly on k for a fixed problem size in three space dimensions, and is almost independent of k in two space dimensions. Motivated by these observations, the implementation is denoted as optimal if it holds

$$E_{\mathbf{Ax}}(h,k) \approx f(k) \approx \text{const} \,. \tag{6.8}$$

Iterative solvers and preconditioners are termed optimal or robust if the convergence rate is independent of the mesh resolution parameters h, k. This property is also termed mesh-independent convergence and allows to construct solvers with costs scaling linearly with the number of unknowns N_{DoFs} under assumption (6.8). In Chapter 5, preconditioners are discussed for which iteration counts are independent of h. The multigrid preconditioners with polynomial Chebyshev smoothing depend mildly on the polynomial degree k of the shape functions. This leads to the following simplifying assumption in case of optimal solvers

$$E_{\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}}(h,k,\Delta t) \approx g(k) \approx \text{const} \,. \tag{6.9}$$

Note that the absolute costs of a preconditioner are also important apart from its asymptotic complexity. For operators including the mass matrix operator scaled by the inverse of the time step size, the linear system of equations might be solved more efficiently in the context of high-order, matrix-free DG methods by using the inverse mass matrix as a preconditioner instead of multigrid preconditioners. In that case, the efficiency also mildly depends on $h, k, \Delta t$, but the absolute efficiency is improved as compared to more complex preconditioners. In practice, this preconditioner often allows to obtain robust and fast convergence in case that the time step size is selected according to the CFL condition, i.e., if the time step size is reduced under mesh refinement.

6.2.3 Optimal selection of time step size

Under the optimality assumptions introduced above, the spatial discretization and temporal discretization emerge as the main factors impacting the overall efficiency of the numerical method w.r.t. the parameters $h, k, \Delta t$. A remaining question is how to select the time step size in an optimal sense. The goal is to express the time step size Δt as a function of the parameters h, k such that overall efficiency is maximized. Generally speaking, the time step size should be selected such that spatial and temporal discretization errors are balanced. The argumentation behind is that one could otherwise either increase the time step size or reduce the spatial resolution without increasing the overall error, which would lead to a more efficient simulation. The optimal time step size Δt_{opt} is illustrated in Figure 6.3, which schematically shows the overall discretization error as a function of the time step size for a fixed spatial resolution. The optimal time step size can be expressed analytically under the assumption of optimal rates of convergence in space, $\varepsilon_{h,k} = C_{h,k}(k) h^{k+1}$, and time, $\varepsilon_{\Delta t} = C_{\Delta t}(J) \Delta t^J$, leading to

$$\varepsilon_{h,k} \overset{!}{=} \varepsilon_{\Delta t} \rightsquigarrow \Delta t = C_{\Delta t_{\mathrm{opt}}}(k,J) h^{\frac{k+1}{J}} \,. \tag{6.10}$$

However, some of the time integration strategies discussed in this work introduce a restriction of the time step size according to the CFL condition, which might be restrictive or non-restrictive

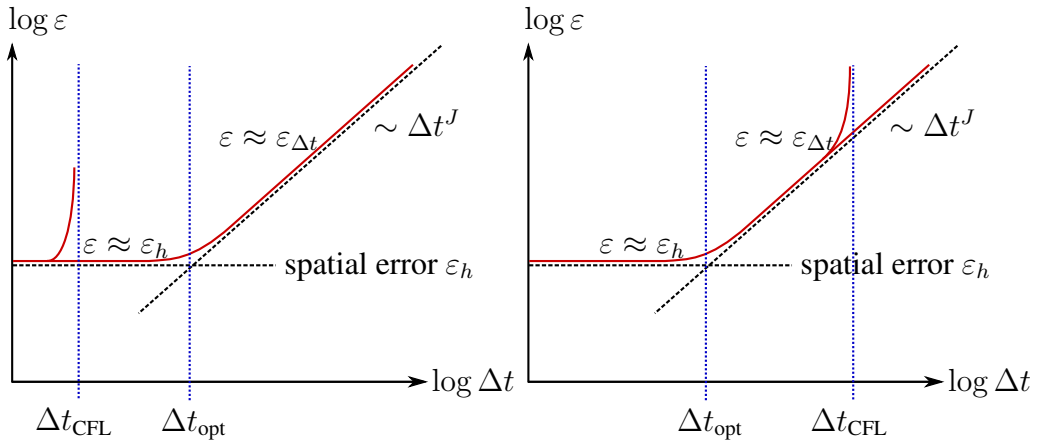


Figure 6.3: Illustration of selection of time step size in order to optimize efficiency for a given spatial resolution. The CFL condition is restrictive on the left ($\Delta t_{\mathrm{CFL}} < \Delta t_{\mathrm{opt}}$) and non-restrictive on the right ($\Delta t_{\mathrm{opt}} < \Delta t_{\mathrm{CFL}}$).

in terms of the optimal selection of the time step size. Both cases are visualized in Figure 6.3. In terms of the notation used here, the CFL condition (2.209) is written as follows (considering the case with constant time step size is sufficient for the purpose of efficiency models discussed here)

$$\Delta t = C_{\Delta t_{\mathrm{CFL}}}(k, r, \mathrm{Cr})h \ . \tag{6.11}$$

In the limit $k \to \infty$, the condition (6.10) can be expected to be more restrictive than the condition (6.11) due to the more rapid exponential decrease of the time step size $h^{\frac{k+1}{J}}$ compared to the CFL-type algebraic decrease according to k^{-r}. For small and moderate polynomial degrees, a typical observation is that the CFL condition is indeed restrictive even if asymptotic, optimal rates of convergence in space are observed with comparably low spatial errors. In cases where optimal spatial convergence as assumed in equation (6.10) is inappropriate (as typically observed in under-resolved scenarios), it can be expected that the CFL condition becomes more restrictive. For ease of notation, the following unifying notation is introduced

$$\Delta t = C_{\Delta t}(k)h^s \ , \tag{6.12}$$

where the parameter s depends on the type of time step criterion. Exemplarily, the following three cases are distinguished:

- $s = 0$: constant time step size independent of h and small enough such that the spatial error is dominant (includes steady problems)
- $s = 1$: CFL-type time step restriction (typical use case for practical simulations such as LES and DNS of turbulent flows)
- $s = \frac{k+1}{J}$: time step size which balances spatial and temporal errors (assuming optimal rates of convergence in space and time)

6.2.4 Analytical efficiency estimates

Similar to equation (6.6), the following relation can be derived for the computational costs

$$\text{computational costs} = \underbrace{\text{DoFs} \cdot \text{timesteps}}_{\text{discretization}} \cdot \underbrace{\text{iterations}}_{\text{solvers/preconditioners}} \cdot \underbrace{\frac{\text{computational costs}}{\text{DoFs} \cdot \text{timesteps} \cdot \text{iterations}}}_{\text{implementation}} . \tag{6.13}$$

Under the assumptions of Section 6.2.2, the factors related to iterative solvers and the implementation are assumed constant w.r.t. the mesh size h and mildly dependent on k, so that the computational costs c depend mainly on the number of unknowns N_{DoFs} and the number of time steps $N_{\Delta \mathrm{t}}$

$$c = N_{\mathrm{DoFs}} N_{\Delta t} f^{-1}(k) g^{-1}(k) \, . \tag{6.14}$$

The number of unknowns can be expressed as a function of the discretization parameters h and k as follows

$$N_{\mathrm{DoFs}} \sim (k+1)^d \, h^{-d} \, . \tag{6.15}$$

The number of time steps is inversely proportional to the time step size, $N_{\Delta \mathrm{t}} \sim \Delta t^{-1}$. Using equation (6.12) yields

$$N_{\Delta t} \sim C_{\Delta t}^{-1}(k) h^{-s} \, . \tag{6.16}$$

To investigate the efficiency of high-order methods from an analytical perspective, the goal is to obtain error-cost-relations. This requires assumptions about the error behavior (convergence) as a function of the discretization parameters. As introduced in Section 6.1.2, one can distinguish between smooth and non-smooth solutions (or asymptotic and pre-asymptotic convergence) as detailed below.

6.2.4.1 Smooth solutions or asymptotic convergence behavior

In the asymptotic regime, the error decreases under h-refinement with optimal rates of convergence according to

$$\varepsilon_{h,k} = C_{h,k}(k) h^{k+1} \, . \tag{6.17}$$

The computational costs can be expressed as follows by inserting equations (6.15) and (6.16) into equation (6.14)

$$c = C_{\Delta t}^{-1}(k) f^{-1}(k) g^{-1}(k) \, (k+1)^d \, h^{-(d+s)} \, . \tag{6.18}$$

Equations (6.17) and (6.18) describe error and costs as a function of h (and other parameters). To obtain the desired ε-c-relation (error-cost-relation) for mesh refinement studies, the parameter h needs to be eliminated

$$\varepsilon(c) = \underbrace{C_{h,k}(k) \, (C_{\Delta t}(k) f(k) g(k))^{-\frac{k+1}{d+s}} \, (k+1)^{\frac{d(k+1)}{d+s}}}_{=C(k,d,J)} \, c^{-\frac{k+1}{d+s}} \, . \tag{6.19}$$

Then, depending on the type of time step criterion used (parameter s), the following three cases can be distinguished:

- $s = 0$: This is the case for which the error decreases most rapidly when investing more computational costs (i.e., for a series of simulations with decreasing mesh size h)

$$\varepsilon(c) \sim c^{-\frac{k+1}{d}} \ . \tag{6.20}$$

- $s = 1$: For the practical use case of a CFL-type time step restriction, the error decreases as

$$\varepsilon(c) \sim c^{-\frac{k+1}{d+1}} \ . \tag{6.21}$$

- $s = \frac{k+1}{J}$: The academic case of balancing spatial and temporal errors leads to the error-cost-relation

$$\varepsilon(c) \sim c^{-\frac{k+1}{d+\frac{k+1}{J}}} = c^{-\frac{J}{1+\frac{dJ}{k+1}}} \ , \tag{6.22}$$

  with the limit $\varepsilon(c) \to c^{-J}$ for $k \to \infty$.

In all cases, the error decreases the faster the larger the polynomial degree when increasing computational costs. Since the slope of the curves in a $\log \varepsilon$-$\log c$-diagram becomes steeper with increasing polynomial degree k, it can be expected that high-order methods are asymptotically more efficient than low-order methods. The results shown in Section 6.3 for a problem with smooth, analytical solution confirm this expected behavior. The proportionality constant $C(k, d, J)$ in equation (6.19) is difficult to model since it depends on many parameters and is further problem dependent, so that this influence is best studied by numerical experiments. The third case has the interesting limit that the slope approaches a constant value for large k that depends only on the order J of the time integration scheme. The reason is that the spatial error decreases exponentially with k, but also the number of time steps (and hence the computational costs) increases exponentially with k in order to keep temporal discretization errors as small as spatial errors. This is exactly the case where one would argue for the use of higher order time stepping.

6.2.4.2 Non-smooth solutions or pre-asymptotic convergence behavior

When it comes to complex engineering applications with complex geometry and high-Reynolds-number turbulent flow problems, it is not realistic that optimal rates of convergence in space can be observed. Under these circumstances, the convergence behavior is better described by

$$\varepsilon_{h,k} = C_{h,k}(k) N_{\mathrm{DoFs}}^{\mathrm{const}} \ , \tag{6.23}$$

where the constant exponent, which is different from the optimal value $-\frac{k+1}{d}$, expresses that methods of different polynomial degree k show similar rates of convergence for comparable spatial resolutions (N_{DoFs}). The only arguments in favor of high-order methods are then better error constants $C_{h,k}(k)$, which might be motivated theoretically from dissipation and dispersion analysis of high-order methods, see for example the '1% rule' proposed in Moura et al. (2015, 2017a) characterizing the resolution capabilities of high-order methods for under-resolved turbulence. The time step size criterion (6.10) does not make sense in this case, since the underlying assumption of an optimal convergence behavior is invalid. Assuming the CFL-type time step

criterion to be the most relevant for such practical problems ($s = 1$), the number of time steps can be expressed as follows as a function of the number of unknowns

$$N_{\Delta t} \sim k^r h^{-1} = \frac{k^r}{k+1}\frac{k+1}{h} \sim \frac{k^r}{k+1} N_{\mathrm{DoFs}}^{\frac{1}{d}} \,. \tag{6.24}$$

Inserting equation (6.24) into equation (6.14) yields the following cost estimate

$$c = \frac{k^r}{k+1} f^{-1}(k) g^{-1}(k) N_{\mathrm{DoFs}} N_{\mathrm{DoFs}}^{\frac{1}{d}} \,. \tag{6.25}$$

Then, elimination of N_{DoFs} by inserting equation (6.25) into equation (6.23) yields the following ε-c-relation

$$\varepsilon(c) = C_{h,k}(k) \left(\frac{k+1}{k^r} f(k) g(k) \right)^{\frac{d \cdot \mathrm{const}}{d+1}} c^{\frac{d \cdot \mathrm{const}}{d+1}} \,. \tag{6.26}$$

This equation reveals that the un-modeled parts in the form of unspecified proportionality constants or factors showing some mild dependency on the polynomial degree are now the relevant factors that determine the efficiency of high-order methods compared to low-order methods. Accordingly, it is difficult to draw precise quantitative conclusions. Nevertheless, some qualitative statements can be made. On the one hand, the constant $C_{h,k}(k)$ principally favors high-order methods or, in other words, is the motivation for the use of high-order methods. On the other hand, the three factors $\frac{k+1}{k^r}$, $f(k)$, and $g(k)$ render high-order methods more expensive than low-order methods for the same number of unknowns. The influence of these three factors is investigated in detail in Fehn et al. (2018a) by the example of the three-dimensional Taylor–Green vortex problem. For the same number of unknowns, high-order methods typically require more time steps under CFL-type time step restrictions with $r > 1$ than low-order methods (factor $\frac{k+1}{k^r}$). The efficiency of the matrix-free operator evaluation can be expected to slightly decrease for large k (factor $f(k)$) according to the results in Chapter 4. Finally, iterative solvers often show iteration counts slightly increasing with k (factor $g(k)$) according to the results in Chapter 5. Although not explicitly modeled here, high-order methods contain less elements for the same number of unknowns and, therefore, exhibit less parallelism with potentially worse parallel scalability. For high-order methods to be more efficient overall, they must be sufficiently more accurate to compensate these factors, see also Brown (2010), Fehn et al. (2018a). Various numerical examples are shown in Section 6.3.

6.3 Numerical results

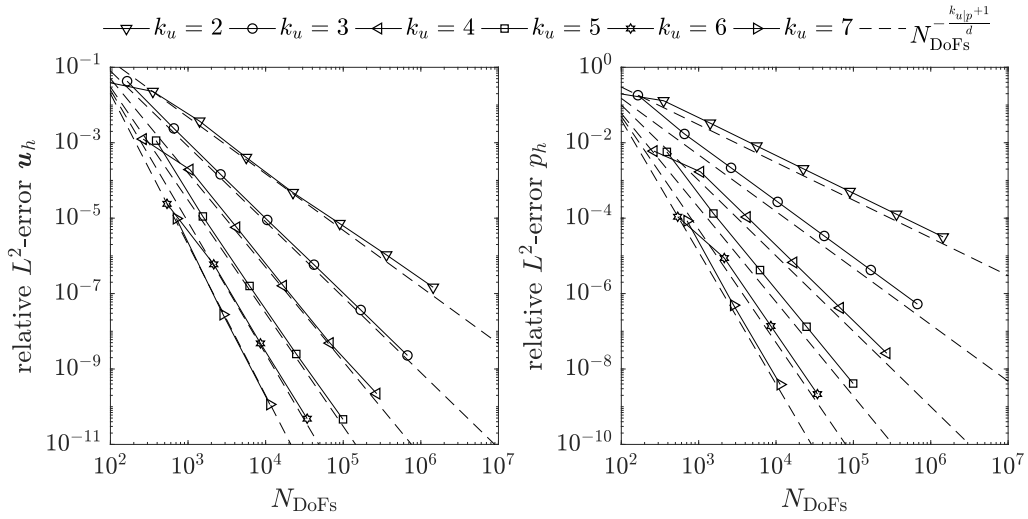
This section complements the theoretical considerations and models of this chapter by numerical experiments. A main focus is on the efficiency of high-order DG discretizations in terms of error vs. computational costs for typical two- and three-dimensional benchmark problems in computational fluid dynamics. Emphasis is also put on documenting the node-level performance of the present solver for incompressible turbulent flow simulations, its parallel scalability on large supercomputers, and performance comparisons to state-of-the-art high-order solvers from the literature.

6.3.1 Two-dimensional Taylor–Green vortex

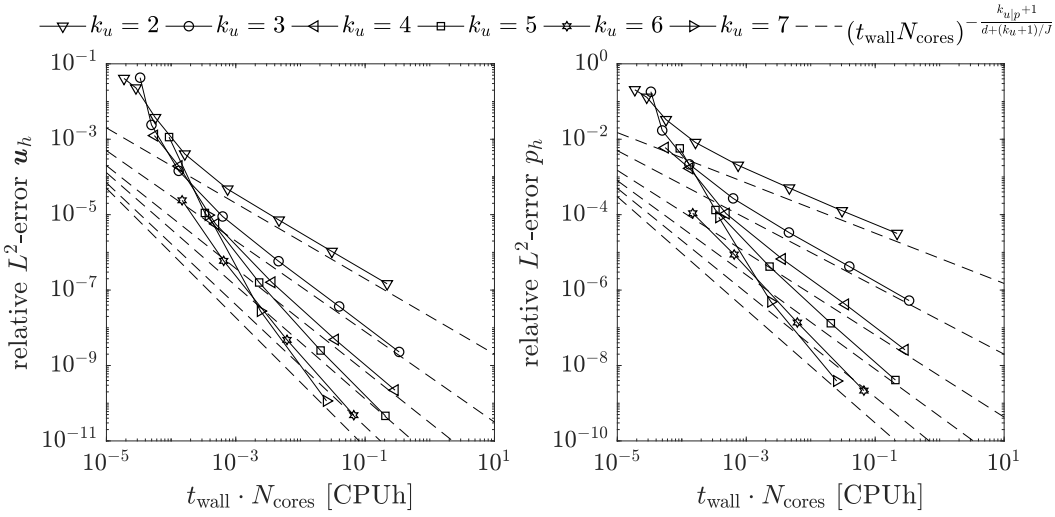
To verify the analytical efficiency models from Section 6.2.4, the two-dimensional Taylor–Green vortex problem is re-investigated. The problem has a smooth solution (see Section 2.6.5.1 for a description of the problem setup) and is therefore well-suited for these investigations. In the following, mainly two characteristic scenarios are considered: (i) the coupled solution approach with an implicit treatment of the convective term, where the time step size is calculated according equation (6.10) ensuring maximum efficiency in the asymptotic regime of convergence; and (ii) the dual splitting scheme with an explicit treatment of the convective term, where the time step size is calculated according to the global CFL criterion (2.209). The BDF3 scheme is used in both cases to allow large time steps and to maximize efficiency. Solver tolerance are $\varepsilon_{\text{abs}} = 10^{-12}$ and $\varepsilon_{\text{rel}} = 10^{-6}$, where linearized system of equations within the Newton solver are solved with a relative tolerance of $\varepsilon_{\text{rel}} = 10^{-2}$ for improved efficiency. Polynomial degrees in the range $k = 2, 3, 4, 5, 6, 7$ are considered and refinements levels from $l = 1$ up to $l = 8$ (for the lowest polynomial degree). All simulations are performed in serial using one core of an Intel Haswell architecture due to the small problem sizes of this two-dimensional problem. Since the coarsest mesh consists of only 4 elements (and a few hundred degrees of freedom overall, depending on k), it can be expected that the performance is limited by latency effects for the coarsest mesh and low mesh refinements levels, see Figure 4.8.

For the implicit coupled solver, a Newton–Krylov solution approach is used, where the linearized system of equations is solved by FGMRES(100) using a block triangular preconditioner. Within this preconditioner, the velocity block is approximated by one multigrid V-cycle of ph-type with a matrix-based block-Jacobi smoother with one smoothing step in pre- and post-smoothing and a relaxation factor of $\omega = 0.7$, and GMRES with point Jacobi preconditioner as coarse-grid solver. The pressure convection–diffusion preconditioner is used for the Schur complement block, where the Laplace operator is inverted approximately by one multigrid V-cycle of cph-type with polynomial Chebyshev smoother of degree 5 and a Chebyshev coarse-grid solver. The preconditioner is updated every time step (and every 10 Newton iterations). This setup provides a preconditioner with robust iteration counts for the fully implicit coupled solver and the chosen parameters in terms of time step size, mesh size, and polynomial degree. The postprocessing step with penalty terms uses a conjugate gradient solver preconditioned by the inverse mass matrix. The number of iterations actually decreases for high refine levels due to a decrease of the time step size, which renders the inertial term more dominant and improves conditioning. The constant $C_{\Delta t_{\text{opt}}}$ in equation (6.10) is determined individually for each polynomial degree by trial and error.

For the semi-implicit dual splitting scheme, all system of equations are solved by the conjugate gradient method. The pressure Poisson equation is preconditioned by a cph-multigrid method with Chebyshev smoother of degree 5 and Chebyshev coarse-grid solver. The viscous step and penalty step use the inverse mass matrix preconditioner. This setup again ensures robust and efficient preconditioning for the considered range of parameters. Iteration counts of the pressure Poisson equation and the penalty step are constant or decrease for increasing mesh refinement level, while iteration counts of the viscous step increase for increasing mesh refinement level (but the inverse mass matrix preconditioner is still most efficient for iteration counts below 20 to 30 observed here). The Courant number is $\text{Cr} = 0.125$ for all polynomial degrees. Additional results



(a) error versus number of unknowns

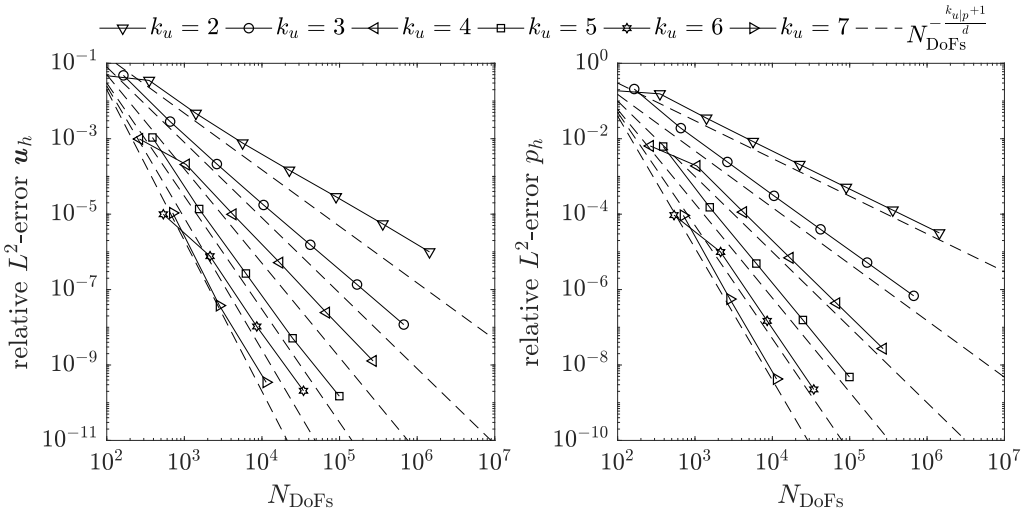


(b) error versus computational costs

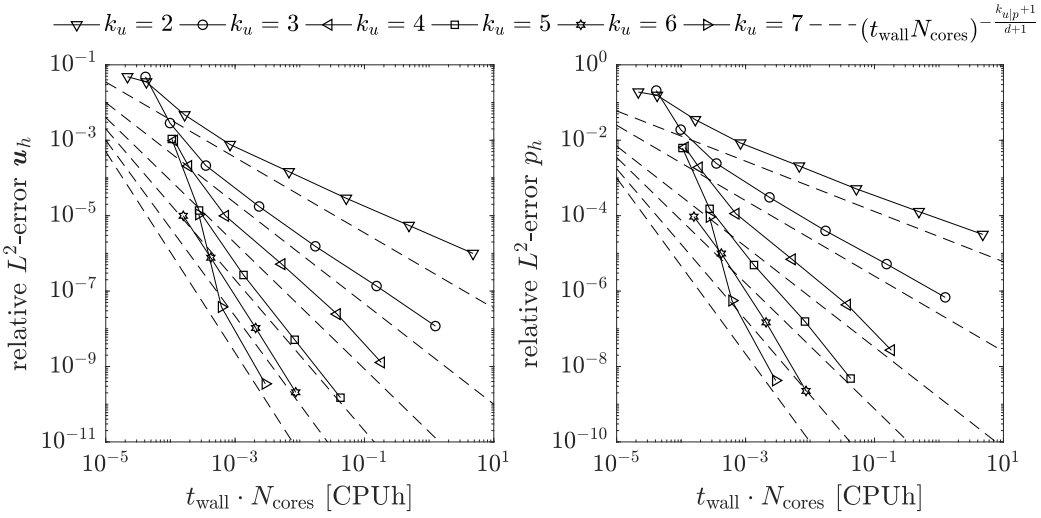
Figure 6.4: Vortex problem: efficiency of high-order methods for coupled solution approach with implicit treatment of the convective term (BDF3 scheme).

shown below for the pressure-correction scheme with explicit convective term use the same preconditioning strategies and the same time step criterion, but the BDF2 scheme for stability.

The results of these efficiency studies are shown in Figure 6.4 for the coupled solution approach and in Figure 6.5 for the dual splitting scheme. The error-vs-DoFs curves reveal that high-order methods are more accurate per degree of freedom than low-order methods for this smooth problem (note that spatial convergence rates for the velocity are found to be slightly sub-optimal in case of the dual splitting scheme for the chosen set of parameters). When studying the error-vs-costs curves, one observes that the behavior for larger mesh refinement levels is described very well by the analytical efficiency model (plotted as dashed reference slopes in the



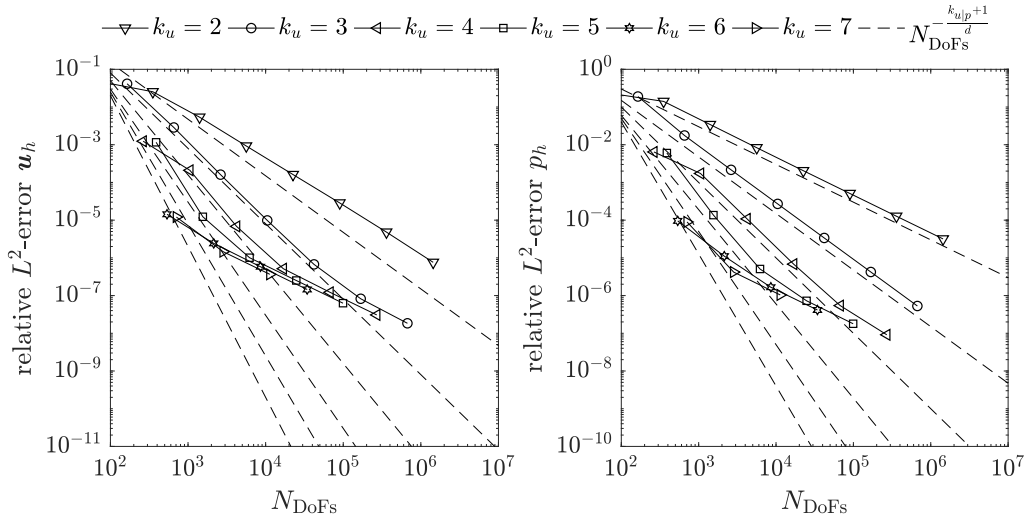
(a) error versus number of unknowns



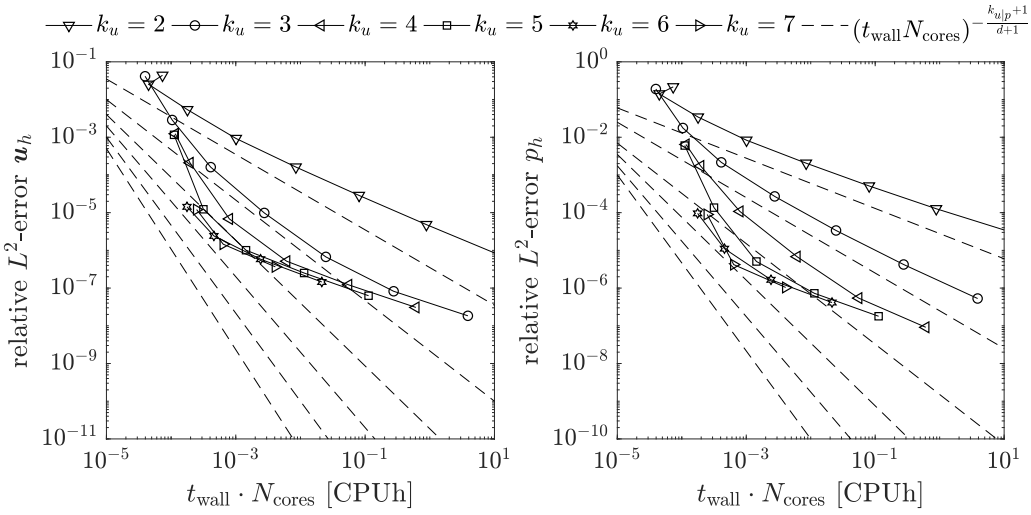
(b) error versus computational costs

Figure 6.5: Vortex problem: efficiency of high-order methods for dual splitting scheme with explicit treatment of the convective term (BDF3 scheme).

figures), equation (6.19) with $s = (k_u + 1)/J$ for the coupled solution approach and $s = 1$ for the dual splitting scheme. This behavior is to be expected since the assumptions of Section 6.2.2 are fulfilled, i.e., robust preconditioners are used that result in constant or slightly varying iteration counts, and the efficiency of the implementation is almost independent of h for sufficiently high workload, see for example Figure 4.8. However, there is a substantial deviation from the expected behavior for coarse meshes, where the curves in the error-vs-costs plane are shifted towards higher costs than expected according to the efficiency model. This behavior can be explained by latency effects, i.e., the workload is too low in order to achieve a saturated throughput. In the efficiency model, this effect can modeled as a reduced throughput of the matrix-free oper-



(a) error versus number of unknowns



(b) error versus computational costs

Figure 6.6: Vortex problem: efficiency of high-order methods for pressure-correction scheme with explicit treatment of the convective term (BDF2 scheme). To provide a point of reference, the dashed reference curves are identical to those shown in Figure 6.5 for the dual splitting scheme.

ator evaluation in equation (6.7) according to Figure 4.8, or alternatively as an additional factor η accounting for the sub-optimal parallel scalability.[3]

Overall, high-order methods are found to be more efficient than low-order variants with orders-of-magnitude improvements in costs for low error levels. Due to the simple geometry of this

[3]Although only a single core is used for these simulations, the effect is comparable to approaching the strong-scaling limit where efficiency reduces due to a small workload per core.

two-dimensional analytical test case and the related latency effects for coarse meshes, it is difficult to draw conclusion in the limit of large errors. Nevertheless, it can be summarized that the error-vs-costs behavior is mainly driven by the error-vs-dofs behavior. When comparing the results for the coupled solver with those for the dual splitting scheme, the improvements in efficiency from low-order to high-order are significantly larger in case of the dual splitting scheme. Moreover, while the coupled solver appears to be significantly more efficient for low polynomial degrees, the dual splitting scheme is significantly more efficient for high polynomial degrees. This is due to the number of time steps on the one hand, and the costs per time step on the other hand. For example, the dual splitting scheme requires a factor of approximately 450 more time steps for $l = 8, k = 2$, while it requires only a factor of 3.2 more time steps for $l = 3, k = 7$ compared to the coupled solver. On the contrary, the coupled solver is a factor of 11.2 more expensive per time step for $l = 8, k = 2$, while it is a factor of 28.0 more expensive per time step for $l = 3, k = 7$ (note that a partially matrix-based preconditioner with a complexity of at least k^4 in two dimensions is used here for the coupled solver, which was nevertheless found to be efficient compared to other preconditioners in preliminary tests for this two-dimensional example). Additional investigations for the coupled solution approach with an explicit treatment of the convective term yield an efficiency similar to the dual splitting scheme, where the costs per time step are a factor of 2 to 5 larger compared to the dual splitting scheme. For brevity, these results are therefore not shown explicitly here.

Additional simulations have also been performed for the pressure-correction scheme, where the BDF2 scheme is used to ensure stability. Using an explicit treatment of the convective term and choosing the time step size according to the CFL condition, the (k, l) parameter space can be divided into two parts. For low degrees and refinement levels, the CFL condition is restrictive and errors are dominated by spatial errors. In this regime, the pressure-correction scheme performs similar to the dual splitting scheme, see Figure 6.6. An implicit formulation of the convective term could prove efficient in this regime as for the coupled solver. However, it can be expected that the pressure-correction scheme does not reach the performance of the coupled solver in this regime since significantly smaller time steps are necessary due to the BDF2 scheme compared to BDF3 for the coupled solver. For high degrees and high refinement levels, the CFL condition is not restrictive and errors are dominated by temporal errors for the pressure-correction scheme as illustrated in Figure 6.6, i.e., time step sizes much smaller than the CFL condition would be necessary to maximize efficiency, see Figure 6.3. It can therefore be expected that the pressure-correction scheme (BDF2) is significantly less efficient than the dual splitting scheme (BDF3) in this regime. Since the CFL condition is not restrictive in this regime, an implicit treatment of the convective term is not promising. Although the time step selection is clearly non-optimal for the pressure-correction scheme according to Figure 6.3, these results are shown here intentionally in order to illustrate how a non-optimal choice affects efficiency.

To sum up, the implicit coupled solver using BDF3 is most efficient for low polynomial degrees by overcoming the CFL restriction, while the semi-implicit dual splitting scheme using BDF3 is most efficient for high polynomial degrees due to a fast solution per time step. Note, however, that these results should not be generalized since the observed behavior is mainly related to the fact that this example exhibits a very smooth solution that is easily resolved in space by meshes of only a few elements, i.e., for other examples one can often observe that second-order accuracy in time is not the limiting effect. One conclusion that can be drawn from the present results is that it is difficult to identify a solution strategy that performs best for all types

of examples or all spatial discretization parameters. This, in turn, motivates and justifies to put an emphasis on the flexibility w.r.t. different time integration schemes and solution strategies as done in this thesis.

6.3.2 Laminar flow around cylinder

This section re-investigates the laminar flow around cylinder example from Section 2.6.6 with a focus on computational efficiency. The aim is an error-vs-costs analysis as for the previous example. The dual splitting scheme is used, where all systems of equations are solved by the conjugate gradient algorithm. A multigrid V-cycle of cph-type with AMG coarse-grid solver is used for the pressure Poisson equation, and the inverse mass matrix preconditioner for the viscous step and penalty step. This setup has proven very efficient for the dual splitting scheme in many examples, especially for high Reynolds numbers. Polynomial degrees of $k = 2, 4, 6, 8, 10$ are considered and refine levels of $l = 0, \dots, 5$. The number of iterations are 3 to 7 for the pressure Poisson equation, 10 to 100 for the viscous step (increasing for increasing refine level due to relatively low Re), and 10 to 30 for the penalty step (decreasing for increasing refine level due to better resolution of the flow). Although iteration counts are not constant, the chosen preconditioners ensure a robust and highly efficient solution. The simulations are run on an Intel Haswell architecture and the number of cores is chosen such that simulations are performed away from the strong-scaling limit. The throughput in terms of degrees of freedom solved per time step and per core (the last two factors $E_{\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}}(h,k,\Delta t)\cdot E_{\mathbf{Ax}}(h,k)$ in equation (6.7)) is typically $(2\dots4)\cdot 10^5$ DoFs/s/core for all polynomial degrees. Hence, the optimality assumptions in terms of optimal iterative solvers and an optimal implementation are approximately fulfilled for this example. The overall efficiency of high-order methods compared to low-order methods is then determined by the efficiency of the spatial discretization and the temporal discretization. Regarding the latter, high-order methods require more time steps for the same number of unknowns due to the factor $k^{1.5}$ in the CFL condition. Putting all these factors together results in the overall efficiency shown in Figure 6.7. A superior efficiency of high degrees $k = 4, 6$ compared to $k = 2$ is evident for error levels of 10^{-4} and smaller. Higher polynomial degrees of $k = 8, 10$ are equally efficient but can not further improve the efficiency substantially. For low accuracy requirements with errors around 10^{-2}, the $k = 2$ discretization is highly competitive to the higher-order variants.

To investigate the potential gain in efficiency through fully implicit formulations in time, two cases are considered examplarily: a low polynomial degree of $k = 2$ with refinement level $l = 3$, and a high polynomial degree of $k = 10$ with refinement level $l = 1$. The pressure-correction scheme (BDF2, incremental, rotational) is investigated with an implicit treatment of the convective term. The implicit momentum equation is solved by FGMRES with a ph-multigrid preconditioner using a matrix-based block-Jacobi smoother (one pre- and postsmoothing step, relaxation factor $\omega = 0.7$) and a GMRES coarse-grid solver with block-Jacobi preconditioner. The preconditioner is updated every 10th time step. The pressure Poisson equation and the penalty step are solved as for the dual splitting scheme. The simulations are first run for a Courant number of $\mathrm{Cr} = 0.35$ as in the explicit case, and the Courant number is then successively increased by a factor of two, studying the impact on both computational costs and accuracy. The results are shown in Table 6.2. One first realizes that the implicit formulation introduces a performance overhead when using the same Courant number as for the explicit solver. For larger Courant

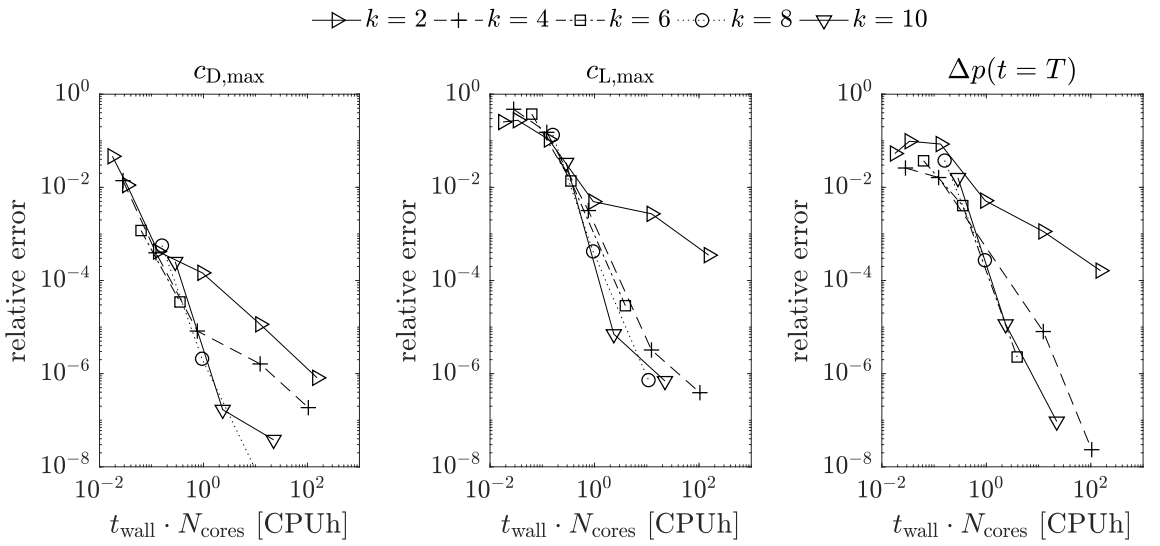


Figure 6.7: Laminar flow around cylinder: overall efficiency for various polynomial degrees $k = 2, 4, 6, 8, 10$. Relative errors of $c_{\mathrm{D,max}}$, $c_{\mathrm{L,max}}$, and $\Delta p(t = T)$ are shown as a function of the computational costs in CPUh in order to assess the computational efficiency of high polynomial degrees.

numbers, this overhead gets amortized due to fewer time steps, and a speed-up by a factor of approximately 4 for $k = 2$ and (only) 1.5 for $k = 10$ is achieved for the largest Courant numbers studied here. For the low-order case $k = 2$, the CFL condition is obviously restrictive, so that the Courant number can be increased by a factor of 8 without a substantial deterioration of accuracy. Hence, an efficiency advantage is achieved for $\mathrm{Cr} = 1.4, 2.8$ by the use of the implicit solver. Finally, the accuracy deteriorates substantially for $\mathrm{Cr} = 5.6$. For the high-order case $k = 10$, however, choosing larger time step sizes affects accuracy already for lower Courant numbers. Note that errors below 10^{-7} should be taken with care since the solution is already relatively close to the reference solution used for the calculation of errors, and a very small error (even for coarse spatial resolutions) might be obtained by coincidence due to the way the error norm is defined for this example. The results clearly reveal that errors increase by a factor of approximately 4 when doubling the Courant number due to the second-order temporal accuracy of the pressure-correction scheme. Hence, the use of an implicit solver does not prove effective for the high-order case. The coupled solver with implicit treatment of the convective term and BDF3 time integration scheme has also been investigated, where a block-triangular preconditioner is used that applies the block-Jacobi multigrid method described above for the velocity block and the pressure convection–diffusion preconditioner for the Schur complement block. The overall behavior is similar to the pressure-correction scheme, with the difference that the implicit coupled solver is a factor of 1.8 to 2 slower for the $k = 2$ simulation, and a factor of 1.3 to 1.5 for the $k = 10$ simulation. Although the implicit coupled solver is more accurate due to third-order accuracy in time, the accuracy of the results also deteriorates for the highest Courant numbers. It has not been possible to outperform the semi-implicit dual splitting scheme, and a detailed discussion of results is therefore omitted here for reasons of brevity.

Table 6.2: Laminar flow around cylinder: efficiency of fully implicit pressure-correction scheme (iPC) compared to explicit dual splitting scheme (DS).

(a) low-order case ($k = 2, l = 3$)

Cr	solver	costs [CPUh]	rel. error $c_{\mathrm{D,max}}$	rel. error $c_{\mathrm{L,max}}$	rel. error $\Delta p(t = T)$
0.35	DS	0.95	1.46E–004	4.95E–003	5.18E–003
0.35	iPC	1.97	2.43E–004	6.22E–003	5.00E–003
0.7	iPC	1.18	2.42E–004	7.49E–003	4.82E–003
1.4	iPC	0.62	2.41E–004	6.09E–003	3.68E–003
2.8	iPC	0.37	2.34E–004	9.05E–003	5.30E–003
5.6	iPC	0.24	2.01E–004	6.58E–002	7.74E–002

(b) high-order case ($k = 10, l = 1$)

Cr	solver	costs [CPUh]	rel. error $c_{\mathrm{D,max}}$	rel. error $c_{\mathrm{L,max}}$	rel. error $\Delta p(t = T)$
0.35	DS	2.34	1.68E–007	7.06E–006	1.15E–005
0.35	iPC	15.6	8.64E–009	3.81E–005	1.32E–006
0.7	iPC	11.1	6.45E–008	1.98E–004	2.45E–005
1.4	iPC	5.00	2.76E–007	8.22E–004	1.24E–004
2.8	iPC	2.79	1.11E–006	3.28E–003	4.53E–004
5.6	iPC	1.55	5.00E–006	1.29E–002	1.35E–003

Remark 6.9 *Specialized time stepping techniques with sub-stepping for the convective term as mentioned in Remark 2.1 promise a more direct route towards reducing computational costs. The reason behind is that a lower number of time steps directly translates into a faster solution as long as the costs per time step do not increase significantly. This is often the case for this type of methods since the explicit convective term has only a small share of overall computational costs and since the type of equations to be solved does not change, which does change when turning to implicit solvers. Put differently, this technique may relax the bottleneck related to the solution of the pressure Poisson equation (and other systems of equations) compared to the comparably cheap convective term (which, however, causes the restriction of the time step size). To address this aspect, the study by Riccius (2019) has assessed the potential of these sub-stepping techniques critically within the metric of overall efficiency in the context of the present discretization schemes. In fact, a substantial speed-up could be achieved, but a simultaneous loss in accuracy has been reported that did not allow to improve overall computational efficiency in general. With a view to the current state-of-the-art (Karakus et al. 2019, Lehrenfeld and Schöberl 2016), further studies would be necessary to rigorously justify this approach according to the study by Riccius (2019).*

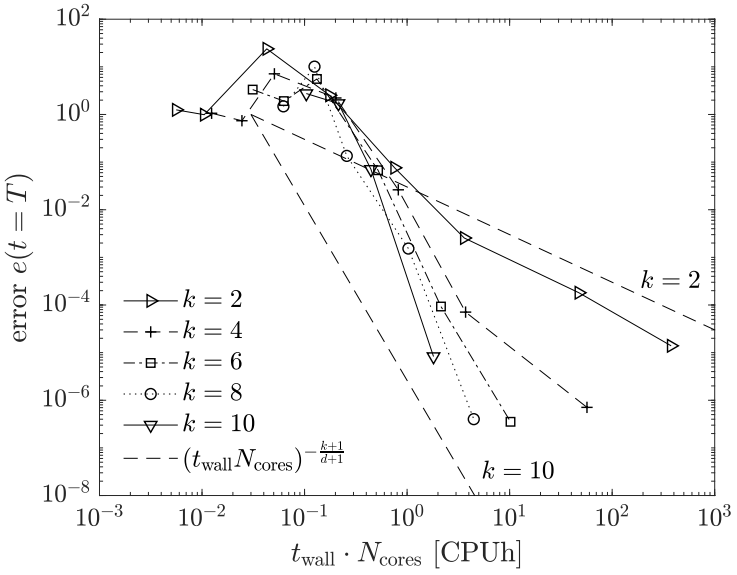


Figure 6.8: Orr–Sommerfeld problem: efficiency in terms of error vs. computational costs for coupled solver with explicit treatment of convective term and time step sizes calculated according to the global CFL condition.

6.3.3 Orr–Sommerfeld problem

This section investigates the efficiency for the two-dimensional Orr–Sommerfeld problem described in Section 2.6.7.1. The coupled solution approach is used (BDF2 scheme, explicit treatment of convective term, time step size calculated according to the global CFL condition (2.209) with $\mathrm{Cr} = 0.2$), where the coupled system of equations is solved by a flexible GMRES solver with block-triangular preconditioner, inverse mass matrix preconditioner for the velocity block, and Cahouet–Chabard preconditioner for the Schur complement block, where the Laplace operator is inverted approximately by one V-cycle of a cph-MG method with Chebyshev(5) smoother. The penalty step is preconditioned by the inverse mass matrix. This setup ensures robust iteration counts and a computationally efficient solution of linear systems of equations. The number of iterations for the coupled system of equations is not larger than 17, and for the postprocessing step with the divergence and continuity penalty terms not larger than 13 for all polynomial degrees and refinement levels. In particular, iteration counts do not increase significantly for higher polynomial degrees. The simulations have been performed on an Intel Haswell architecture and for finer meshes the number of cores is chosen such that simulations are performed away from the strong-scaling limit.

Figure 6.8 details the overall efficiency of the high-order DG solver for the Orr–Sommerfeld problem. Dashed lines indicate theoretically optimal complexity according to the analytical efficiency models from Section 6.2.4. For high spatial resolutions, the obtained results seem to agree well with these theoretical estimates, and improvements in efficiency by orders of magnitude can be observed for high-order methods for low error levels of 10^{-6} to 10^{-4}. High-order methods are also clearly more efficient for an engineering accuracy with errors around 10^{-2}. Towards low refinement levels, substantial deviations from the theoretical behavior can be observed

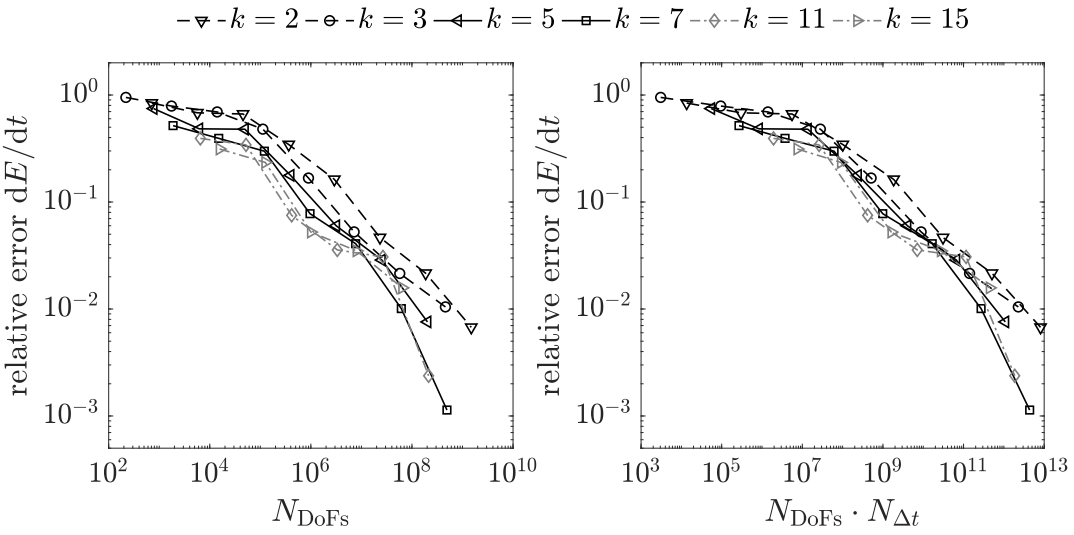
and improvements in efficiency for high-order methods diminish for error levels of 10^{-1} to 10^{0}. On the one hand, this is due to the sub-optimal convergence behavior of high-order methods in the pre-asymptotic regime, see Figure 2.18. On the other hand, another main factor explaining the difference between the error-vs-unknowns curves in Figure 2.18 and error-vs-costs curves in Figure 6.8 is the throughput, which increases by about two orders of magnitude from low to high refinement levels for this example (characterized by a trivial geometry and a coarse mesh consisting of only one element, similar to the two-dimensional vortex problem). Put differently, simulations on low refinement levels are slowed down by up to two orders of magnitude as compared to an imaginary problem where the coarsest mesh consists of a number of elements large enough to saturate a core (where one might argue that this is the case for engineering problems with non-trivial geometry). Hence, care has to be taken when trying to generalize these results. Finally, high-order methods require more time steps for the same number of unknowns due to the CFL condition with exponent $k^{1.5}$, which further decreases the efficiency of high polynomial degrees shown in Figure 6.8 compared to Figure 2.18.

6.3.4 Three-dimensional Taylor–Green vortex

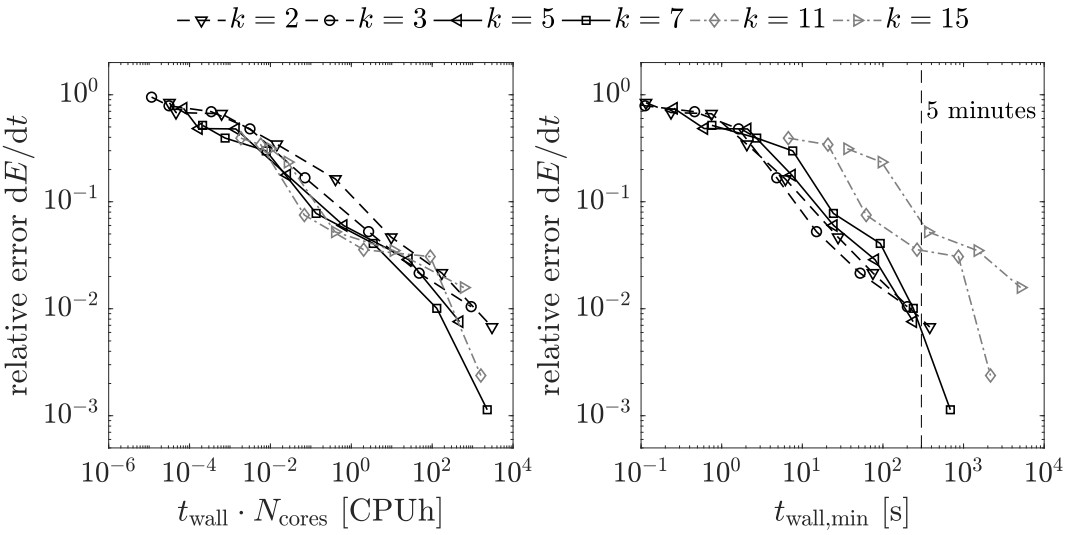
This section details the performance of the present incompressible Navier–Stokes DG solver for three-dimensional turbulent flow problems by the example of the Taylor–Green benchmark problem, see Section 2.6.7.2. For the results shown here, the dual splitting scheme is used due to its computational efficiency, using adaptive time stepping and a constant Courant number of $\mathrm{Cr} = 0.4$ for all simulations. The preconditioners used for this example are described in Section 5.6. The efficiency of high-order methods is investigated in detail for this problem, the node-level performance is considered with a comparison to state-of-the-art high-order solvers from the literature, and the parallel scalability of the solver on large supercomputers is investigated. Simulations have been performed on the Intel Skylake system specified in Table 4.2 and are run in a saturated regime with sufficient workload per core unless specified otherwise (for small problem sizes, simulations are run in serial or only on a partially-loaded node in order to obtain a more realistic estimate of computational costs).

6.3.4.1 Efficiency of high-order methods

Figure 6.9 analyzes the efficiency of high-order discretization schemes for the Taylor–Green vortex problem according to the proposed efficiency model. Mesh convergence studies are performed for polynomial degrees $k = 2, 3, 5, 7, 11, 15$. The efficiency of the spatial and temporal discretization schemes is analyzed in Figure 6.9 (a), while the overall efficiency in terms of accuracy versus computational costs is analyzed in Figure 6.9 (b). Recalling the spatial convergence results from Section 2.6.7.2, a loss of optimal rates of convergence is observed for this transitional/turbulent flow problem in the under-resolved regime. Nevertheless, high-order methods tend to be systematically more accurate than low-order methods for the same number of unknowns. The impact of time stepping is included in the right panel of Figure 6.9 (a). Taking into account the time step size, the curves for high polynomial degrees are shifted towards the curve of the low polynomial degree $k = 2$ in agreement with the theoretical expectation according to the $k^{1.5}$ relation in the CFL condition. Including the efficiency of iterative solvers and preconditioners (see Table 5.10) and the efficiency of the matrix-free evaluation (see Figure 4.10) leads



(a) efficiency of discretization



(b) overall efficiency in terms of error versus computational costs

Figure 6.9: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: investigation of efficiency of high-order methods for polynomial degrees $k = 2, 3, 5, 7, 11, 15$ using the dual splitting scheme (BDF2, $\mathrm{Cr} = 0.4$, $\varepsilon_{\mathrm{rel}} = 10^{-3}$). An accurate reference solution published in Fehn et al. (2018a) with effective resolution of 1024^3 for polynomial degree $k = 7$ is used to calculate the errors.

to the overall efficiency in Figure 6.9 (b). The best efficiency is obtained for moderately large polynomial degrees $k = 5, 7$, for which the efficiency is significantly improved as compared to the lower polynomial degrees $k = 2, 3$. Very high polynomial degrees $k = 11, 15$ are not more efficient than moderately large polynomial degrees $k = 5, 7$. Although the curves for different polynomial degrees appear to be very close to each other, the improvement in overall efficiency

is up to one order of magnitude for $k = 7$ as compared to $k = 2$. These results are in agreement with observations made in previous examples and can be summarized as follows. The efficiency of high-order methods with optimal-complexity implementation is mainly determined by the spatial convergence properties of the discretization scheme. Then, an analysis of the efficiency according to the model presented in Section 6.2 allows to identify three main components explaining a reduction in efficiency for high-order methods for the present DG solver when turning to the more relevant metric of accuracy versus computational costs: (i) restrictions of the time step size according to the CFL condition in case of an explicit treatment of the convective term reduce the efficiency of high-order methods, (ii) iteration counts for solving algebraic systems of equations increase slightly for high-order methods (see Table 5.10), (iii) the computational efficiency of the matrix-free evaluation of discretized operators reduces for very high-order methods in three space dimensions (see Figure 4.10). Put differently, high-order methods need to be significantly more accurate in order to achieve an advantage in terms of efficiency. Note also that the problem analyzed here is a viscous problem at moderate Reynolds number solved on a trivial geometry. Additional complexity in terms of complicated geometries (involving singularities), wall-bounded turbulent flows, higher Reynolds numbers or inviscid problems can be expected to render the efficient use of very large polynomial orders even more challenging, e.g. reducing the polynomial degree at which efficiency is optimal.

In the right panel of Figure 6.9 (b), an alternative definition of computational costs is used, namely the minimum wall-time obtained in the strong-scaling limit. The number of compute nodes is selected such that the number of SIMD-cells per core is approximately one (note that the $l = 8, k = 2$ simulation was run on 792 nodes and a larger number of processors would have been necessary to also reach the strong-scaling limit for this very fine mesh), the natural scaling limit of the matrix-free implementation in `deal.II` with vectorization over elements (and the domain decomposition in `deal.II` that does not assign fewer than 2^d elements to one MPI process), see also Figure 6.12 below. The same problem size in terms of refine level and polynomial degree is therefore run on a different number of cores in the left and right panels of Figure 6.9 (b). One can clearly see that lower polynomial degrees of $k = 2, 3$ perform best in this metric, while the minimum wall-time is significantly larger for high polynomial degrees. This is related to the aspect of parallel scalability and is explained in more detail in Section 6.3.4.3. It is remarkable that grid-converged results with relative errors in the kinetic energy dissipation rate around 1% can be obtained in wall-times of a few minutes for the 3D Taylor–Green vortex problem and the present high-order DG solver. The overall behavior is vastly different as compared to the left panel of Figure 6.9 (b), highlighting that the chosen metric has a substantial influence on answering the question "What is the optimal polynomial degree?". According to these results, polynomial degrees of $k = 2, 3$ can be expected to be the first for which the problem termed "LES crisis" (see Section 6.1.3) can be overcome in the future. Note again that the cost metric $t_{\text{wall,min}}$ assumes and requires unbounded compute resources. In practice, this implies that such jobs require a significant fraction of the compute resources of large supercomputers, for which the time to wait in the queue might be substantial (depending on the scheduling policy), and substantially larger than the actual execution time. Since these wall-times would actually have to be included in the wall-time metric (the results in Figure 6.9 (b) only show the execution time), it is clear that this cost metric also has major deficiencies and limitations.

6.3.4.2 Node-level performance and comparison to state-of-the-art

To quantify the efficiency of the present solver, the overall computational costs for the Taylor–Green vortex problem at $\mathrm{Re} = 1600$ (including setup costs like mesh generation and initialization of data structures as well as postprocessing costs, but excluding MPI initialization) are compared to results published recently in the literature for state-of-the-art high-order methods and implementations. Table 6.3 lists overall computational costs for various polynomial degrees. Compared to results published previously for the present high-order DG code, compare Table 6.3 to (Fehn et al. 2018a, Table 5), the results shown here are a factor of 4 to 5 faster. This speed-up mainly originates from a reduction in the number of time steps by a factor of approximately 3.5. This is realized by adaptive time stepping and a factor of $\zeta_{\mathrm{LF}} = 0.5$ for the Lax–Friedrichs flux increasing the critical Courant number by up to a factor of 2 compared to $\zeta_{\mathrm{LF}} = 1.0$ (note, however, that an increase in the time step size also causes an increase in iteration counts for the chosen inverse mass matrix preconditioner). Moreover, the solver tolerances are larger in the present work ($\varepsilon_{\mathrm{rel}} = 10^{-3}$ versus $\varepsilon_{\mathrm{rel}} = 10^{-6}$), a hybrid cph-MG method is used here compared to a pure h-MG method in Fehn et al. (2018a), and newer hardware is used in the present work (Intel Skylake vs. Intel Haswell).

In Table 6.4 (a) and (b), the efficiency of the present solver is compared to other CPU implementations in terms of computational costs $t_{\mathrm{wall}} \cdot N_{\mathrm{cores}}$. Compared to the high-order DG solver of the compressible Navier–Stokes equations by Carton de Wiart et al. (2014), the present approach is two to three orders of magnitude more efficient. The results published in Fehn et al. (2019c) suggest that this difference mainly stems from the efficiency of the high-order DG implementation, while the incompressible vs. compressible speed-up is much lower when using the same optimal-complexity implementation for both incompressible and compressible solvers. Compared to the high-order continuous finite element discretization of the incompressible Navier–Stokes equations in Huismann et al. (2019), a performance improvement of approximately one order of magnitude is achieved, which is mainly due to a reduced number of time steps required by the present solver. Both solvers achieve a similar throughput in terms of degrees of freedom solved per time step. Since the implementation in Huismann et al. (2019) uses special optimizations restricted to Cartesian geometries and since the present solver uses a computationally more complex DG discretization and a more general implementation also applicable to deformed hexahedral meshes, the performance of the present solver appears to be remarkable. The reported speed up can be considered conservative since the present DG solver at $k = 7$ is more accurate than the results reported in Huismann et al. (2019) for $k = 8$ on a mesh with the same number of elements (see Figure 6.10 for a quantitative assessment).

In Table 6.4 (c), the performance is compared to the PyFR solver (Loppi et al. 2018) based on the flux-reconstruction approach with a high-level platform-unified Python and performance-optimized implementation. The PyFR simulations were run on two Nvidia P100 GPUs and two Intel KNL accelerators. For this reason and since a one-to-one comparison for these different hardware systems is difficult, the problem has been simulated on both 1 and 2 Intel Skylake compute nodes with 48 cores each. The present solver outperforms the GPU implementation of PyFR run on Nvidia P100 by a factor of 2.9–5.7, and the CPU implementation run on Intel KNL by a factor of 7.4–14.3.

The present approach is also highly competitive to other discretization schemes. Compared to performance results published in DeBonis (2013) for finite difference methods where resolutions

Table 6.3: Taylor–Green vortex problem: performance results for dual splitting scheme for polynomial degrees $k = 2, 3, 5, 7, 11, 15$ and effective resolutions in the range 192^3 to 512^3. The time interval is $0 \leq t \leq T = 20$. Adaptive time stepping with a Courant number of $\mathrm{Cr} = 0.4$ is used for all computations.

k	l	resolution	N_{DoFs}	$N_{\Delta t}$	N_{cores}	$t_{\mathrm{wall}} \cdot N_{\mathrm{cores}}$ [CPUh]
2	6	192^3	$2.3 \cdot 10^7$	$1.3 \cdot 10^3$	48	$9.74 \cdot 10^0$
	7	384^3	$1.9 \cdot 10^8$	$2.7 \cdot 10^3$	384	$1.73 \cdot 10^2$
3	6	256^3	$5.7 \cdot 10^7$	$2.5 \cdot 10^3$	384	$4.77 \cdot 10^1$
	7	512^3	$4.6 \cdot 10^8$	$5.0 \cdot 10^3$	3072	$9.23 \cdot 10^2$
5	5	192^3	$2.5 \cdot 10^7$	$2.6 \cdot 10^3$	48	$2.63 \cdot 10^1$
	6	384^3	$2.0 \cdot 10^8$	$5.4 \cdot 10^3$	384	$4.74 \cdot 10^2$
7	5	256^3	$6.2 \cdot 10^7$	$4.4 \cdot 10^3$	48	$1.30 \cdot 10^2$
	6	512^3	$4.9 \cdot 10^8$	$8.9 \cdot 10^3$	384	$2.30 \cdot 10^3$
11	4	192^3	$2.7 \cdot 10^7$	$4.3 \cdot 10^3$	48	$8.98 \cdot 10^1$
	5	384^3	$2.1 \cdot 10^8$	$8.7 \cdot 10^3$	384	$1.60 \cdot 10^3$
15	4	256^3	$6.4 \cdot 10^7$	$6.9 \cdot 10^3$	96	$6.06 \cdot 10^2$
	5	512^3	$5.1 \cdot 10^8$	$1.4 \cdot 10^4$	768	$1.12 \cdot 10^4$

of 256^3 and 512^3 yield computational costs of $2.56 \cdot 10^3$ CPUh and $4.78 \cdot 10^4$ CPUh, respectively, the present solver reduces the computational costs by a factor of approximately 50 for $k = 3$ and a factor of approximately 20 for $k = 7$ (see Table 6.3).

For a finite volume discretization with a resolution of 64^3 and an end time of $T = 10$, a wall-time of 63 min is specified in Schranner et al. (2016) on an Intel SandyBridge system with 16 cores resulting in computational costs of approximately 17 CPUh. For the AVM^4 turbulence model (Rasthofer and Gravemeier 2013) based on a low-order continuous finite element discretization, computational costs of 29 CPUh have been measured on an Intel Haswell system for a resolution of 64^3 and an end time of $T = 20$. Selecting $l = 4$ and $k = 3$ (resolution 64^3) for the present discretization approach with $T = 20$ results in costs of 0.152 CPUh when simulated on one Skylake node with 48 cores, so that the present solver achieves an improvement in computational costs as compared to the finite volume approach by Schranner et al. (2016) and the finite element approach by Rasthofer and Gravemeier (2013) by a factor of approximately 200.

Of course, newer hardware is used in the present work that needs to be taken into account for a one-to-one comparison. However, the large differences in performance can not be explained solely by improvements in hardware, given that improvements in the single-core performance have been moderate over the last decade. As a conclusion, these numbers reveal that the use of efficient high-order DG methods allows to significantly improve the performance compared to many state-of-the-art numerical methods for turbulent flows.

As a summary of the computational efficiency, Figure 6.10 compares the present high-order DG solver for the most efficient degree $k = 7$ to state-of-the-art solvers from the literature.

Table 6.4: Three-dimensional Taylor–Green vortex at $\mathrm{Re} = 1600$: performance results for present solver (ExaDG) using the dual splitting scheme compared to state-of-the-art methods from the literature.

(a) comparison to compressible Navier–Stokes DG solver by Carton de Wiart et al. (2014) (where an Intel Xeon X5675 @3.07GHz architecture has been used) in terms of computational costs $t_{\mathrm{wall}} \cdot N_{\mathrm{cores}}$ (final time $T = 10$).

N_{eff}	k	N_{el}	solver	costs [CPUh]
256^3	3	64^3	compressible DG solver, Carton de Wiart et al. (2014)	$6.4 \cdot 10^3$
256^3	3	64^3	present solver (ExaDG)	$2.52 \cdot 10^1$
240^3	4	48^3	compressible DG solver, Carton de Wiart et al. (2014)	$1.08 \cdot 10^4$
240^3	4	48^3	present solver (ExaDG)	$2.53 \cdot 10^1$

(b) comparison to incompressible Navier–Stokes continuous spectral element solver solver by Huismann et al. (2019) (where an Intel Xeon E5-2590 v3 @2.5GHz has been used) in terms of computational costs $t_{\mathrm{wall}} \cdot N_{\mathrm{cores}}$ (final time $T = 20$).

N_{eff}	k	N_{el}	solver	costs [CPUh]
256^3	8	32^3	continuous FEM, Huismann et al. (2019)	$1.98 \cdot 10^3$
288^3	8	32^3	present solver (ExaDG)	$1.96 \cdot 10^2$

(c) comparison to high-order flux reconstruction solver by Loppi et al. (2018) in terms of overall wall-time t_{wall} using a comparable amount of hardware resources and considering both CPU and GPU hardware (final time $T = 20$).

N_{eff}	k	N_{el}	solver	t_{wall} [h]
260^3	4	52^3	PyFR, Loppi et al. (2018)	3.64 on 2x Nvidia P100
260^3	4	52^3	PyFR, Loppi et al. (2018)	9.12 on 2x Intel KNL
260^3	4	52^3	present solver (ExaDG)	1.24 on 1x Intel Skylake
260^3	4	52^3	present solver (ExaDG)	0.64 on 2x Intel Skylake

The methods chosen for comparison are two open-source CFD solvers, namely the CFD solver OpenFOAM [4] based on low-order finite volume methods, and the high-order DG code FLEXI [5] for the compressible Navier–Stokes equations for degree $k = 7$ (the three data points correspond to the same mesh resolution but using three different dealiasing techniques, namely filtering, split-form DG, and over-integration from left to right). The simulation results for OpenFOAM have been obtained in Wang (2019), where the results for the laminar model are used here, but the results are comparable in accuracy when using the Smagorinsky or WALE sub-grid scale models. The parameters for OpenFOAM have been investigated and chosen carefully with the aim to allow a fair comparison, and the reader is referred to Wang (2019) for detailed information on the chosen setup. A comparison is also made against results published in Huismann

[4] see https://openfoam.org/
[5] see https://www.flexi-project.org/

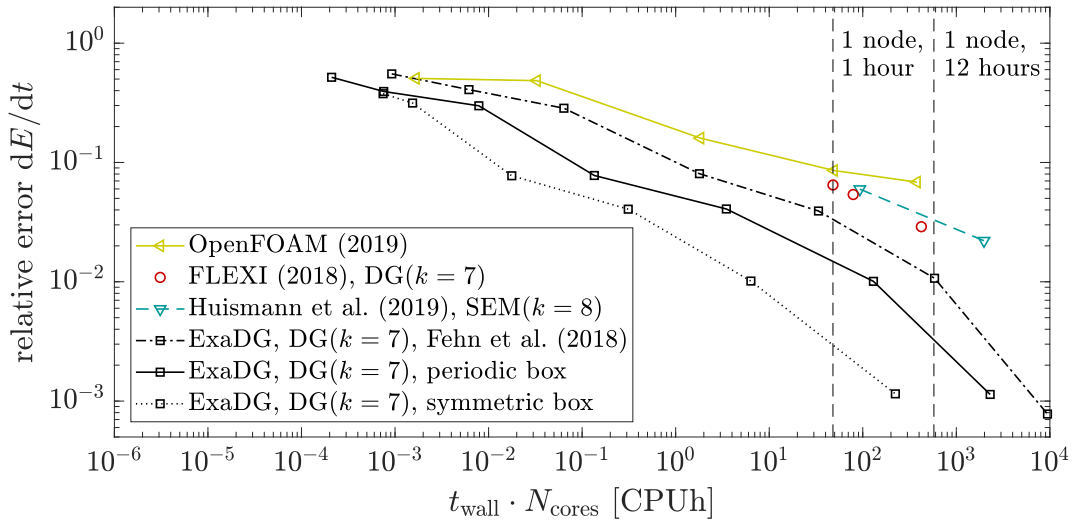


Figure 6.10: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: comparison of computational efficiency for polynomial degree $k = 7$ to state-of-the-art solvers from the literature.

et al. (2019) for degree $k = 8$ using a continuous spectral element solver for the incompressible Navier–Stokes equations. Also shown are previous results obtained with the present DG solver and published in Fehn et al. (2018a), where the performance optimizations described above have not been used. These reference results have all been performed on comparable Intel Haswell architectures (while the present results have been run on an Intel Skylake architecture). The present solver is clearly most efficient overall, with orders of magnitude improvements in overall efficiency. In Fehn et al. (2018a), a reduction in computational costs by a factor of at least 10 to 100 has been reported for a given level of accuracy compared to previous results for various CFD solvers shown in (Wang et al. 2013, Figure 25). Therefore, the present work only compares against most recent results from the literature. The symmetry of the Taylor–Green vortex can be exploited by simulating the problem only on 1/8 of the domain with symmetry boundary conditions (the setup used in Chapter 7 and denoted as symmetric box as compared to the periodic box used most often), which can be expected to yield a performance gain by another factor of 8. Of course, similar optimizations could be implemented or used for the other solvers as well to achieve a similar speed-up. Figure 6.10 plots two vertical lines to illustrate the amount of computational costs required to obtain grid-converged results with errors in the range 10^{-3} to 10^{-2} for the present solver. Using one compute node with 48 cores, grid converged results can be obtained in wall-time limits of "over lunch" (1 hour) or "over night" (12 hours). To the best of the author's knowledge, this is clearly the most efficient solution of this benchmark problem shown to date for a general-purpose PDE solver.

6.3.4.3 Parallel scalability

This section details the results of parallel scaling studies performed on the SuperMUC-NG supercomputer (see Table 4.2), where scaling studies up to the full machine with approximately 300k cores have been performed. The considered test case is the three-dimensional Taylor–Green

vortex problem at $\mathrm{Re} = 1600$ as described in Section 2.6.7.2 (note that an h-MG method is used for the pressure Poisson equation for these scaling studies since the cph-MG has not been implemented at that time). The periodic box is considered here since this is the standard setup used most often in the literature. Mainly a polynomial degree of $k = 3$ is chosen as the "working horse" high-order DG solver for incompressible turbulent flows. Further investigations run on a smaller scale show the parallel scalability of high-order methods compared to lower-order methods.

Strong scalability is here exclusively analyzed in terms of absolute runs times for the whole application (including setup costs like mesh generation and initialization of data structures as well as postprocessing costs, but excluding MPI initialization). This follows the credo that the aim of strong scalability is not parallel speed-up, but a minimization of the time-to-solution metric. Figure 6.11 shows results of a parallel scalability study for polynomial degree $k = 3$ and effective resolutions of $128^3, 256^3, 512^3, 1024^3, 2048^3$. The results of this scaling study have already been published in Arndt et al. (2020b). The figure shows both strong and weak scaling, where the weak scaling curves (which can be obtained by combining points of different strong-scaling curves horizontally in the right panel of the figure) are somewhat interrupted since one island of the machine contains 792 nodes (which is not a power of two). Figure 6.11 shows both the overall wall-time as well as the wall-time per time step. The minimum overall wall-time achieved in the strong-scaling limit increases with increasing spatial resolution. This is due to the fact that a solver with an explicit treatment of the convective term is used and that the time step size is restricted according to the CFL condition, $\Delta t \sim h$, so that the number of time steps increases by a factor of two when doubling the resolution (note that the time step size would, of course, also be reduced for implicit solvers when refining the spatial resolution). The minimum wall-time per time step reached in the strong-scaling limit is almost independent of the spatial resolution, which can be expected theoretically under certain optimality assumptions but is, nevertheless, a remarkable result when observed in practice.

To give a point of reference, the "realtime limit" is also plotted. Realtime refers to the fact that the physical time interval of the Taylor–Green vortex problem is $t = 20$ s with the parameters of the flow problem specified in SI units, i.e., a numerical simulation requiring a wall-time of $t_{\text{wall}} \leq 20$ s is able to simulate the problem in "realtime". Figure 6.11 reveals that the present solver is able to solve the three-dimensional Taylor–Green vortex problem in realtime for effective resolutions up to 128^3 for degree $k = 3$, a result that can be considered outstanding when compared to state-of-the-art high-order DG solvers.

The results also demonstrate that parallel scalability can be achieved up to the full machine with hundreds of thousands of CPU cores. A maximum speed-up factor of 424 is achieved for the intermediate 512^3 resolution, where the 1-node simulation has been run on a fat memory node (a manifestation of the memory limit in Figure 6.1). The 2048^3 resolution is too large a problem size to reach the strong-scaling limit on SuperMUC-NG with 300k cores. A parallel efficiency of 80.6% is achieved with a speed-up factor of 79.8 when scaling from 3072 to 304,128 cores. Note that being able to perform scalability studies for a given problem size over such a broad range of CPU cores is a result of a memory efficient software design that allows great flexibility in terms of problem sizes being solved on a given system (the strong-scaling limit and memory limit from Figure 6.1 are well-separated).

The results in Figure 6.11 reveal that scale resolving simulations for increasingly large Reynolds numbers (requiring increasing problem sizes and an increasing number of time steps) can

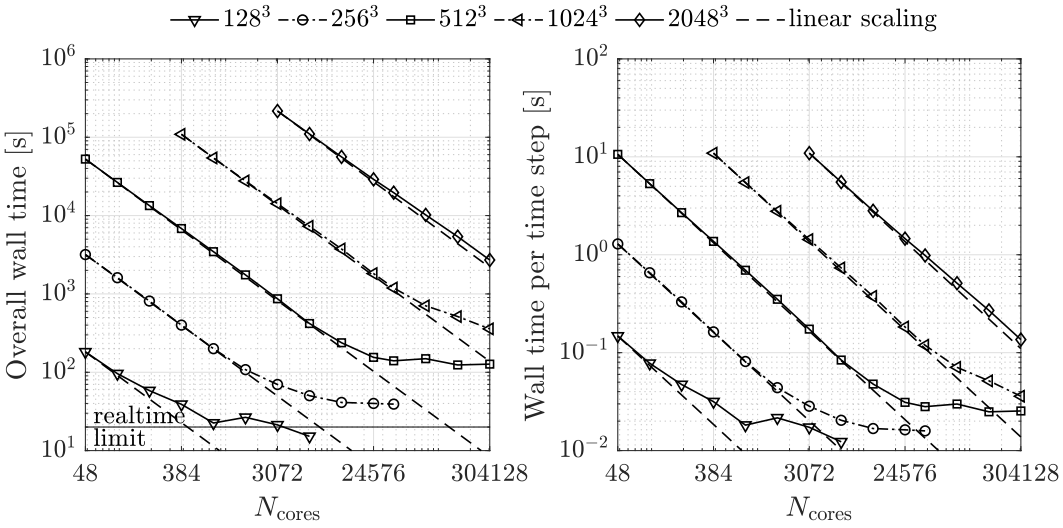


Figure 6.11: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: combined strong and weak scaling study for polynomial degree $k = 3$ and various effective resolutions from 128^3 to 2048^3.

not be realized by just using larger supercomputers (weak scaling) if not substantial progress is also made in reducing the wall-times in the strong-scaling limit, see the discussion in Section 6.1.3. To address the "LES crisis" problem described above, the minimum wall-time per time step reached in the strong-scaling limit allows extrapolations regarding how many time steps can be solved within a given wall-time limit. This can be considered a typical use case for industrial large-eddy simulation and direct numerical simulation of turbulent flows. A wall-time of 0.03 s per time step as shown in Figure 6.11 would allow to solve almost 3 million time steps per day of wall-time. Note, however, that these scaling studies have been performed for a structured grid and a Chebyshev coarse-grid solver, while complex geometries can be expected to require an AMG coarse-grid solver (see Chapter 5) that might have a strong impact on the strong-scaling limit (an aspect not addressed in the present work).

When increasing the polynomial degree k, scalability is reduced due to a coarser granularity. This is shown in Figure 6.12, where parallel scalability is compared for the degrees $k = 3, 7, 15$. In the left panel, strong scalability is shown for the 256^3 resolution. The minimum wall-time reached for high polynomial degrees is significantly larger than for lower degrees when considering the same effective resolution, which is a clear performance disadvantage of high-order methods that is not reflected in the computational costs metric when running simulations away from the strong-scaling limit. A question of practical interest is to quantify the strong-scaling limit in terms of how many unknowns or degrees of freedom should be used per core to make sure that the code is operating away from the strong-scaling limit. This is addressed in the right panel of Figure 6.12, which shows the wall-time per time step as a function of the number of SIMD-cells per core (see Section 4.4.4.3), i.e., the workload per core. Since the curves for different problem sizes overlap almost perfectly for sufficiently large workload per core, weak scalability of the overall turbulent flow solver is demonstrated. Note that this analysis does not only demonstrate weak scalability of the matrix-free operator evaluation, but also optimal al-

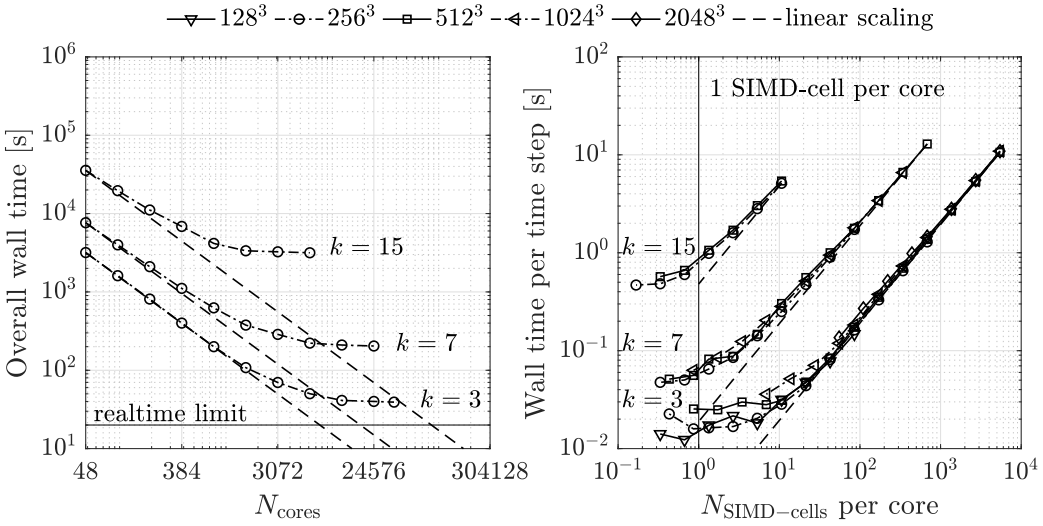


Figure 6.12: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: scalability for varying polynomial degree k for effective resolution of 256^3 in left panel, and wall-time per time step versus workload per core in right panel.

gorithmic complexity of solvers and preconditioners resulting in mesh-independent convergence rates. Furthermore, this analysis includes the strong-scaling limit and reveals how many cells per core should be used before approaching the strong-scaling limit with the theoretical scalability limit being one SIMD-cell per core. The wall-time levels off in the range of 1–10 SIMD-cells per core for $k = 3$, and scalability down to 1–2 SIMD-cells per core is observed for the higher polynomial degrees of $k = 7, 15$. Since high polynomial degrees involve more work for the same number of elements, the wall-time per time step increases with increasing polynomial degree. The range of workloads investigated for $k = 15$ is smaller than for the lower degrees. This is a manifestation of the "balanced regime" in Figure 6.1, where the workload is limited by the requirement that the simulation completes within a wall-time limit. Hence, for more complex problems requiring more time steps than the TGV problem considered here, the large polynomial degree $k = 15$ will be affected more strongly by the "LES crisis" problem in the sense that the maximum number of time steps (and hence the problem size) is limited more severely than for lower polynomial degrees in order for the simulation to stay within a wall-time limit. This is an aspect where the present matrix-free implementation could be improved, e.g. by changing the vectorization strategy (from vectorization over elements to vectorization within elements) and the mesh partitioning algorithm used by `deal.II`. This would imply a finer granularity and promise improved parallel scalability. Of course, this needs careful investigations given that it is unclear to which extent the node-level performance will deteriorate.

Remark 6.10 *Figures 6.11 and 6.12 present parallel scalability results in different ways. Intentionally, none of these plots shows normalized speed-up factors as this would remove important information from the plots. While improved parallel scalability is often reported in the literature for higher degrees and the same number of elements in terms of normalized speed-up factors (Altmann et al. 2013, Hindenlang et al. 2012), it would be a wrong conclusion to inter-*

pret such results as an advantage of high-order methods. The aim of using high-order methods is to obtain more accurate results for the same problem size (degrees of freedom, not elements), or to reduce the problem size for which a certain error level can be realized. In such a setting, the mesh consists of significantly less elements for the high-order variant, which makes it more challenging to reach a desired minimum wall-time in the strong-scaling limit for conventional parallelization strategies. A recent study by Houba et al. (2019) advertises high-order methods for reasons of better parallel scalability, but the aspect of computational costs remains elusive given that only normalized speed-up factors are shown.

6.4 Conclusion and outlook

This chapter has addressed the interdisciplinary topic of the computational efficiency of PDE solvers. The theoretical scaling of the problem size and the number of time steps with the Reynolds number for accurate simulations of incompressible turbulent flows emphasizes the need for computationally efficient techniques. Defining computational efficiency as the ratio of accuracy and computational costs, an evaluation of the efficiency of high-order discretization methods should distinguish between smooth or non-smooth problems on the one hand, and matrix-free or matrix-based implementation techniques on the other hand. The importance of parallel scalability has been discussed as well as the role of different hardware characteristics as limiting resources for PDE solvers. Further, possible definitions of computational costs have been discussed in detail, where the metric CPUh is the most common choice that is also mainly considered in this work. Regarding these theoretical considerations, a number of remarks shed light on different perspectives and highlighted common pitfalls. The complexity related to the interdisciplinary nature of the topics discussed in this chapter might explain why some of these topics often cause heated discussions in the CFD community. For this reason, an attempt has been made in providing a methodology according to which numerical methods for PDE solvers can be analyzed more systematically in terms of computational efficiency. A very simple efficiency model has been proposed that identifies the temporal and spatial discretization, iterative solvers and preconditioners, and the implementation as the main contributors to efficiency. Under certain optimality assumptions and an optimal selection of the time step size, analytical efficiency models have been derived that describe the error-vs-costs relation as a function of the polynomial degree for both smooth problems and those operating in the pre-asymptotic regime of convergence typically encountered when dealing with the simulation of turbulent flows.

Detailed numerical investigations have been presented for a set of examples spanning the range from smooth to non-smooth problems as well as from computationally cheap two-dimensional examples to computationally more demanding three-dimensional transitional and turbulent flow problems. For these examples, the efficiency of high-order methods compared to low-order methods has been assessed critically. The results have been verified by the use of analytical efficiency models. The results demonstrate that high-order methods are superior in the asymptotic regime. For more complex problems involving geometric complexities or operating in the regime of under-resolved turbulence, the asymptotic convergence behavior of high-order methods is lost. Across the range of examples considered here, it becomes clear that the spatial convergence behavior might be considered the key factor that decides about the efficiency of high-order methods. According to the efficiency model, the error-vs-DoFs relation then trans-

lates into the error-vs-costs relation by taking into account additional factors that tend to increase the computational costs for high-order methods compared to low-order methods. These factors are an increased number of time steps, a slight increase in iteration counts, and a reduction in the throughput of matrix-free operator evaluation in three space dimensions for high polynomial degrees. Aspects of parallel scalability further reduce the efficiency of high-order methods for the current implementation. A detailed investigation of the efficiency for the 3D TGV problem leads to the conclusion that a clear accuracy advantage of high-order methods appears to be necessary in order for high-order methods to be also more efficient overall. Despite the fact that algorithms and implementations with the best-known complexity are used, the efficiency tends to stagnate or even deteriorate for very high-order methods. Overall, polynomial degrees in the range $k = 5, \ldots, 7$ are found to be most efficient, and it is expected that the optimal degree will further decrease in case of geometrically complex problems or problems with very high Reynolds numbers. If the wall-time is used as an alternative cost metric, the present results indicate that lower degrees of $k = 2, 3$ perform best since these polynomial degrees show improved parallel scalability. It should be noted, though, that all polynomial degrees $k \geq 2$ considered here are denoted as high-order methods in the CFD community (Wang et al. 2013). In terms of absolute performance numbers, excellent node-level performance is demonstrated, significantly outperforming state-of-the-art approaches from the literature. Parallel scalability results are shown with excellent scalability down to the granularity limit of the present matrix-free implementation with vectorization over elements. In the strong-scaling limit, the 128^3 resolution for degree $k = 3$ can be simulated in realtime for the 3D TGV problem, and grid-converged results with relative errors in the kinetic energy dissipation rate of approximately 1% for the $\mathrm{Re} = 1600$ case are obtained in wall-times of approximately 5 minutes. These results demonstrate the HPC capability of the proposed incompressible Navier–Stokes DG solver and render this method highly attractive for future studies.

As part of future work, an in-depth investigation of the parallel scalability and the strong-scaling limit with a focus on the present hybrid multigrid methods could be addressed, in particular the role of AMG coarse-grid solvers that can be expected to be necessary for complex engineering applications. Alternative vectorization strategies of the matrix-free implementation could prove more efficient overall for very high polynomial degrees due to better parallel scalability, given that the serial performance might not necessarily become worse for vectorization within elements due to smaller working sets with less pressure on caches according to the results in Chapter 4. Moreover, an interesting research direction appears to be the question whether and to which extent fully implicit formulations can be realized with matrix-free algorithms of improved or optimal computational complexity, and whether these implicit solvers prove more efficient overall. First steps in this direction have been made in Bastian et al. (2019), Pazner and Persson (2018). Implicit matrix-based solvers with a complexity of at least $\mathcal{O}(k^{2d})$ are widely used in the literature for high-order DG discretizations of the incompressible Navier–Stokes equations, see Franciolini et al. (2017, 2020), but their efficiency relative to the matrix-free solvers presented here is unclear. The results in this chapter have shown that computational costs might easily vary by one or two orders of magnitude between different realizations of similar high-order flow solvers. This implies that conclusions of comparative studies w.r.t. different formulations of a method (for example implicit vs. semi-implicit vs. explicit) have to be taken with care if competitiveness to leading solvers has not been demonstrated. For example, the conclusions drawn in Yan et al. (2021) for implicit vs. explicit compressible DG solvers (favoring

implicit schemes) might be somewhat different if the most efficient explicit solver was used for comparison, e.g., compare the results in Yan et al. (2021) for the 3D TGV with the much faster explicit simulations shown in Fehn et al. (2019c). One might argue that both implicit and explicit formulations would benefit similarily from a faster implementation, but this is not to be expected since the matrix-based implicit solver is memory-bound so that there is typically only minor potential for performance optimizations. This points to another important aspect: According to the author's opinion, significant progress in the field of high-order DG discretizations of the incompressible Navier–Stokes equations could be achieved by cross-comparisons between different methods and implementations in terms of benchmarking (error-vs-costs analysis) in the spirit of Wang et al. (2013), where mainly DG discretizations of the compressible Navier–Stokes equations have been considered and where a comparison of implicit vs. explicit formulations or matrix-based vs. matrix-free formulations has not been of primary interest. Consider the multitude of DG approaches for incompressible flows (the HDG solver presented in Lehrenfeld and Schöberl (2016), the fully-explicit DG solver presented in Fu (2019), the staggered semi-implicit and space-time DG solver presented in Fambri and Dumbser (2016), the DG-Fourier solver presented in Ferrer (2017), the SIMPLE-based un-stabilized DG solver presented in Klein et al. (2015), the stabilized DG solvers presented in Fehn et al. (2018a,b) and Piatkowski et al. (2018), the hp-adaptive, unstabilized DG solver presented in Chalmers et al. (2019), the combined CG-DG solver presented in Gao et al. (2017), and the implicit, matrix-based, artificial-compressibility-flux DG solver presented in Franciolini et al. (2017, 2020)), which have been developed independently and for which often only limited information is publicly available regarding their computational efficiency.

7 Anomalous energy dissipation in incompressible Euler flows

Turbulence has fascinated researchers over decades, and still, many fundamental questions are unanswered. Two of these questions are: Is it possible that the incompressible Euler equations develop singularities in three space dimensions in finite time? Do turbulent flows dissipate kinetic energy in the inviscid limit, or is the kinetic energy preserved if no viscous effects are present? There is empirical evidence that the dissipation rate does not tend to zero in the limit $\mathrm{Re} \to \infty$, which is known as the zeroth law of turbulence or energy dissipation anomaly. Onsager hypothesized that anomalous energy dissipation is linked to the occurrence of velocity singularities in the absence of viscosity. Numerical evidence is still outstanding.

This chapter addresses this topic by high-resolution numerical simulations of the inviscid three-dimensional Taylor–Green vortex problem. Many prerequisities are required for such a numerical study to be successful. First and foremost, a robust discretization scheme is required that remains stable in the inviscid limit and exhibits suitable mechanisms of dissipation, see Chapter 2. Next, advanced numerical methods and implementations are required that are able to efficiently solve algebraic systems of equations of around $\mathcal{O}(10^{12})$ unknowns and beyond. This requires optimal-complexity multigrid algorithms that scale linearly in the number of unknowns and an optimal-complexity implementation with a minimum of memory transfer and arithmetic operations, see Chapters 4 and 5. Finally, a massively parallel implementation is needed to conduct such a study on high-performance computing facilities, see Chapter 6. One aim of this chapter is to demonstrate that the methodology developed in this thesis can be successfully used to conduct numerical studies with an unprecedented level of resolution and accuracy, and to potentially gain new physical insights into the underlying mechanisms of turbulence. The content of this chapter has already been published in Fehn et al. (2021b).

In Section 7.1, a detailed introduction is given with an explanation of basic terminologies, a review of the state-of-the-art is provided, and a new technique to explore finite-time singularities and anomalous energy dissipation is proposed. Section 7.2 briefly describes the numerical methods used for this study and summarizes the quantities of interests. Section 7.3 shows results for the one-dimensional Burgers equation with formation of a shock, a two-dimensional shear layer problem that can be expected to be energy-conserving in the limit $\nu \to 0$, and finally the three-dimensional inviscid Taylor–Green vortex problem that has been suspected to exhibit finite-time singularities. In Section 7.4, the results of this study are summarized and conclusions are drawn based on the present observations.

7.1 Motivation

Singularities play a key role in fluid mechanics (Eggers 2018). While singularities in the form of shocks are well-understood for the compressible Euler equations and the inviscid Burgers equation (Burgers 1948) as a simplified model, the occurrence of singularities that develop in finite time is discussed controversially for the three-dimensional incompressible Euler equations. The occurrence of finite-time singularities is strongly related to anomalous dissipation of kinetic energy in three-dimensional incompressible Euler flows according to the pioneering work by Onsager (1949), which is well-documented in the review articles by Eyink (2008), Eyink and Sreenivasan (2006), and in the recent essay by Dubrulle (2019). Due to the relation between singularities and dissipation, one can distinguish between

(i) a *direct* approach to identify finite-time singularities for incompressible Euler flows, e.g., by showing that the vorticity blows up in finite time through different methods (e.g., an analysis of the kind $\|\boldsymbol{\omega}\|_\infty \sim (t_* - t)^{-\gamma}$ according to the Beale–Kato–Majda theorem (Beale et al. 1984) trying to identify t_* and γ from numerical results), and

(ii) an *indirect* approach proposed in the present work that provides indications of finite-time singularities by observing an "anomalous" dissipative behavior in the kinetic energy evolution.

While most approaches in the literature can be identified as belonging to the first category, the focus is on a new technique related to the second category in this chapter. To complement these results, additional numerical results related to the direct identification approach such as the temporal evolution of the maximum vorticity $\|\boldsymbol{\omega}\|_\infty$ and the enstrophy $\mathcal{E}$ are shown.

7.1.1 State-of-the-art and limitations in tracing finite-time singularities

A strategy to identify potential singularities directly are time series expansions shown in Brachet et al. (1983), Morf et al. (1980), Pelz and Gulak (1997), Taylor and Green (1937), but it was found that numerical inaccuracies prevent a definite answer when using this technique. Numerical investigations by means of PDE solvers have therefore played the most dominant role in the exploration of finite-time singularities.

A difficulty in identifying singularities with the direct approach by numerical simulations is the inherent conflict that arbitrarily small structures can not be resolved with a numerical simulation of finite resolution, which renders this problem one of the most challenging topics in computational fluid dynamics. Much work has been done in this field. In the 1980s and 1990s, several early works on direct numerical simulation of both inviscid and high-Reynolds-number viscous incompressible flows reported indications of finite-time singularities for the incompressible Euler equations (Boratav and Pelz 1994, Brachet et al. 1992, 1983, Kerr 1993, Kerr and Hussain 1989). Symmetry in the initial conditions plays an important role as specifically mentioned and addressed by some works (Boratav and Pelz 1994, Pelz and Gulak 1997, Pelz 2001), raising the question whether singularities are possible for problems that are not perfectly symmetric. However, these works do not allow a definite answer to the question of finite-time singularities, see also the review articles by Gibbon (2008), Hou and Li (2008) on this topic. One of the

main reasons why the results of these studies have been inconclusive is that the spatial resolution has been limited due to the computational power and computational approaches available at the time. Numerical results shown in Hou and Li (2008) suggest that dynamic depletion of vortex stretching could be a mechanism that prevents a finite-time blow-up, but the same authors report evidence for a finite-time singularity for a different flow configuration with solid boundaries in a later work by Luo and Hou (2014).

In terms of the flow configuration being studied, numerical investigations on finite-time singularities can be categorized as follows: The Taylor–Green vortex has been analyzed in Brachet et al. (1992, 1983), Brachet (1991), Bustamante and Brachet (2012), Cichowlas and Brachet (2005), Shu et al. (2005) and for a regularized problem considering the Euler–Voigt equations in Larios et al. (2018), the high-symmetry Kida–Pelz initial condition in Cichowlas and Brachet (2005), Grafke et al. (2008), Hou and Li (2008), colliding Lamb dipoles in Orlandi et al. (2012), and other perturbed cylindrical vortex tubes in Grauer et al. (1998), Hou and Li (2008), Kerr (1993, 2013), Kerr and Hussain (1989). Most studies used spectral methods as discretization schemes.

For the direct numerical simulations, common approaches to trace singularities are monitoring the maximum vorticity $\|\boldsymbol{\omega}\|_\infty$ over time, see the Beale–Kato–Majda theorem (Beale et al. 1984), and the "analyticity strip" method (Sulem et al. 1983), which aims at capturing the smallest scales of the flow. The width of the analyticity strip $\delta(t)$, obtained from fitting the energy spectrum to $E(k,t) = C(t)k^{-n(t)}\exp(-2k\delta(t))$, is monitored over time for successively finer spatial resolutions up to a resolution for which extrapolations of $\delta(t)$ allow conclusions whether $\delta(t)$ reaches 0 in a finite time (finite-time singularity) or whether $\delta(t)$ decreases only exponentially in time (regularity at all times).

Numerical results for the three-dimensional inviscid Taylor–Green vortex shown in Brachet et al. (1983), Cichowlas and Brachet (2005) indicate only an exponential decay, but this might be due to the limited spatial resolution and also due to the fact that only small times of the TGV flow have been considered, so that a finite-time singularity can not be excluded from these results. In a later work by Bustamante and Brachet (2012), a change in regime indicating potentially faster than exponential decay is reported and the results are "not inconsistent with the occurrence of a singularity", but again resolutions higher than the maximum one of 4096^3 would be required for definite answers. In Cichowlas and Brachet (2005) it is estimated that conclusions regarding finite-time singularities using the analyticity strip method would require spatial resolutions of $(16\mathrm{k})^3$ to $(32\mathrm{k})^3$ for the Kida–Pelz initial data. A recent study by Campolina and Mailybaev (2018) suggests that the resolution available via classical DNS is not sufficient to investigate blow-up.

The development of pancake-like structures with exponentially growing vorticity during the early development of turbulence from smooth initial data is studied in Agafontsev et al. (2015). In Kerr (2013), a new kind of analysis based on rescaled vorticity moments is proposed studying anti-parallel vortex tubes, and only double-exponential growth in vorticity is observed as opposed to the singular behavior suspected in a previous work by Kerr (1993). A model describing a cascade of transformations between vortex filaments and sheets potentially explaining the mechanism of singularity formation in the Euler equations is proposed in Brenner et al. (2016). Another model has been described recently in Moffatt (2019). In McKeown et al. (2018), an iterative cascade of instabilities for head-on collisions of vortex rings is investigated both experimentally and numerically.

7.1.2 Energy dissipation anomaly

The focus is now put on the evolution of kinetic energy in incompressible Euler flows. Of particular interest is the question whether inviscid flows are able to dissipate energy, and if so, by which mechanism such a behavior can be explained, given that no viscous effects are present. The kinetic energy dissipation equation – valid for incompressible viscous ($\nu > 0$) flows with continuously differentiable solution on a domain with periodic boundaries – reads (Eyink and Sreenivasan 2006, Onsager 1949)

$$\frac{\partial E(t,\nu)}{\partial t} = -\int_\Omega \nu \nabla \boldsymbol{u}^\nu : \nabla \boldsymbol{u}^\nu \,\mathrm{d}\Omega \,, \tag{7.1}$$

which implies conservation of energy in the inviscid limit $\nu = 0$ provided that the solution is sufficiently regular. However, from phenomenological descriptions of turbulence, there is empirical evidence that the dissipation rate does not tend to zero in the limit $\mathrm{Re} \to \infty$ or $\nu \to 0$ but takes a positive value independent of ν, which is known as dissipation anomaly or the zeroth law of turbulence (Dubrulle 2019, Eyink 2008). As noted in Eyink (2008), this has first been observed by Taylor (1935), and also Kolmogorov's similarity theory of turbulence (Kolmogorov 1991) is based on the assumption of a non-vanishing energy dissipation rate in the inviscid limit. Numerical evidence that the dissipation rate is independent of ν for large Re is for example given in Kaneda et al. (2003), Orlandi et al. (2012), Sreenivasan (1998), and experimental evidence for example in Dubrulle (2019), Pearson et al. (2002). Under certain regularity or smoothness assumptions (existence of a strong L^3-limit; see Drivas and Eyink (2019), Drivas and Nguyen (2019), Duchon and Robert (2000) for a precise discussion), weak Euler solutions are the $\nu \to 0$ limit of Leray–Hopf weak solutions $\boldsymbol{u}^\nu$ of the Navier–Stokes equations, so that the dissipation rate in the inviscid limit equals the viscous dissipation rate in the limit $\nu \to 0$

$$\frac{\partial E_{\nu=0}(t)}{\partial t} = \lim_{\nu\to 0} \frac{\partial E(t,\nu)}{\partial t} = \lim_{\nu\to 0} -\nu \int_\Omega \nabla \boldsymbol{u}^\nu : \nabla \boldsymbol{u}^\nu \,\mathrm{d}\Omega = -D(t) \leq 0 \,, \tag{7.2}$$

where anomalous energy dissipation means that $D(t) > 0$ for some (or all) $t > t_*$.[1] In general, weak Euler solutions may neither be unique nor the zero-viscosity-limit of weak Navier–Stokes solutions, which might themselves be non-unique (Buckmaster and Vicol 2019, 2020, Buckmaster et al. 2021, Daneri et al. 2021, Isett 2017). In this sense, note that the first equality in equation (7.2) is a conditional one. The reader is also referred to Brenier et al. (2011), Wiedemann (2017) regarding the topic of weak–strong uniqueness of Euler solutions.

The above argumentation already indicates that the theory explaining dissipation of energy in the absence of viscosity is related to the spatial regularity of the solution. According to Onsager (1949), energy dissipation in three-dimensional incompressible flows can take place in the absence of viscosity by the formation of singularities,[2] with the cascade from large to (arbitrarily)

[1] As a consequence, the enstrophy is inversely proportional to the viscosity for large Reynolds numbers in the case of anomalous dissipation with $D(t) > 0$.

[2] Hence, the mechanism explaining the occurrence of kinetic energy dissipation in the limit $\nu \to 0$ is that the velocity gradient might tend to infinity in this limit. Literally, Onsager (1949) wrote: "In the absence of viscosity, the standard proof of the conservation of energy does not apply, because the velocity field does not remain differentiable!". Interestingly, Onsager did not consider the energy-dissipating behavior of inviscid flows ("ideal turbulence") an anomalous behavior but rather a matter of fact.

small scales taking place in finite time, see also Eyink and Sreenivasan (2006). According to Onsager's conjecture, dissipation of energy may occur if the velocity is Hölder continuous with exponent $\leq 1/3$ (while Onsager's assertion says that energy is conserved for exponents $> 1/3$, see Cheskidov et al. (2008), Constantin et al. (1994), Duchon and Robert (2000), Eyink (1994) for proofs). For mathematical literature dealing with proofs of Onsager's conjecture, the reader is referred to Buckmaster et al. (2018, 2016), De Lellis and Székelyhidi Jr. (2013, 2014), Isett (2018) and references therein, where dissipative weak Euler solutions up to Onsager's critical regularity have been constructed using convex integration techniques. Newest insights from these works thus confirm that the Hölder exponent of $1/3$ is indeed the critical one in terms of energy dissipation. As noted in Dubrulle (2019), the original Kolmogorov cascade picture implies irregularities of the velocity field (at least locally) with Hölder exponent $\leq 1/3$, but it was Onsager who established the link between energy dissipation and irregularities of the velocity field for the Euler equations.

The above considerations might explain why this phenomenon is denoted as "kinetic energy dissipation anomaly", an alternative term used e.g. in Dubrulle (2019) is "inertial dissipation" (as opposed to viscous dissipation). The one-dimensional inviscid Burgers equation (Burgers 1948) with formation of a shock and the associated dissipation of energy serves as a prominent and well-understood example, and is for example discussed in Sulem et al. (1983) in the context of finite-time singularities and in Dubrulle (2019) in the context of inertial energy dissipation. In Josserand et al. (2020), the phenomenon of energy dissipation through finite-time singularities is illustrated for another one-dimensional model problem, the nonlinear Schrödinger equation. For the two-dimensional incompressible Euler equations, it is known that singularities can not develop in finite time from smooth initial data (Eyink and Sreenivasan 2006). This fundamentally different behavior is attributed to the vortex stretching term in the vorticity form of the Euler equations (Gibbon 2008)

$$\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} = (\boldsymbol{\omega} \cdot \nabla)\, \boldsymbol{u}\,, \tag{7.3}$$

where the vortex stretching term on the right-hand side vanishes in two dimensions since the vorticity $\boldsymbol{\omega}$ is perpendicular to the velocity $\boldsymbol{u}$ in that case. It can be expected that this mechanism makes up the nature of turbulence such as the energy transfer to small scales according to a turbulence cascade in three dimensions (Onsager 1949). Regarding three-dimensional turbulent flows, Onsager's conjecture appears to be widely accepted by now with the occurrence of singularities representing a building block of modern understandings of turbulence (Dubrulle 2019). Whether numerical methods for turbulent flows are able to reflect the phenomenon of anomalous dissipation in the inviscid limit is discussed below.

7.1.3 Interplay between physics and numerics

Having a closer look at numerical simulations of the inviscid Taylor–Green vortex, it can be observed that many of these simulations have been performed mainly for small times up to $t \approx 5$ (up to $t = 4$ in Brachet et al. (1992, 1983), Bustamante and Brachet (2012), Cichowlas and Brachet (2005) and up to $t \leq 6$ in Chapelier et al. (2012), Shu et al. (2005)), but not beyond the time at which finite-time singularities have been suspected, especially not up to the time at which the transition to a fully turbulent state takes place, with maximum kinetic energy dissipation rate at

time $t \approx 8 - 9$ (expected from high-Reynolds number viscous simulations (see Brachet et al. 1983)) and subsequent decaying turbulence. As mentioned by some of these works, one reason for this is that the results for a specific resolution are no longer reliable at later times once the flow becomes under-resolved (note that the spatial resolution is severely limited by computational resources even for large supercomputers available today). The key aspect, however, is that numerical simulations of the incompressible Euler equations are very challenging in terms of energy stability and numerical blow-up of the discretization scheme. Often, a lack of robustness of the numerical discretization scheme is reported for this challenging inviscid Taylor–Green vortex problem (see for example Chapelier et al. 2012, Winters et al. 2018).

The lack of understanding of what is to be expected in terms of kinetic energy dissipation from a physical perspective (energy dissipation anomaly discussed above) manifests itself in an uncertainty regarding the optimal design of discretization schemes from a numerical perspective. In Moura et al. (2017a,b), Piatkowski (2019), it is argued that a fundamentally different behavior in terms of energy dissipation and time reversibility is expected between viscous flows in the limit $\mathrm{Re} \to \infty$ and inviscid flows at $\mathrm{Re} = \infty$. Especially in numerical studies, it is often assumed that energy conservation holds for an exact solution of the Euler equations not only in two space dimensions but also in three space dimensions, see for example Bustamante and Brachet (2012), Chapelier et al. (2012), Grauer et al. (1998), Krais et al. (2020b), Schroeder (2019), Shu et al. (2005), Winters et al. (2018) and the recent review article by Coppola et al. (2019) to mention just a few. Numerical schemes that are exactly energy-conserving can indeed be constructed and have the advantage that non-linear blow-up of the numerical discretization scheme can be avoided in the challenging inviscid limit. For these reasons, energy-conserving schemes appear to be the current gold standard for the simulation of this type of problems in turbulence research. Inviscid TGV simulations performed in Schroeder (2019) using exactly divergence-free, energy-conserving discretization methods result in an exact conservation of energy, and the results are considered superior as compared to simulations with upwind fluxes that show a dissipative behavior.

However, the use of energy-conserving schemes is accompanied by a major limitation, namely that it excludes – by construction – the occurrence of anomalous energy dissipation. Onsager's conjecture dictates to rethink whether it is really a desirable quality criterion that a numerical method preserves the kinetic energy exactly in the inviscid limit $\nu = 0$. If anomalous dissipation occurs, there is an inconsistency between the physical dissipation behavior and the numerical dissipation behavior of energy-conserving discretization methods. Then, energy-conserving numerical methods would result in an $O(1)$ error in the case of inviscid flows with anomalous/inertial energy dissipation. Since no energy can leave the system, energy accumulates in small scales, a well-known phenomenon called thermalization. In terms of the kinetic energy spectrum, an energy-conserving numerical scheme can be expected to lead to an unphysical equipartitioning of energy when simulating beyond the time of the finite-time singularity (Orlandi 2009, Orlandi et al. 2012, Ray et al. 2011). The energy-conserving results of Schroeder (2019) indeed show such a behavior. One may conclude that the application of energy-conserving numerical methods is only reasonable for times $t < t_*$ before a potential singularity forms, since anomalous dissipation might occur afterwards. Further, one may formulate that a numerical scheme must contain mechanisms of dissipation as a minimal requirement in order to address the topic of anomalous energy dissipation. To describe discretization schemes suitable for investigating anomalous dissipation more precisely, it is considered a prerequisite to use consistent and stable discretization

schemes whose dissipation mechanisms are coupled to under-resolution effects in the numerical approximation of the solution, i.e., the dissipation mechanisms act on the finest resolved scales and shift to smaller scales under mesh refinement. As discussed in more detail below, it appears to be unclear mathematically whether such a scheme is able to find a dissipative weak solution of the Euler equations.

By studying Galerkin-truncated, energy-conserving simulations of the one-dimensional Burgers equation, the work by Ray et al. (2011) describes an interesting phenomenon, called "tyger phenomenon" in that work, where short-wavelength oscillations occur out-of-the-blue in the presence of singularities, that finally lead to thermalization. The importance of numerical dissipation to avoid the effect of thermalization for this one-dimensional Burgers problem is emphasized in a recent study by Murugan et al. (2020), which can therefore be seen in close analogy to the present work focusing on 3D Euler.

The situation is less complicated for two-dimensional Euler flows that are non dissipative. In that case, it can be expected that the kinetic energy dissipation rate converges to zero under mesh refinement for a consistent and energy-stable discretization scheme, and that there is per se no conflict with physics if an energy-conserving scheme is applied.

7.1.4 An indirect approach to identify finite-time singularities by energy considerations

The *indirect* approach to identify finite-time singularities relies on the physical intuition that the appearance of anomalous energy dissipation in free decay from smooth initial data requires a finite-time singularity. The basic idea is to capture the temporal evolution of the kinetic energy by a numerical method with appropriate inbuilt dissipation mechanisms as described in Section 7.1.3. If grid-convergence to a dissipative solution with non-zero kinetic energy dissipation rate can be demonstrated numerically, indirect evidence of a finite-time singularity is provided by the following line of arguments:

(i) Assume convergence of a sequence of numerical solutions to a dissipative weak Euler solution for $h \to 0$.

(ii) Weak–strong uniqueness holds for dissipative weak Euler solutions (Brenier et al. 2011, Lions 1996, Wiedemann 2017).

(iii) Supposing that an energy dissipation anomaly with non-zero kinetic energy dissipation rate is observed, it follows from items (i) and (ii) that a strong solution cannot exist but must have become singular.

The conclusion in item (iii) is based on an indirect proof. Assume that a strong solution exists. This strong solution is energy-conserving. By weak–strong uniqueness, the weak solution must be identical and, therefore, energy-conserving. By contradiction it follows that a strong solution cannot exist.

Note that assumption (i) could be weakened. Rather than (i), only convergence to a dissipative generalized weak solution in the sense of Lions (1996) or DiPerna and Majda (1987) is required. Interestingly, this weaker assumption might be provable for the limit $h \to 0$, as it has in fact

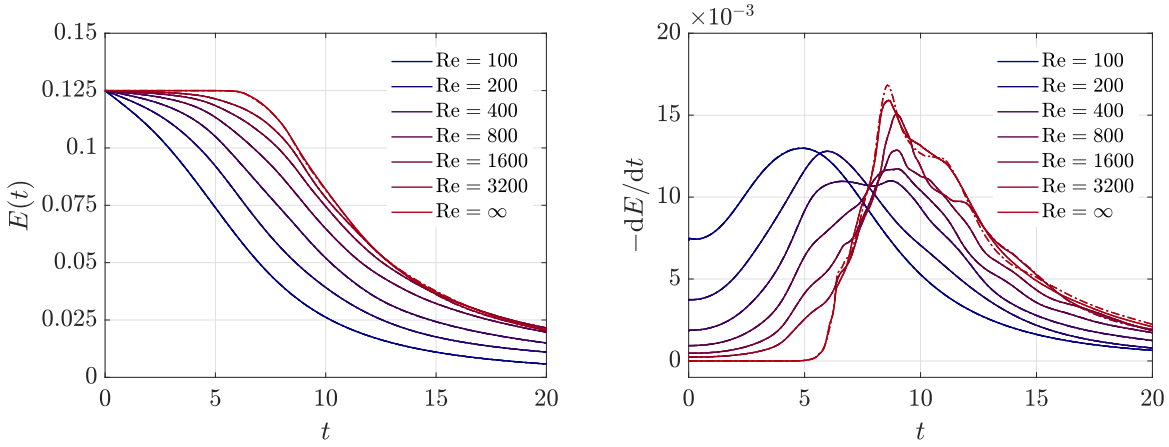


Figure 7.1: Temporal evolution of kinetic energy and kinetic energy dissipation rate for the three-dimensional Taylor–Green vortex problem for increasing Reynolds number of $\mathrm{Re} = 100, 200, 400, 800, 1600, 3200, \infty$. For each Re, results are shown for two mesh resolutions (fine mesh as solid line, coarse mesh as dashed-dotted line). The effective resolutions (see Section 7.3 for a definition) are $64^3, 128^3$ for $\mathrm{Re} = 100$, $128^3, 256^3$ for $\mathrm{Re} = 200, 400$, $256^3, 512^3$ for $\mathrm{Re} = 800$, $1024^3, 2048^3$ for $\mathrm{Re} = 1600$, $2048^3, 4096^3$ for $\mathrm{Re} = 3200$, and $4096^3, 8192^3$ for $\mathrm{Re} = \infty$. The results suggest that the kinetic energy reduces to a value as low as approximately 0.02 at time $t = 20$ for large Reynolds numbers, and that a similar amount of energy dissipation takes place also in the inviscid limit.

been proven for the limit $\nu \to 0$ (at least along sub-sequences). Since the existence of generalized weak Euler solutions as limits along sub-sequences relies on very general compactness arguments that require only L^2 (kinetic energy) bounds for both the Lions and DiPerna and Majda theories, a proof for the limit $h \to 0$ for a DG Euler discretization scheme as used here might be conceivable in analogy to what has been shown for the limit $\nu \to 0$. To the best of the author's knowledge, such a proof is still outstanding.

Note that a similar idea to identify singularities experimentally based on energy arguments has been used in Kuzzay et al. (2017), Saw et al. (2016) by calculating the inertial dissipation at scale l from PIV measurements. To the best of the author's knowledge, the present work makes first attempts in using energy arguments for singularity detection in numerical simulations of three-dimensional Euler flows.

From numerical simulations of viscous problems at finite Reynolds number, there are indications that the zeroth law of turbulence holds for the Taylor–Green vortex problem. Numerical results for the kinetic energy dissipation rate for increasing Reynolds numbers up to $\mathrm{Re} = 3000$ shown in Brachet et al. (1983), additional results for a higher Reynolds number of $\mathrm{Re} = 5000$ shown in Brachet (1991), $\mathrm{Re} = 10000$ in Arndt et al. (2020b), and $\mathrm{Re} = 20000$ in Lamballais et al. (2019) strongly suggest that the function $D(t)$ does not tend to zero in the limit $\nu \to 0$. This argument is summarized in Figure 7.1 showing results for viscous and inviscid simulations of the TGV obtained with the present discretization approach. For each Reynolds number, results are shown for two resolutions of the numerical discretization approach to judge whether the results are mesh-independent. Converged results are achieved for all finite Reynolds numbers shown

in Figure 7.1. In the inviscid limit, the temporal evolution of the kinetic energy is almost indistinguishable for the two finest resolutions, while small differences are still visible in the energy dissipation rate that slightly differs between the two resolutions 4096^3 and 8192^3 at later times around the dissipation maximum and beyond. However, the onset of dissipation around $t \approx 6$ appears to be converged also for this challenging inviscid simulation. These results are consistent with grid-convergence to a dissipative solution of the incompressible Euler equations for the three-dimensional Taylor–Green problem.

While the accumulation of energy in small scales in case of energy-preserving schemes is unphysical, it can be exploited under certain circumstances in order to gain insights into the physical dissipation behavior. In Cichowlas et al. (2005), an effective dissipation is estimated from the small-scale thermalized energy of energy-conserving, spectrally truncated Euler simulations and it is found that the large-scale Euler dynamics are similar to high-Reynolds number Navier–Stokes dynamics. Although the underlying numerical methods in that work are different from the present work,[3] there are interesting parallels. The onset of dissipation around $t = 5 - 6$ and the dissipation maximum around $t = 8 - 9$ appear to be very similar to the present results, so that both studies can be seen to complement and verify each other. From this perspective, the study by Cichowlas et al. (2005) supports the main conclusions of the present work.

Instead of considering a two-parameter limit problem $h \to 0, \nu \to 0$ as illustrated in Figure 7.1, this chapter focuses on the one-parameter limit $h \to 0$ for the inviscid limit $\nu = 0$. This decision can be explained as follows: A two-parameter study $h \to 0, \nu \to 0$ would technically not be realizable due to the large amount of computational costs required for such simulations. Already for moderate Reynolds numbers of $\mathrm{Re} = \mathcal{O}(10^4)$ as considered for example in Arndt et al. (2020b), Lamballais et al. (2019), the spatial resolutions required for grid-convergence are comparable to the highest resolution simulations realized in the present work for the inviscid limit. These highest resolution simulations require computational costs of tens of millions of CPUh, despite the fact that a highly efficient implementation (which is well-optimized for the hardware under consideration) is used in the present work. With the goal to realize spatial resolutions as high as those shown here, computational costs would allow to only consider a single finite Reynolds number beyond what is shown in Figure 7.1, explaining why this work immediately addresses the inviscid limit $\nu = 0$.

7.2 Numerical methods and analysis tools

This chapter seeks (weak) numerical solutions to the incompressible Euler equations solved on a domain $\Omega \subset \mathbb{R}^d$ in $d = 2, 3$ space dimensions. As introduced in Chapter 2, the Euler equations are obtained from the Navier–Stokes equations in the limit $\mathrm{Re} \to \infty$ or $\nu = 0$. The conservative formulation of the convective term is considered in this chapter

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) + \nabla p = \boldsymbol{0} \,, \tag{7.4}$$

$$\nabla \cdot \boldsymbol{u} = 0 \,. \tag{7.5}$$

[3] While the energy dissipation rate is derived by a postprocessing of results in Cichowlas et al. (2005), it is simulated directly in the present work.

Discretization in time is based on projection methods that solve for velocity and pressure unknowns in different sub-steps of a time step. In particular, the high-order dual splitting scheme in its second-order formulation $J = 2$ is used in this chapter, along with adaptive time stepping for reasons of computational efficiency. The reader is referred to Chapter 2.3 for an introduction and a discussion of temporal discretization schemes.

Discretization in space is based on high-order discontinuous Galerkin methods with suitable stabilization techniques that render the method robust for under-resolved, high-Reynolds-number flows. This chapter makes use of the DG discretization developed in Chapter 2.4. The conservative formulation of the convective term with Lax–Friedrichs flux and parameter $\zeta_{\mathrm{LF}} = \frac{1}{2}$ is used here. For viscous flow simulations, additionally shown in this chapter to complement inviscid results, the Laplace formulation of the viscous term is used. The velocity-pressure coupling terms are discretized in weak formulation. The divergence and continuity penalty terms are crucial for the robustness of the scheme when applied to turbulent flows in general, and in particular for the inviscid limit studied here, for which the spatial resolution will never be sufficient to resolve the smallest scales of the flow and for which the spatial discretization is always under-resolved. Default parameters of the penalty terms are used unless specified otherwise. In terms of the occurrence of finite-time singularities and anomalous energy dissipation, the present work makes use of the argument that – due to the weighted residual formulation – the present discretization can be applied to problems which lack regularity and for which the differential form of the equations is no longer an appropriate description.

The system of partial differential equations (7.4) and (7.5) does not extend to $d = 1$ in a meaningful way, since the incompressibility constraint $\partial u/\partial x = 0$ would imply $u = \text{const}$. However, the one-dimensional inviscid Burgers equation (Burgers 1948)

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\frac{u^2}{2} = 0 \,, \tag{7.6}$$

serves as a simplified mathematical model for more complex higher-dimensional problems. This model problem is used in this chapter in order to study the proposed methodology of identifying finite-time singularities by energy arguments for a well-understood one-dimensional problem, before applying this technique to two and three-dimensional problems. The spatial discretization of the inviscid Burgers equation is based on a discontinuous Galerkin scheme very similar to the one described above for the two- and three-dimensional case, i.e., the convective term is discretized with a local Lax–Friedrichs flux. Gaussian quadrature with a $3/2$-overintegration rule is used as in the higher-dimensional case due to the quadratic nonlinearity of the convective term, but intentionally no additional measures such as limiting, filtering, or other Riemann fluxes are taken to specifically address the occurrence of jumps that might form in the solution. It should also be emphasized that no artificial viscosity approach is used to deal with a potential singularity. For time integration, the classical explicit fourth-order Runge–Kutta method is used.

As a preparation for the numerical results shown below, important quantities of interest are summarized. Since turbulent flows in three space dimensions are the primary interest of this chapter, the following discussion is restricted to the three-dimensional case implying extensions of certain relations to one- and two-dimensional problems only where possible. Of primary importance for the present study is the temporal evolution of the kinetic energy

$$E(t) = \frac{1}{V_\Omega}\int_\Omega \frac{1}{2}\boldsymbol{u}(\boldsymbol{x},t)\cdot\boldsymbol{u}(\boldsymbol{x},t)\,\mathrm{d}\Omega \,, \tag{7.7}$$

and its dissipation rate $\mathrm{d}E/\mathrm{d}t$. The kinetic energy is normalized by the volume $V_\Omega = \int_\Omega 1 \mathrm{d}\Omega$ of the computational domain. The integrals are evaluated numerically by means of Gaussian quadrature with $k+1$ quadrature points in each coordinate direction. The time derivative used to obtain the dissipation rate is computed numerically from the kinetic energy at discrete instants of time via a second-order finite difference formula for variable time step sizes with first-order approximations at the end points. If anomalous dissipation ($\mathrm{d}E/\mathrm{d}t < 0$) occurs, the temporal evolution of the enstrophy $\mathcal{E}$,

$$\mathcal{E}(t) = \frac{1}{V_\Omega} \int_\Omega \frac{1}{2} \boldsymbol{\omega}(\boldsymbol{x},t) \cdot \boldsymbol{\omega}(\boldsymbol{x},t) \, \mathrm{d}\Omega \,, \tag{7.8}$$

is expected to exhibit a singularity $\mathcal{E} \to \infty$ in finite time. Integrals are computed by Gaussian quadrature, which is exact down to round-off errors due to polynomial integrands and Cartesian meshes. A related local quantity is the maximum vorticity

$$\|\boldsymbol{\omega}\|_\infty(t) \,, \tag{7.9}$$

where the maximum is taken over all quadrature points over all elements in the discrete case. The evolution of the maximum vorticity is monitored over time with the interest in a detection of potentially singular behavior in finite time. Although the maximum vorticity will remain finite for every numerical simulation of finite resolution, a mesh refinement study may give hints on the expected behavior if the resolution was further increased. Finally, kinetic energy spectra are considered by transformation into wavenumber space $\boldsymbol{k}$ (Bustamante and Brachet 2012, Cichowlas and Brachet 2005)

$$E(k,t) = \frac{1}{2} \lim_{\Delta k \to 0} \frac{\int_{\|\boldsymbol{k}\|=k}^{\|\boldsymbol{k}\|=k+\Delta k} \|\hat{\boldsymbol{u}}(\boldsymbol{k},t)\|^2 \mathrm{d}\boldsymbol{k}}{\Delta k} \overset{\text{DFT}}{\approx} \sum_{\substack{\boldsymbol{k}\in\mathbb{Z}^3 \\ k-\frac{1}{2} \le \|\boldsymbol{k}\| < k+\frac{1}{2}}} \frac{1}{2} \|\hat{\boldsymbol{u}}_{\mathrm{DFT}}(\boldsymbol{k},t)\|^2 \,, \tag{7.10}$$

where $\hat{\boldsymbol{u}}(\boldsymbol{k},t)$ denotes the Fourier transform of the velocity, which only exists at discrete wavenumber vectors $\boldsymbol{k}$ in case of a discrete Fourier transformation (DFT) obtained from sampled data of a discrete velocity field. The solution is first interpolated onto N equidistant points per element and per coordinate direction[4] to which the discrete Fourier transformation is then applied. For this purpose, the library `FFTW` is used in the present work (Frigo and Johnson 2005). Note that considering $E(k,t)$ as a function of a scalar wavenumber k as well as the summation over spheres of radius k introduces the assumptions of homogeneity and isotropy. In case of anomalous energy dissipation, the enstrophy is expected to become infinite. Hence, exploiting the relation $\mathcal{E}(k,t) = k^2 E(k,t)$ between the enstrophy and energy spectra and further assuming a power law behavior for the kinetic energy spectrum of the form $E(k,t) = C(t)k^{-n(t)}$, a singularity at time $t = t_*$ with $\mathcal{E}(t_*) = \int_0^\infty \mathcal{E}(k,t_*)\mathrm{d}k = \infty$ would correspond to a decay

[4]The number of sampling points is chosen as $N = k+1$ in the present work, i.e., equal to the number of nodal points of the discontinuous Galerkin discretization, where k is the polynomial degree of the shape functions and should not be mixed up with the wavenumber k typically used in the context of energy spectra. The equidistant interpolation points all lie within the element away from the boundaries where the solution is discontinuous. If interpolation points on the boundary are used, one typically takes the average of the solution from neighboring elements.

with slope $n(t_*) = 3$ in the energy spectrum, see also Orlandi (2009), Orlandi et al. (2012). This criterion is used as a further validation of the results in case other quantities give hints of a potentially singular behavior. Apart from that, energy spectra are typically investigated to assess the well-known $k^{-5/3}$ Kolmogorov spectrum for fully-developed, homogeneous isotropic turbulence. This property is used to investigate whether the numerical results match expected physical behavior obtained from classical cascade pictures in case of inviscid flows and beyond the time of potential singularities, $t > t_*$.

7.3 Numerical results

This section presents numerical results for three test cases in $d = 1, 2$, and 3 space dimensions. The one-dimensional problem is the well-known inviscid Burgers equation developing a singularity in finite-time for appropriate initial conditions. The two-dimensional example is a shear layer roll-up problem. The particular example used for the $d = 2$ investigations is not of primary importance as it is known theoretically that regularity is expected for two-dimensional Euler flows when starting from regular initial data. Instead, the aim of these one- and two-dimensional examples is an investigation to which extent the numerical discretization scheme is able to mimic physical behavior with potentially singular solutions. Having validated the numerical method for these well-understood problems, it is applied to the three-dimensional inviscid Taylor–Green problem, for which the physical understanding in terms of the occurrence of finite-time singularities and the related aspect of anomalous energy dissipation is speculative at present.

7.3.1 One-dimensional inviscid Burgers equation

As a simplified model for the incompressible Euler equations, the one-dimensional inviscid Burgers equation (7.6) is studied in a first step. It is well known that this equation develops singularities in finite time, see for example Dubrulle (2019), Sulem et al. (1983), and that this singularity is accompanied by the occurrence of energy dissipation. Previous works demonstrate that the use of energy-conserving discretization schemes leads to thermalization for this type of problems, see for example Murugan et al. (2020), Ray et al. (2011). Hence, it is particularly interesting to study the behavior of a DG discretization scheme for this simple 1D problem and investigate whether the numerical dissipation is able to predict the physical dissipation correctly. For the results shown below, a Courant number of $\mathrm{Cr} = 0.4$ is used for the explicit Runge–Kutta time integrator.

Figure 7.2 shows the numerical solution $u_h(x)$ at various instants of time and the formation of a shock. The problem is solved on the domain $\Omega = [-1, 1]$ with Dirichlet boundary conditions prescribed at both boundary points of the one-dimensional domain. Exemplarily, two different initial solutions are selected, a sine function, $u_h(x, t = 0) = -\sin(\pi x)$, and a hat function, $u_h(x, t = 0) = -2|x + 0.5| + 1$ for $x < 0$ and $u_h(x, t = 0) = 2|x - 0.5| - 1$ for $x \geq 0$. Due to the chosen initial conditions with $u > 0$ for $x < 0$ and vice versa, the solution piles up in the middle of the domain and a singularity ($\partial u/\partial x \to \infty$) forms at $x = 0$ in both cases. An equidistant grid with 2^l elements is used where l denotes the level of refinement. For polynomial approximations of degree k, the effective resolution becomes $(k + 1)2^l$. The oscillating behavior of the numerical solution around the singularity could be improved by more advanced

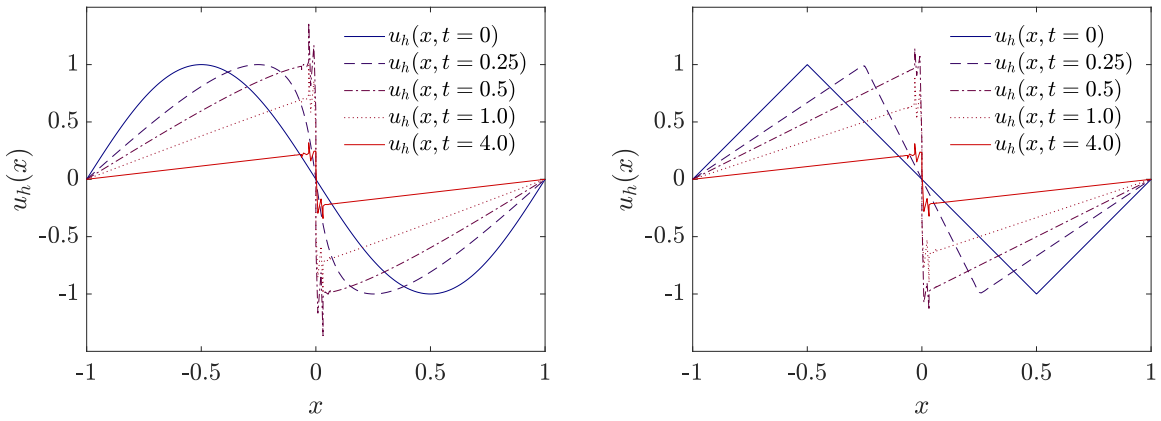


Figure 7.2: One-dimensional inviscid Burgers equations for two different initial solutions that form a singularity: On the left, the initial condition is a sine function, while it is a simple hat function that is piecewise linear on the right. The spatial resolution used for the computations corresponds to refinement level $l = 6$ and polynomial degree $k = 3$, resulting in an effective resolution of 256^1.

discretization techniques briefly mentioned in Section 7.2. From the results shown in Figure 7.2 it is plausible that the kinetic energy is conserved until the formation of the shock and that energy will be dissipated at later times. For the hat function chosen as initial condition, it is straightforward to derive an analytical expression for the temporal evolution of the kinetic energy as well as its dissipation rate, which is why this setup is considered in more detail in the following. According to the method of characteristics it follows that the shock forms at time $t_* = 0.5$. From that time on, the solution can be written as

$$u(x,t) = f(t)\left(x - \text{sign}(x)\right) \ , \tag{7.11}$$

where $\text{sign}(x)$ takes values of ± 1 depending on the sign of the argument. The temporal evolution part $f(t)$, which describes the absolute value of u taken to the left and right of the origin of the coordinate system at $x = 0$, can be obtained from the following consideration

$$f(t + \mathrm{d}t) = f(t) - \underbrace{\left.\frac{\partial u(x,t)}{\partial x}\right|_{x=0^-}}_{=f(t)} \underbrace{\mathrm{d}x}_{\substack{=u(x=0^-,t)\mathrm{d}t \\ =f(t)\mathrm{d}t}} = f(t)\left(1 - f(t)\mathrm{d}t\right) \ , \tag{7.12}$$

i.e., the solution at $x = 0^-$ at time $t + \mathrm{d}t$ equals the solution at position $-\mathrm{d}x = -u(x = 0^-, t)\mathrm{d}t$ at time t. Separation of variables and integration yields the result $f(t) = 1/(t + t_*)$. The kinetic energy $E(t) = \int_\Omega \frac{1}{2} u^2(x,t)\mathrm{d}x$ is therefore given as

$$E(t) = \int_{-1}^{1} \frac{1}{2} f^2(t)\left(x - \text{sign}(x)\right)^2 \mathrm{d}x = \frac{f^2(t)}{3} \ . \tag{7.13}$$

The kinetic energy dissipation rate is obtained by differentation which yields a $(t + t_*)^{-3}$ decay for times $t \geq t_*$. The dissipation rate is $\Delta u^3/12$ when expressed in terms of the jump Δu of

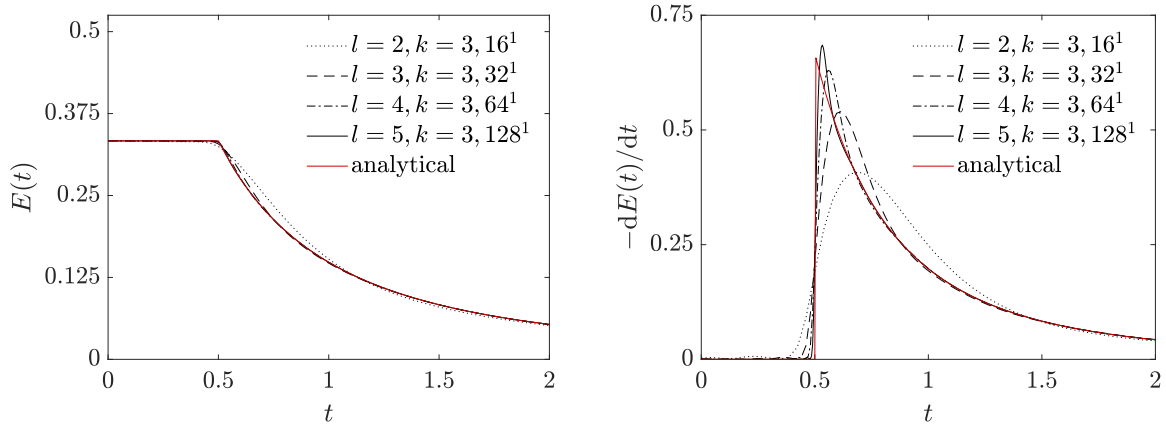


Figure 7.3: One-dimensional inviscid Burgers equations with hat function as initial condition: temporal evolution of kinetic energy as well as dissipation rate and convergence towards analytical profile for a mesh refinement study with refine levels $l = 2, \dots, 5$ and polynomial degree $k = 3$, resulting in effective resolutions of $16^1, \dots, 128^1$.

the solution, in agreement with the result in Dubrulle (2019) where it is noted that this inviscid dissipation is identical to the dissipation of the viscosity solution in the limit $\nu \to 0$. In Figure 7.3, results are shown for both the kinetic energy and the dissipation rate for a sequence of mesh refinement levels of $l = 2, \dots, 5$ with degree $k = 3$, resulting in effective resolutions of $16^1, \dots, 128^1$. For increasing spatial resolution, the numerical results converge to the analytical profiles. It can be seen that achieving grid-convergence for the dissipation rate requires higher spatial resolutions as compared to the temporal evolution of the kinetic energy itself. This is expected since the dissipation rate contains a temporal derivative that results in a higher sensitivity with respect to deviations (here numerical discretization error) from the exact solution.

Figure 7.4 shows the same results for the problem with sine function as initial condition. It can be observed that the onset of energy dissipation is smooth as opposed to the hat function where the kinetic energy exhibits a kink and the dissipation rate a jump at the time of the singularity. In other words, the occurrence of a finite-time singularity does not imply an instantaneous onset of dissipation. This should be kept in mind when considering the three-dimensional inviscid Taylor–Green problem, which is a problem that also starts from sine-like initial data.

The point to make here is that a discretization scheme involving purely numerical mechanisms of dissipation can provide the physically correct amount of dissipation for a sufficiently fine spatial resolution, see also the discussion in the introduction. Note that this is fundamentally different from viscosity solutions u^ν for small $\nu > 0$, for which the required dissipation is realized by the additional viscous term in the equations and for which the dissipation stemming from the numerical discretization scheme tends to zero if the mesh resolves the viscosity solution u^ν exhibiting steep but finite gradients. Although the one-dimensional Burgers equation can not reflect the complexity of three-dimensional turbulent flows, these results put confidence in numerical discretization schemes to also predict the solution in a physically correct way for the higher-dimensional problems studied below.

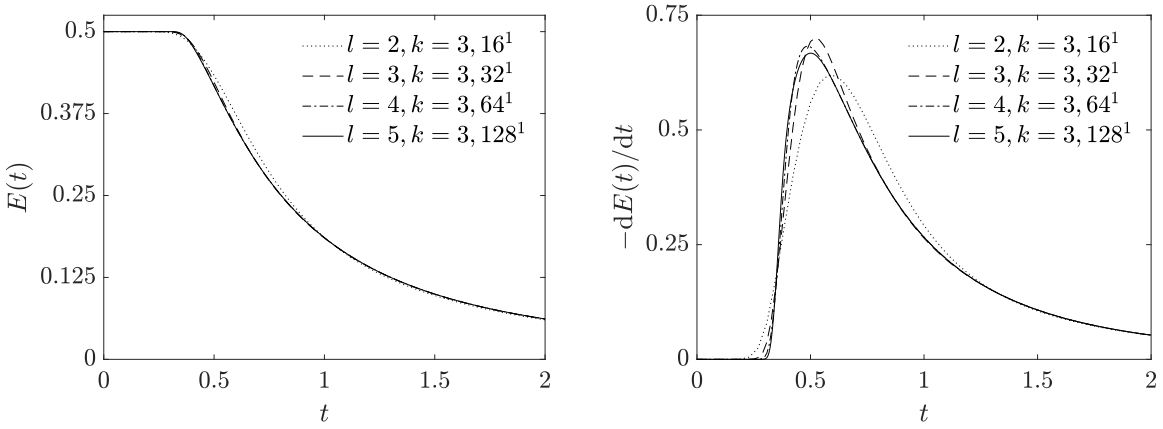


Figure 7.4: One-dimensional inviscid Burgers equations with sine function as initial condition: temporal evolution of kinetic energy as well as dissipation rate and mesh refinement study with refine levels $l = 2, \ldots, 5$ and polynomial degree $k = 3$, resulting in effective resolutions of $16^1, \ldots, 128^1$.

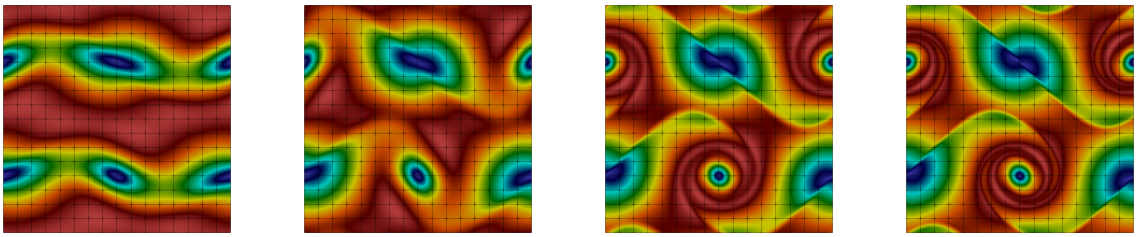
7.3.2 Two-dimensional shear layer problem

Next, the two-dimensional shear layer roll-up problem (Brown 1995) is considered, for which the initial velocity is given as

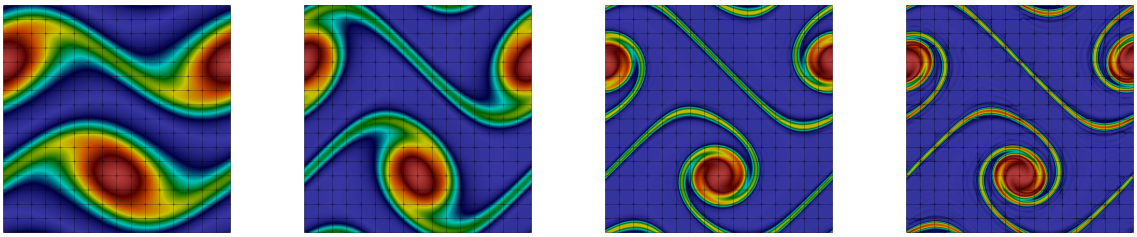
$$\boldsymbol{u}(\boldsymbol{x}, t = 0) = \left(\tanh\left(\rho(0.25 - |x_2 - 0.5|)\right), \delta \sin\left(2\pi x_1\right)\right)^{\mathsf{T}}. \tag{7.14}$$

Following Brown (1995), the two parameters ρ, δ are set to $\rho = 30$ and $\delta = 0.05$. The problem is solved on the domain $\Omega = [0, 1]^2$ with periodic boundaries in both directions. In the following, different viscous simulations with viscosities $\nu = 2.5 \cdot 10^{-3}, 10^{-3}, 10^{-4}$ are considered, as well as the inviscid limit with $\nu = 0$. The mesh is uniform Cartesian with $(2^l)^2$ elements for refinement level l, the polynomial degree of the shape functions is $k = 7$, resulting in the effective resolution of $\left((k+1)2^l\right)^2$. The simulations are run for the time interval $0 \leq t \leq 4$. The time step size is adapted dynamically with a Courant number of $\mathrm{Cr} = 0.25$.

The aim of this example is to verify the robustness and accuracy of the present high-order discontinuous Galerkin discretization for a simple two-dimensional example. As mentioned in the introduction, the energy is conserved for the two-dimensional incompressible Euler equations, and this property should be preserved by a consistent discretization scheme for sufficiently fine spatial resolutions. Figure 7.5 shows contour plots of velocity magnitude and vorticity magnitude at time $t = 1.2$ for a mesh with 16^2 elements (refinement level $l = 4$) for different values of the viscosity. In Figure 7.6, the temporal evolution of the kinetic energy and the kinetic energy dissipation rate is shown for the different viscosity values. For each viscosity, results obtained on three meshes of increasing resolution with $4^2, 8^2$, and 16^2 elements are shown. For large viscosities, $\nu = 2.5 \cdot 10^{-3}$ and 10^{-3}, the results for the temporal evolution of the kinetic energy and dissipation rate coincide for all meshes. Also for the smallest viscosity of $\nu = 10^{-4}$ and the inviscid limit $\nu = 0$ the results obtained on the two finest meshes coincide and only minor deviations can be observed for the coarsest mesh. This is in qualitative agreement with the contour plots of the velocity magnitude in Figure 7.5, which demonstrate that the velocity field



(a) velocity magnitude for $\nu = 2.5 \cdot 10^{-3}, 10^{-3}, 10^{-4}, 0$ (from left to right)



(b) vorticity magnitude for $\nu = 2.5 \cdot 10^{-3}, 10^{-3}, 10^{-4}, 0$ (from left to right)

Figure 7.5: Two-dimensional shear layer roll-up problem: contour plots of velocity magnitude and vorticity magnitude at time $t = 1.2$ for four different values of the viscosity (blue indicates low value and red high value). The results shown correspond to a mesh with 16^2 elements with a polynomial degree of the shape functions of $k = 7$ (effective resolution 128^2).

is smooth and well-resolved on the finest mesh for all viscosities. The resolution requirements are higher for the vorticity containing spatial derivatives of the velocity field. As already noted in Brown (1995), the vorticity field is still not well-resolved even if convergence has already been achieved for the velocity or kinetic energy. It can be seen from Figure 7.5 that the vorticity field is well-resolved for the viscous cases $\nu = 2.5 \cdot 10^{-3}, 10^{-3}, 10^{-4}$, but shows grid-dependence with numerical artefacts in the form of elevations of the vorticity at the element corners especially in the thin shear layer that is most difficult to resolve. In agreement with what is expected physically, the kinetic energy dissipation rate tends to zero for $\nu \to 0$ and the kinetic energy is conserved in the inviscid limit $\nu = 0$. Of particular importance w.r.t. the interpretation of results shown in Section 7.3.3 for the three-dimensional Taylor–Green problem is the observation that the numerical dissipation occurring in the inviscid limit $\nu = 0$ for coarse spatial resolutions decreases to zero under mesh refinement for this two-dimensional problem.

An important aspect concerns the numerical robustness of the discretization scheme. For example, the work by Chalmers et al. (2019) reports instabilities for the same shear layer problem with viscosity $\nu = 0$ for a discontinuous Galerkin discretization with polynomial degree $k = 7$ and refinement level $l = 4$. This originates from the fact that the stabilized discretization techniques developed in Chapter 2 that render the discretization robust in under-resolved scenarios (divergence and continuity penalty terms) have not been applied in that study. No robustness problems have been observed for the present discretization scheme even for the coarsest resolutions, and also for refinement levels of $l = 0, 1$ not explicitly shown here, which is a

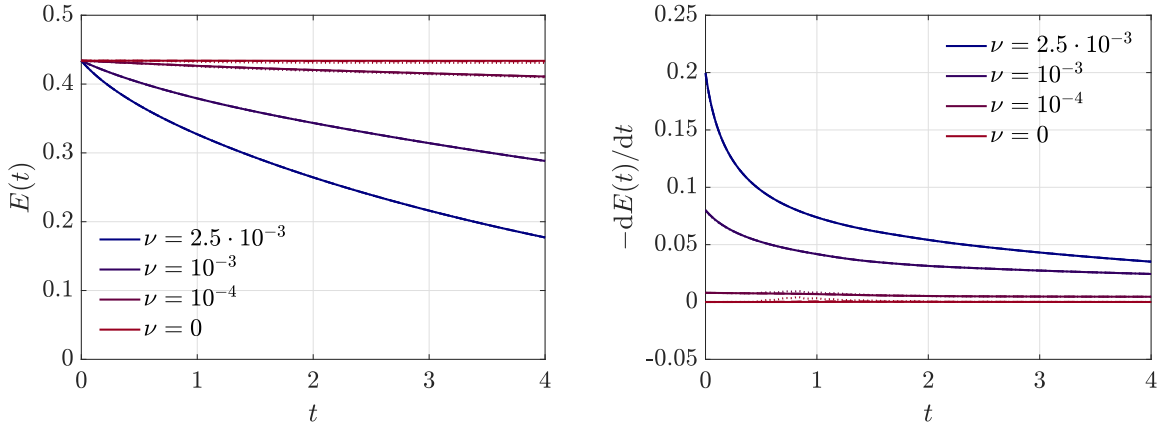


Figure 7.6: Two-dimensional shear layer roll-up problem: temporal evolution of kinetic energy and kinetic energy dissipation rate for decreasing viscosity values of $\nu = 2.5 \cdot 10^{-3}, 10^{-3}, 10^{-4}$, and 0. For each visosity, results are shown for three different effective resolutions of 32^2 (dotted lines), 64^2 (dashed lines), and 128^2 (solid lines) corresponding to meshes with $4^2, 8^2$, and 16^2 elements with polynomial degree $k = 7$.

prerequisite to obtain a feasible incompressible flow solver for three-dimensional turbulent flow problems that appear to be even more challenging in terms of numerical stability. Instabilities have also been reported for continuous spectral element discretizations for this two-dimensional shear layer problem, where filtering techniques can be used to recover stability (Fischer and Mullen 2001). A discretization technique with properties similar to the present stabilized DG discretization in terms of robustness and accuracy are exactly divergence-free $H(\mathrm{div})$-conforming discretizations, see for example the studies by Fu (2019), Guzmán et al. (2016) analyzing this shear-layer problem, the study by Schroeder and Lube (2018) discussing other two-dimensional examples such as the Kelvin–Helmholtz instability problem, and the study by Fehn et al. (2019a) comparing $H(\mathrm{div})$- and stabilized L^2-conforming discretizations for three-dimensional turbulent flow problems in under-resolved scenarios.

7.3.3 Three-dimensional Taylor–Green vortex problem

Finally, the 3D Taylor–Green vortex problem (Taylor and Green 1937) is considered, which is defined by the following initial velocity field

$$\boldsymbol{u}(\boldsymbol{x}, t = 0) = (\sin x_1 \cos x_2 \cos x_3, -\cos x_1 \sin x_2 \cos x_3, 0)^{\mathsf{T}} . \tag{7.15}$$

The reader is also referred to Section 2.6.7.2 for a more detailed description of this test case. Results of viscous simulations for increasing Reynolds number $\mathrm{Re} = \frac{1}{\nu}$ have already been shown in Figure 7.1. In the following, the focus in entirely on the inviscid limit $\nu = 0$. The simulations are run over the time interval $0 \leq t \leq T = 20$ to cover the different flow regimes of laminar flow, transition to turbulence, and decaying turbulence. To reduce computational costs for fixed resolution of the flow (or to increase the effective resolution for a given amount of computational costs), it is common practice to exploit the symmetry of the Taylor–Green problem and

simulate the flow on the impermeable box $\Omega = [0, \pi]^3$ with symmetry boundary conditions on all boundaries (Brachet et al. 1983), i.e.,

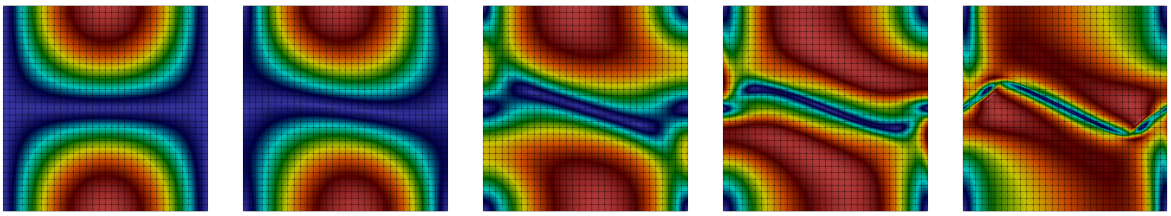
$$\boldsymbol{u} \cdot \boldsymbol{n} = 0 \, , \frac{\partial \boldsymbol{u}}{\partial n} = 0 \, , \tag{7.16}$$

as opposed to the periodic box $\Omega = [-\pi, \pi]^3$ that is also used in computational studies. This optimization allows to reduce computational costs by a factor of 8 for the present DG discretization. Further symmetries can be exploited by spectral methods leading to the so-called fundamental box (Brachet et al. 1983, Kida 1985).

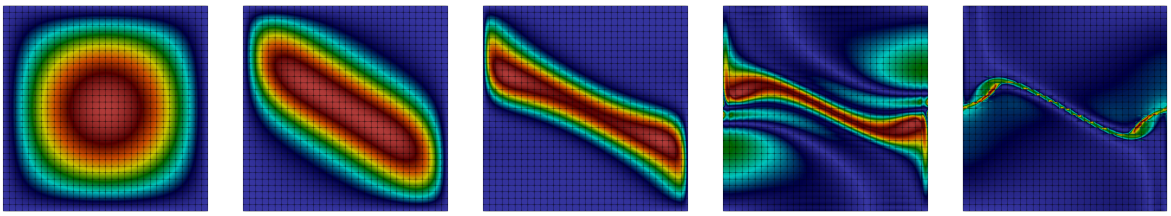
The computational domain $\Omega = [0, \pi]^3$ is discretized using a uniform Cartesian grid consisting of $(2^l)^d$ elements, where l denotes the level of refinement. The number of unknowns is given as $N_{\mathrm{DoFs}} = (2^l)^d (d(k_u + 1)^d + (k_p + 1)^d) = (2^l)^d (d(k + 1)^d + k^d)$. It is common practice in the literature to express the effective mesh resolution in terms of the periodic box to obtain comparability between different discretization techniques that exploit different levels of symmetry. Hence, the effective spatial resolution is defined as $(2^{l+1}(k + 1))^d$, e.g., the effective resolution is 64^3 for refine level $l = 3$ and polynomial degree $k = 3$. Absolute tolerances of 10^{-12} and relative tolerances of 10^{-6} are used for the iterative linear solvers, where relative tolerance means that the residual is reduced by a factor of 10^{-6} compared to the initial residual that uses as initial guess a high-order extrapolation of the solution from previous time steps. The polynomial degree used for the TGV simulations is $k = 3$ and adaptive time stepping with $\mathrm{Cr} = 0.25$ is used for all simulations. The default value $\zeta = 1$ is used for the penalty factors of the divergence and continuity penalty terms, except for the finest resolution of 8192^3 where the penalty factors have been increased by a factor $\zeta = 2$ compared to the standard definition. For this fine resolution, the simulation remained stable also for the default value of $\zeta = 1$, but oscillations in the maximum vorticity have been observed at early times, indicating the need for a slightly larger penalization of the divergence-free constraint and normal continuity of the velocity field. Since these oscillations disappeared when increasing the penalty factors by a factor of 2, this value has finally been used for this highest resolution simulation. The highest resolution of 8192^3 has $N_{\mathrm{DoFs}} = 2.35 \cdot 10^{11}$ unknown degrees of freedom, and $2.27 \cdot 10^5$ time steps have been solved during this simulation. The computations have been performed on the SuperMUC-NG supercomputer using almost $100\mathrm{k}$ cores for the highest resolution, requiring a run time of approximately 8.4 days.

7.3.3.1 Recapitulating the state-of-the-art

This section briefly summarizes the type of discretization, the maximal effective resolutions, and the final time T of the simulations considered in previous numerical studies for the three-dimensional inviscid Taylor–Green vortex problem. In Brachet et al. (1983), a spectral method with maximum resolution of 256^3 (exploiting symmetry) has been used and a direct simulation has been performed up to times $t \leq 4$. In a subsequent work by Brachet et al. (1992), a maximum resolution of 864^3 (exploiting symmetry) has been reached, and again a direct simulation has been performed up to times $t \leq 4$. A comparsion of a spectral method and WENO finite difference method can be found in Shu et al. (2005), where a maximum resolution of 368^3 (exploiting symmetry) and a simulation up to times $t \leq 6$ has been considered. A modal discontinuous Galerkin method has been studied in Chapelier et al. (2012), with the simulations performed



(a) velocity magnitude on plane $x = \pi$ at times $t = 0, 1, 2, 3, 4$ (from left to right)



(b) vorticity magnitude on plane $x = \pi$ at times $t = 0, 1, 2, 3, 4$ (from left to right)

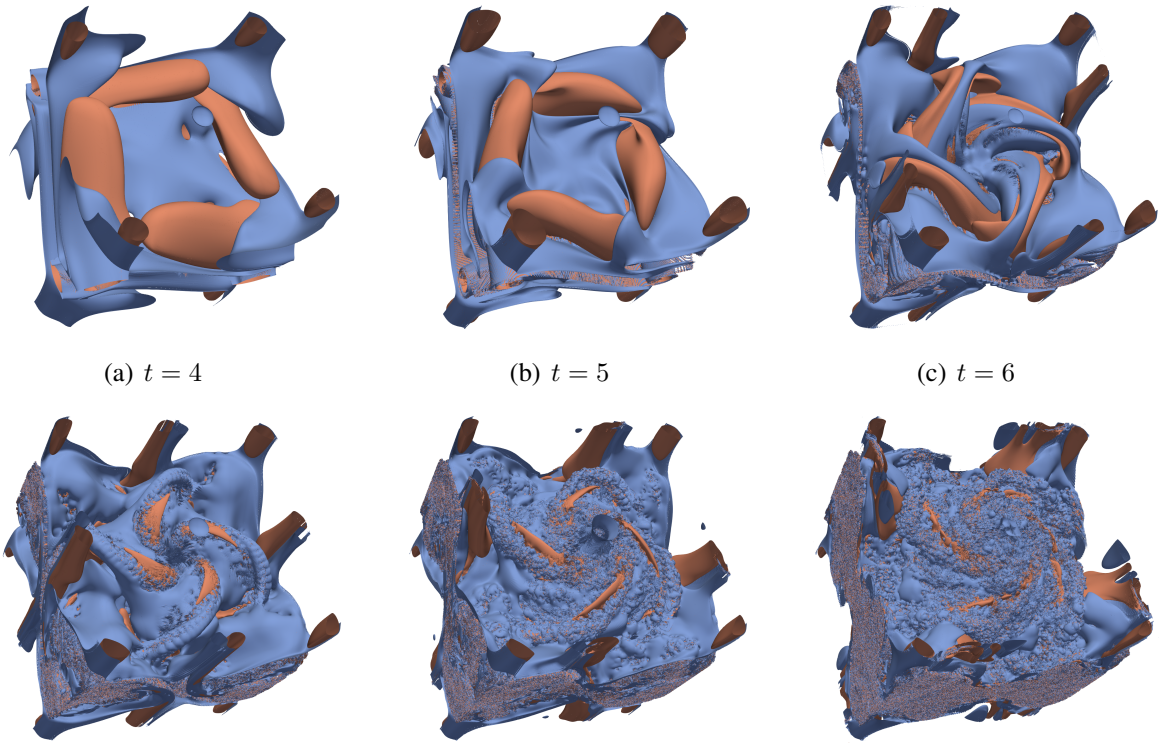
Figure 7.7: Three-dimensional inviscid Taylor–Green problem: contour plots of velocity magnitude and vorticity magnitude (blue indicates low value and red high value) on plane $x = \pi$ of impermeable box. The results correspond to a mesh with 32^3 elements with a polynomial degree of the shape functions of $k = 3$ (effective resolution 256^3).

up to times $t \approx 5 - 7$ until the simulations became unstable, with the maximum resolution of around 96^3 for different polynomial degrees from $k = 1$ to $k = 5$. DG discretizations used to study the inviscid TGV problem have also been analyzed in Fernandez et al. (2018), Manzanero et al. (2020), Moura et al. (2017a,b), Schroeder (2019), but with a focus on LES modeling. The study by Cichowlas and Brachet (2005) used a spectral method with maximum resolution of 2048^3 with simulations performed up to times $t \leq 4$. The highest resolution of 4096^3 has been achieved in Bustamante and Brachet (2012) using a spectral method, and the simulations have been performed up to times $t \leq 4$.

In these works, indications of finite-time singularities are mentioned. In Morf et al. (1980), $t_* = 5.2$ is obtained from power series expansions. A more accurate variant using power series expansions presented in Brachet et al. (1983) leads to $t_* = 4.4 \pm 0.2$. Furthermore, the study by Brachet et al. (1983) reports indirect evidence for a finite-time singularity according to the direct numerical simulation results, but the authors conclude that the resolution of 256^3 is not sufficient to investigate the occurrence of finite-time singularities for times $t \geq 4$. The more recent study by Bustamante and Brachet (2012) estimates a blow-up time of $t_* \approx 4$ and concludes that the results are not inconsistent with the occurrence of a singularity. The work by Larios et al. (2018) obtains a blow-up time of $t_* \approx 4.2$ similar to the blow-up time in Brachet et al. (1983).

7.3.3.2 Results in physical space

The early stage of the Taylor–Green vortex evolution with the formation of thin flow structures is visualized in Figure 7.7. Similar results have been shown and discussed in detail already



(a) $t = 4$ (b) $t = 5$ (c) $t = 6$

(d) $t = 7$ (e) $t = 8$ (f) $t = 9$

Figure 7.8: Three-dimensional inviscid Taylor–Green problem: iso-surfaces of Q-criterion at times $t = 4, 5, 6, 7, 8, 9$, where the blue surface corresponds to a value of -0.5 and the orange surface to a value of 0.5. The results correspond to a mesh with 256^3 elements with a polynomial degree of the shape functions of $k = 3$ (effective resolution 2048^3). Visualization by Nisarg Patel @LRZ in Garching, Germany.

in Brachet et al. (1983) for the same effective resolution of 256^3 using a spectral method. In agreement with the results for the two-dimensional shear layer problem, one can observe that the velocity field is resolved at all times, while under-resolution effects are clearly visible in the contour plots of the vorticity magnitude at later times $t = 3$ and $t = 4$ for the chosen resolution. A high-resolution visualization of the thin vortex sheet shown in Figure 7.7 with a volume rendering of the vorticity magnitude has been shown in Bustamante and Brachet (2012) for an effective resolution of 4096^3. Figure 7.8 shows visualization results for a high-resolution simulation (effective resolution 2048^3) at later times $t = 4, 5, 6, 7, 8, 9$, around which small-scale features occur and transition to turbulence takes place. These results illustrate that the present discretization scheme does not lead to thermalization, which appears to be a prerequisite to obtain physically meaningful results, compare for example the present results to the thermalized results shown in (Schroeder 2019, Figure 9.14). To the best of the author's knowledge, numerical results and visualizations of this quality have not been shown to date in the literature for the inviscid TGV problem. Flow visualization is discussed in the literature as one possibility to trace finite-time singularities, but appears to be impractical due to the difficulties in handling large data sets for high-resolution simulations necessary for such investigations, and due to the difficulties

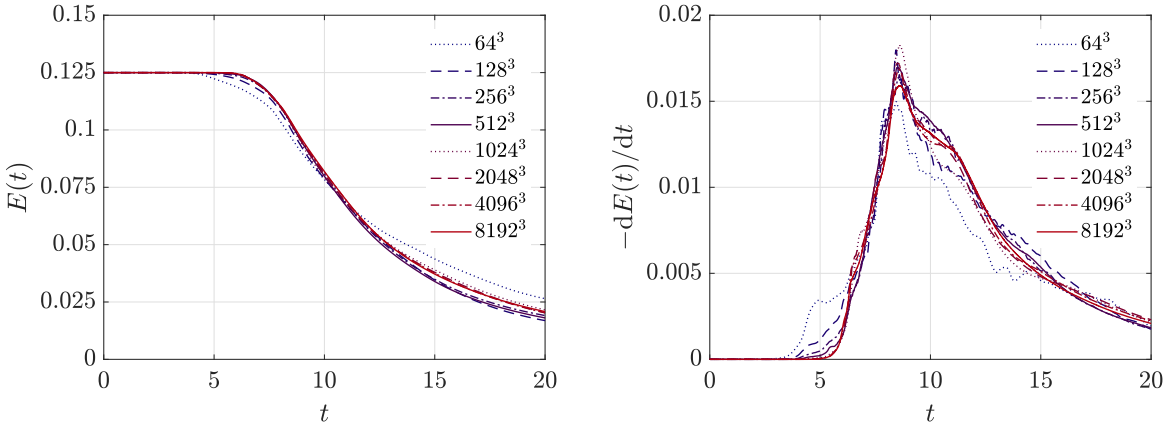


Figure 7.9: Three-dimensional inviscid Taylor–Green problem: temporal evolution of kinetic energy (left) and kinetic energy dissipation rate (right) for increasing effective spatial resolution.

in visualizing singularities (that do not show up as singularities for a finite-resolution numerical simulation). Hence, the attention is turned to other techniques in the following.

Numerical results of a mesh convergence study for refinement levels $l = 3, ..., 10$ and polynomial degree $k = 3$ are presented in the following. Figure 7.9 shows the temporal evolution of the kinetic energy and the kinetic energy dissipation rate. At small times t, the energy is constant and the energy dissipation rate is zero. This agrees with the expected theoretical behavior stating energy conservation as long as the solution remains smooth and has also been shown in previous works in a similar way. As mentioned above, the simulations are not terminated once the simulation can be expected to become under-resolved, but the simulations are instead continued until $t = 20$. Depending on the effective mesh resolution, an onset of energy dissipation can be observed that shifts towards later times for increasing spatial resolutions. However, this time of onset of dissipation does not seem to be pushed beyond $t \approx 5$ even for the finest spatial resolutions. This is illustrated more clearly in Figure 7.10, which plots the kinetic energy dissipation rate in logarithmic scaling as well as the error against the fine-resolution simulation (ref). The fact that the dissipation rate of the simulations with $2048^3, 4096^3$ resolution "converges" to that of the finest resolution at a time $t \approx 5$ for 2048^3 and $t < 5$ for 4096^3 is consistent with a potential blow-up time $t_* < 5$. Overall, the interesting phenomenological observation is made that the kinetic energy evolution and its dissipation rate tend to converge to a dissipative solution rather than an energy-conserving solution with vanishing dissipation rate. As in high-Reynolds number viscous simulations of this problem, the kinetic energy dissipation rate reaches a maximum at $t = 8 - 9$ and decreases afterwards. In Section 7.3.3.5, grid-convergence of the sequence of discrete solutions to a dissipative reference solution is investigated in more detail. Note that the present results for the dissipation rate agree well with those shown in Cichowlas et al. (2005) for an energy-conserving scheme, where an effective dissipation rate is deduced from the thermalized energy $E_{\mathrm{th}}(t)$, the energy associated to the small scales with wavenumber $k > k_{\mathrm{th}}$, where k_{th} is the wavenumber at which the energy spectrum exhibits a local minimum.

It is now examined whether this behavior is consistent with what is observed for the temporal evolution of the maximum vorticity and the enstrophy shown in Figure 7.11. For both quantities,

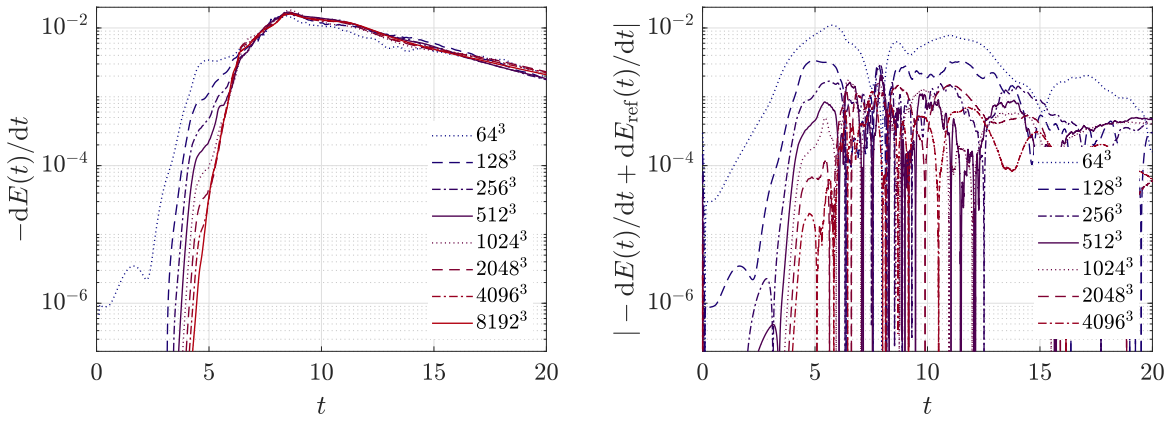


Figure 7.10: Three-dimensional inviscid Taylor–Green problem: temporal evolution of kinetic energy dissipation rate in logarithmic scaling (left) and error against fine-resolution simulation (right).

one can immediately identify three phases: (i) a first phase up to approximately $t \approx 3$ in which the flow is well resolved for all spatial resolutions so that the results essentially overlap for all simulations, (ii) an intermediate phase $3 < t < 5$ in which the different simulations start to deviate from each other due to under-resolution effects depending on the spatial resolution of each simulation, and (iii) a final phase $t > 5$ in which the results of all simulations deviate substantially due to the different resolution capabilities of the different simulations. In the first phase, the vorticity first decreases, reaches a minimum, and then begins to grow exponentially in agreement with the results shown in (Bustamante and Brachet 2012, Figure 1 (b)). In Figure 7.11, a reference curve with $\exp\left(\frac{2}{3}t\right)$ growth is added, describing the growth of the maximum vorticity very well in this regime. In the second phase at around $t \approx 3.5$, the maximum vorticity begins to grow substantially faster, and the growth of vorticity essentially depends on the spatial resolution that is directly linked to the maximum velocity gradient that can be represented on a given mesh. As already mentioned in the introduction, for every simulation that does not blow up due to numerical instabilities of a discretization scheme, the maximum vorticity will remain finite no matter how fine the spatial resolution is. Therefore, the occurrence of a finite-time singularity with $\lim_{t \to t_*} \|\boldsymbol{\omega}\|_\infty = \infty$ remains speculative. While the present results might be considered consistent with such a vorticity blow-up scenario, a concrete blow-up time $t = t_*$ can not be identified. It can rather be observed that the time of maximum vorticity reached in this second phase is shifted to later times also for the finest spatial resolutions. At the same time, one might argue that a finite-time blow-up at a time $t = t_*$ with $t_* \approx 4.5 - 5$ would produce results similar to those shown here, with the maximum vorticity following the exact profile until the curve of a specific spatial resolution branches off due to under-resolution of the simulation. In such a scenario, one would expect the maximium vorticity to grow by a factor of 2 for refinement level $l \to l+1$ due to the mesh size being reduced by a factor of 2, allowing numerical gradients becoming twice as large. While such an increase in maximum vorticity is observed from one refinement level to the next, it is not clear whether this suspected blow-up would happen in finite time. In the third phase, the maximum vorticity reaches a global maximum between $t = 6 - 7$ for each resolution before it starts to decrease slowly. In this phase, the maximum vorticity is

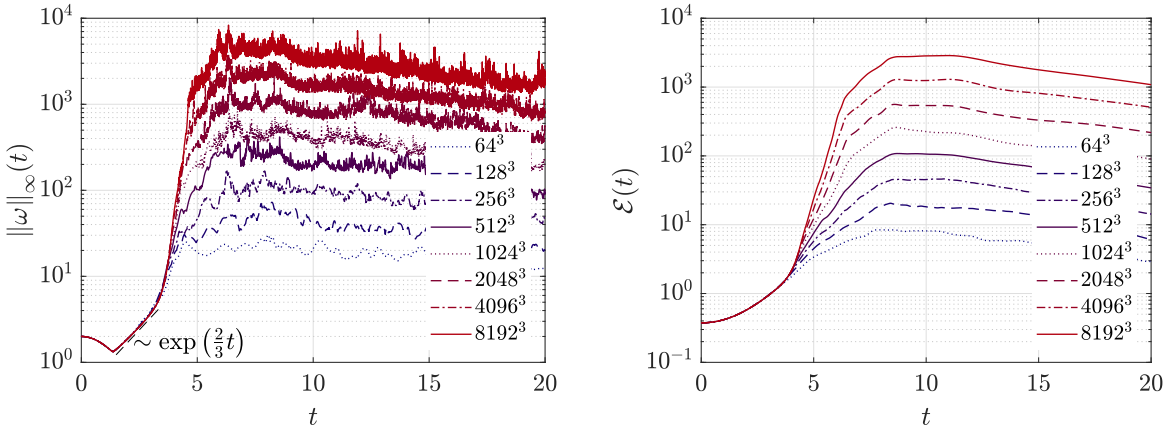


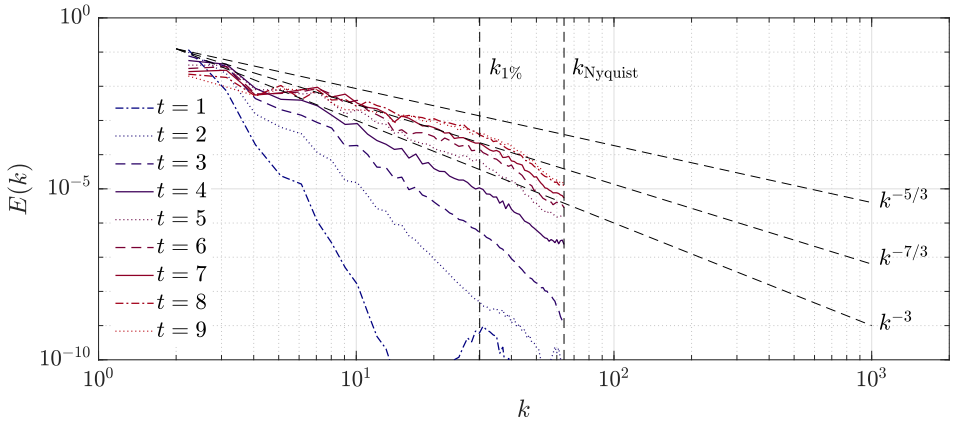
Figure 7.11: Three-dimensional inviscid Taylor–Green problem: temporal evolution of maximum vorticity (left) and enstrophy (right) for increasing effective spatial resolution.

offset by a factor of approximately 2 from one refine level to the next. This is a clear indication that none of the simulations is able to resolve the finest structures, and it is plausible that a factor of 2 in mesh size also gives a factor of 2 in maximum vorticity. Finally, the maximum vorticity shows a strongly fluctuating behavior in the third phase. Note that the maximum vorticity is determined numerically by taking the maximum over all quadrature points, i.e., the vorticity field is sampled in discrete points. This effect is negligible for well-resolved scenarios but might explain an oscillatory behavior in case a local maximum travels through the domain.

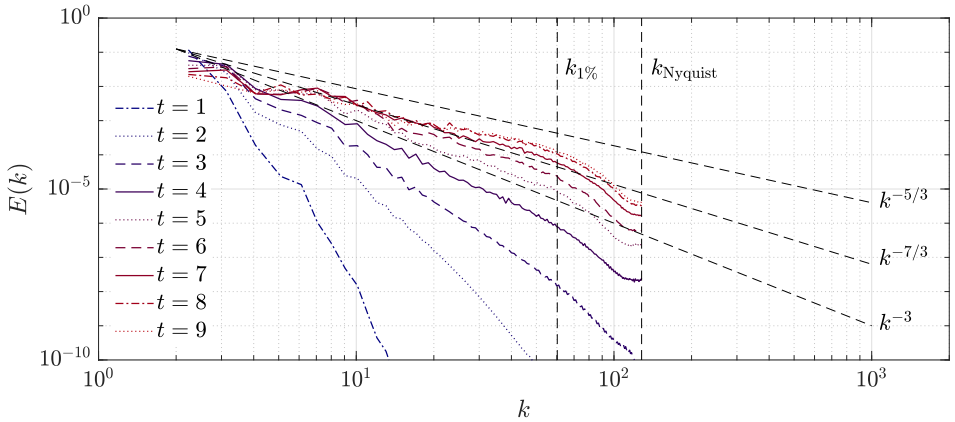
The temporal evolution of the enstrophy is overall similar to that of the maximum vorticity. An important difference is that the enstrophy does not reach a local minimum at early times as observed for the maximum vorticity. In the second phase, the growth of enstrophy is more moderate compared to the maximum vorticity. In the third phase, the enstrophy curves can be described essentially as smoothed variants of the maximum vorticity from which the high-frequency content has been removed. A possible explanation for the enstrophy behavior in the second and third phase is that the enstrophy is not a local quantity, but an average in space over the computational domain. Again, a growth in enstrophy by a factor of 2 from one mesh level to the next is observed at later times, which is consistent with an enstrophy evolution theoretically taking infinite values, or taking finite values but much larger than those obtained numerically in Figure 7.11. Considering Onsager's conjecture as valid, it is clear that one can not expect convergence for the temporal evolution of the maximum vorticity and the enstrophy. Theoretically, convergence can then only be expected for the kinetic energy evolution and to some extent for kinetic energy spectra up to the resolution limits of the discretization scheme, as discussed in the following.

7.3.3.3 Results in spectral space

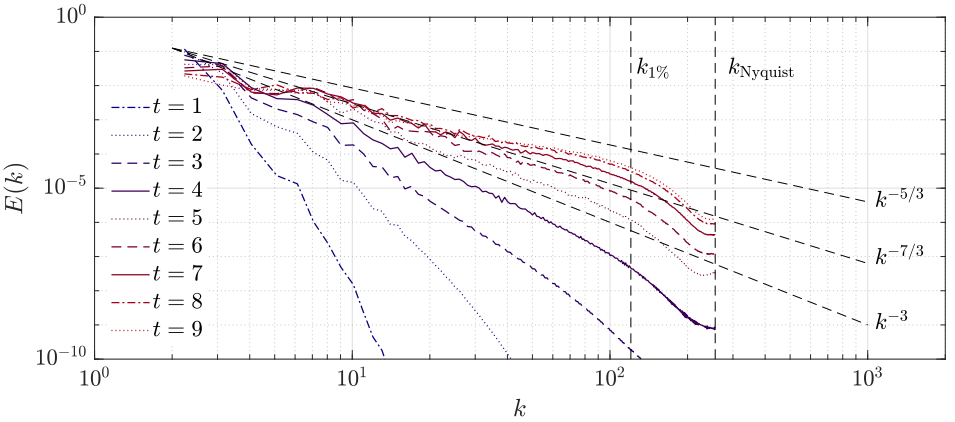
Figure 7.12 shows kinetic energy spectra for increasing spatial resolution at various instants of time, namely at $t = 1, \ldots, 9$ in steps of width 1. Results are shown for resolutions of 128^3 to 2048^3. High computational costs and memory requirements of the FFT part of the simulations did not allow to perform the spectral analysis for the highest resolutions of 4096^3 and 8192^3. For a discussion of the general behavior of energy spectra as a function of time regarding the



(a) effective resolution of 128^3

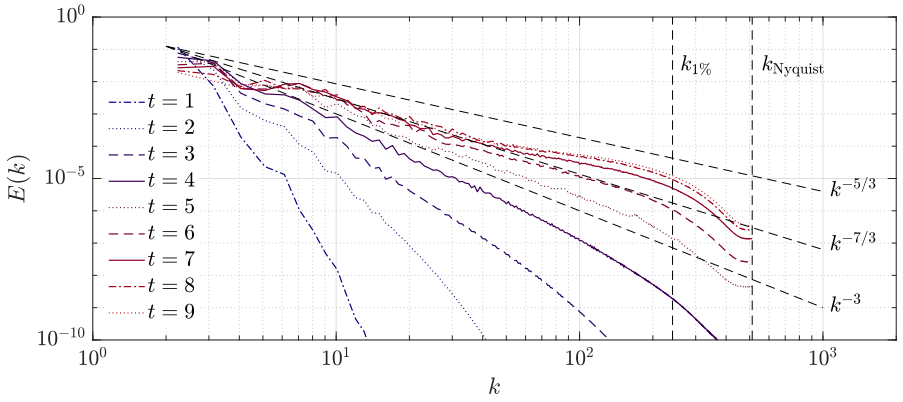


(b) effective resolution of 256^3

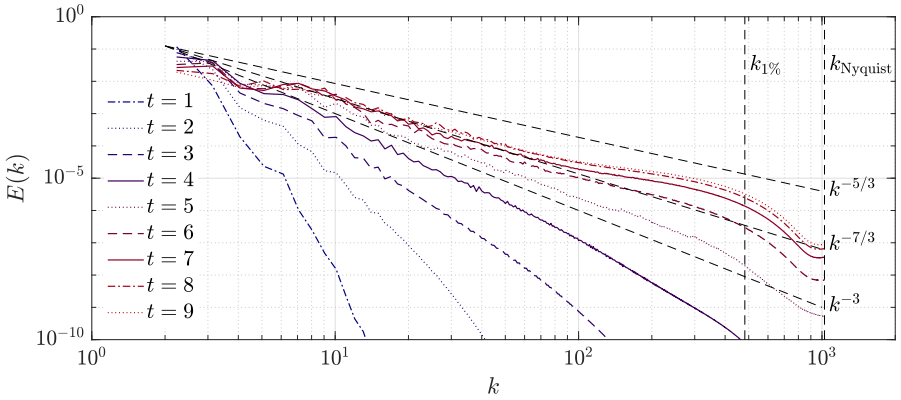


(c) effective resolution of 512^3

Figure 7.12: Three-dimensional inviscid Taylor–Green problem: kinetic energy spectra for effective resolutions of $128^3, 256^3, 512^3$ at times $t = 1, 2, ..., 9$.



(d) effective resolution of 1024^3



(e) effective resolution of 2048^3

Figure 7.12: Three-dimensional inviscid Taylor–Green problem: kinetic energy spectra for effective resolutions of $1024^3, 2048^3$ at times $t = 1, 2, ..., 9$.

early time behavior $t \leq 4$, the reader is referred to the works by Brachet et al. (1983), Bustamante and Brachet (2012), Cichowlas and Brachet (2005), where it is shown how the energy spectra can be fitted to functions of the form $E(k,t) = C(t)k^{-n(t)} \exp(-2k\delta(t))$ and where values obtained for $n(t)$ and $\delta(t)$ are discussed in detail. To verify the present results, references curves of slope $n = 3$ (blow-up of enstrophy) and $n = 5/3$ (Kolmogorov's inertial scaling law) or $n = 7/3$ (motivated by results obtained in Piatkowski (2019) for viscous Taylor–Green simulations) are included in the figures. The energy spectra are compared against the slope $n = 3$ as a means to investigate the plausibility of a potentially singular behavior and to identify a time $t = t_*$ at which such a blow-up could occur, as motivated in Section 7.2. Once the flow has transitioned to a fully turbulent state, it can be expected that the energy spectrum exhibits some form of Kolmogorov scaling. For this purpose, the energy spectra are considered at times $t = 8$ and 9 where the maximum dissipation rate occurs, even though the flow might not be fully homogeneous isotropic at that time and the results might deviate from Kolmogorov's $k^{-5/3}$ scaling. To quantify the resolution capabilities of the present discretization, the Nyquist wavenumber k_{Nyquist} is also plotted as well as the wavenumber $k_{1\%}$ according to the 1%-rule by Moura et al. (2017a)

that aims at obtaining an accurate resolution limit for upwind-like DG discretizations for a specific polynomial degree of the function space.

Figure 7.12 shows that the range of scales resolved by the numerical method increases with increasing spatial resolution as expected theoretically, and that the resolution limit for polynomial degree 3 is described very well by the 1%-rule corresponding to this degree. The energy spectra reach a slope of -3 between $t = 4$ and $t = 5$, so that the results are consistent with the occurrence of singular behavior around that time ($4 < t_* < 5$), in agreement with the rapid growth of the maximum vorticity observed in the same time interval, see Figure 7.11. The spectra in the inertial range show a decay slightly stronger than $k^{-5/3}$ and better agreement is achieved when compared to a $k^{-7/3}$ scaling that is also shown in Figure 7.12. This behavior has already been observed in Piatkowski (2019) for viscous Taylor–Green vortex simulations, where it was found that Kolmogorov's $k^{-5/3}$ scaling can only be observed at later times, e.g., $t \approx 20$. Regarding the inertial scaling, the present results are therefore in agreement with results shown in the literature. Towards the Nyquist wavenumber, a moderate pile-up of energy can be observed by comparison against the $-5/3$ and $-7/3$ references slopes before the energy falls off very rapidly. The energy pile up is characteristic of this type of high-order discontinuous Galerkin approach and becomes stronger for higher polynomial degrees, see Moura et al. (2017a,b) and references therein. This particular behavior is often the primary target when optimizing discretization schemes, see for example the recent studies by Flad and Gassner (2017), Manzanero et al. (2020), Winters et al. (2018) and references therein, where it is suggested to counteract this energy bump behavior by explicit sub-grid scale modeling. However, the recent study by Fernandez et al. (2018) shows that this topic is delicate and improvements in some quantities by the use of explicit sub-grid models cause deviations in other quantities such as the temporal evolution of the kinetic energy dissipation rate. The challenge therefore lies in improving the spectral behavior and at the same time not giving up the improved resolution capabilities (per degrees of freedom) of high-order discretizations. The overall goal can be formulated as achieving a discretization method that is accurate w.r.t. both the spectral behavior and the behavior in physical space, e.g., the temporal evolution of kinetic energy and its dissipation rate. From such a holistic view it appears to be unclear whether an explicit sub-grid scale model optimizing the energy spectrum according to the inertial $k^{-5/3}$ law is advantageous overall. This point is taken up again in the outlook in Section 7.4 discussing possible directions of future research. The energy spectra shown in (Schroeder 2019, figure 9.15) for an exactly energy-conserving discretization scheme illustrate that such a scheme leads to physical inconsistencies for the $E(k)$-curves in the inertial range.

7.3.3.4 Does the numerical dissipation have artificial or predictive character?

The dissipation of kinetic energy observed for the inviscid Taylor–Green simulations originates from the numerical method. At first sight, one might argue that changing the discretization scheme by choosing another numerical flux or varying certain parameters leads to results that are more or less dissipative, i.e., that the amount of dissipation is artificial and is determined by the discretization parameters. The results of the mesh convergence study shown above do not support this point of view, and this section provides further results that might allow insights into the predictive character of these dissipative numerical solutions. In this context, it is illustrative to study the temporal evolution of the kinetic energy and its dissipation rate under a variation of parameters of the discretization scheme. Figure 7.13 shows a parameters study of the penalty

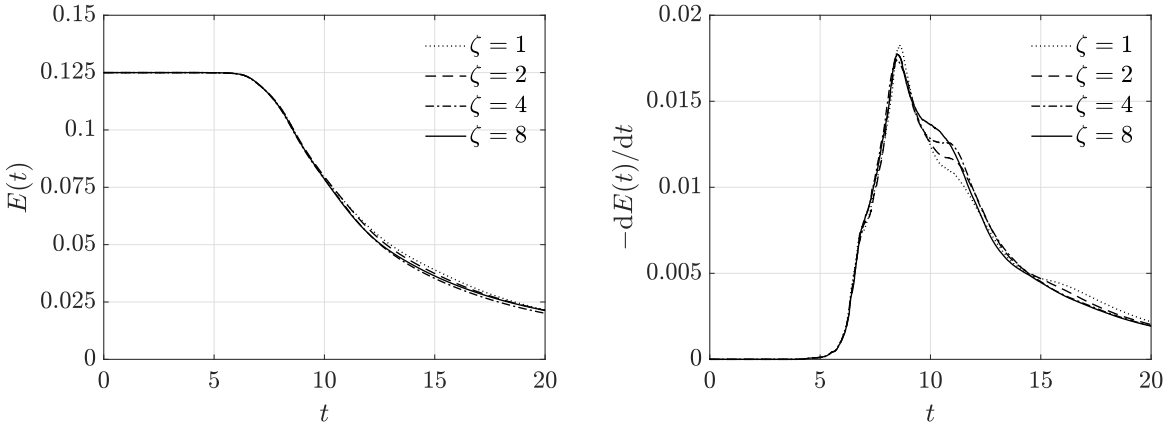


Figure 7.13: Three-dimensional inviscid Taylor–Green problem: variation of penalty factor ζ and its influence on the temporal evolution of the kinetic energy (left) and the kinetic energy dissipation rate (right) for effective resolution of 1024^3.

factor ζ of the divergence and continuity penalty terms of the present discretization, considering values of $\zeta = 1, 2, 4, 8$ by the example of the 1024^3 spatial resolution. Note that this parameter is the crucial one in stabilizing the method in the under-resolved and high-Reynolds regime, see Fehn et al. (2018b, 2019a). The overall amount of dissipation as well as the dissipation maximum are essentially unaltered by a variation of this parameter. It is worth noting that the time of onset of dissipation is also not affected by the value of ζ. This has important implications. Assuming that a non-dissipative (energy-conserving) solution is the correct physical behavior and that the numerical dissipation is artificial, one might expect that an increase of the penalty parameter affects the numerical solution more strongly, e.g., changes the amount of overall dissipation or leads to a delayed onset of dissipation due to a better fulfillment of the divergence-free constraint for example. Since such a behavior is not observed, these results are an indication that the obtained dissipative solutions have a predictive character. It is clear that the results can not be expected to be identical for different penalty factors, since the results are not grid-converged for the chosen 1024^3 resolution, meaning that discretization scheme and its parameters affect the numerical solution.

7.3.3.5 Are the results grid-converged?

Finally, the question is addressed whether and to which extent the inviscid Taylor–Green simulation results can be considered as grid-converged. For this purpose, relative L^2-errors are computed for the kinetic energy evolution and the kinetic energy dissipation rate

$$\varepsilon_E^2 = \frac{\int_{t=0}^{T} \left(E(t) - E_{\mathrm{ref}}(t)\right)^2 \mathrm{d}t}{\int_{t=0}^{T} \left(E_{\mathrm{ref}}(t)\right)^2 \mathrm{d}t} \quad , \quad \varepsilon_{\mathrm{d}E/\mathrm{d}t}^2 = \frac{\int_{t=0}^{T} \left(\frac{\mathrm{d}E(t)}{\mathrm{d}t} - \frac{\mathrm{d}E_{\mathrm{ref}}(t)}{\mathrm{d}t}\right)^2 \mathrm{d}t}{\int_{t=0}^{T} \left(\frac{\mathrm{d}E_{\mathrm{ref}}(t)}{\mathrm{d}t}\right)^2 \mathrm{d}t} \quad . \tag{7.17}$$

Since no analytical solution is available, the errors are measured using the finest resolution of 8192^3 as a reference (ref). This implies that the error can not be computed for the 8192^3

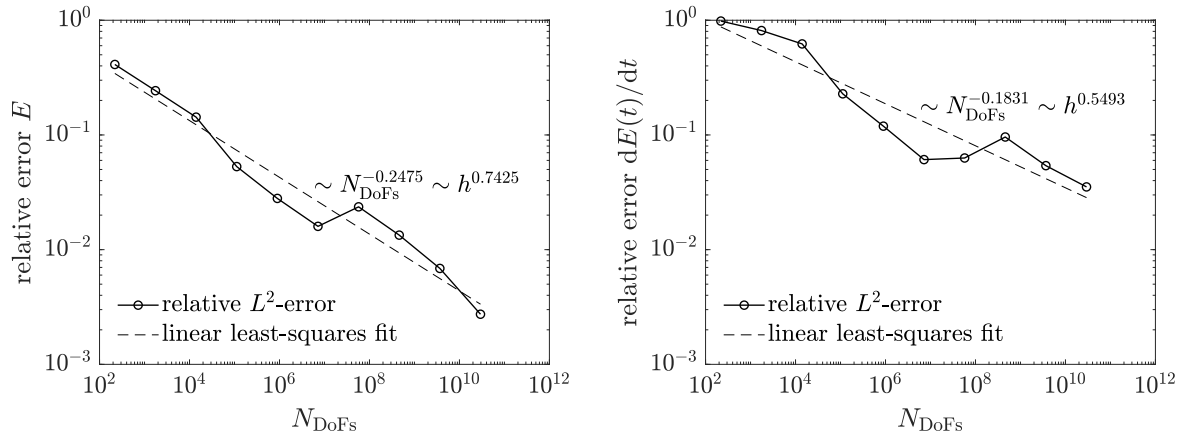


Figure 7.14: Three-dimensional inviscid Taylor–Green problem: relative L^2-errors of the temporal evolution of the kinetic energy (left) and the kinetic energy dissipation rate (right) for effective resolutions ranging from 8^3 to 4096^3.

resolution. The error of this simulation can only be roughly estimated by extrapolating the convergence trend observed for the coarser resolutions and assuming that this convergence behavior continues for the finest resolution. Defining a simulation with a relative error of 1% or less as grid-converged, the results in Figure 7.14 reveal that grid-convergence is indeed achieved in the kinetic energy evolution with errors below 1%. For the second finest resolution (the last data point in Figure 7.14), the measured error is 0.27%. For the kinetic energy dissipation rate, the error is significantly larger, demonstrating that this quantity is more sensitive, in agreement with what has been observed in Figure 7.1. For the second finest resolution, the measured error is 3.52%. While the errors can be expected to be smaller for the finest resolution 8192^3 that is used as a reference solution here, even finer resolutions would be required to achieve grid-convergence also for the kinetic energy dissipation rate in terms of the 1% error level. Figure 7.14 also shows linear least-squares fits (e.g., $\log(E) \approx a \log(N_{\mathrm{DoFs}}) + b$ for the kinetic energy) to the data obtained from the numerical experiments, where a mean converge rate of approximately $h^{3/4}$ is obtained for the kinetic energy and $h^{1/2}$ for the dissipation rate. This result – seemingly providing numerical evidence of grid-convergence to a dissipative solution of the three-dimensional incompressible Euler equations under Taylor–Green initial conditions – is the main result of this chapter. While one might conjecture that this solution could be a weak Euler solution, there is currently no rigorous mathematical theory guaranteeing convergence to a weak solution. Moreover, even if convergence to a weak Euler solution was in fact obtained, it might not be the viscosity solution for $\nu \to 0$. Finally, from the results in Figure 7.14 it can not be excluded that the dissipation rate would (slowly) tend to zero in the limit $h \to 0$ if finer spatial resolutions were realized. This decisive aspect is explained in more detail in the following: For a solution that is non-dissipative (i.e., energy-conserving) from a physical perspective, one might argue that dissipation in the numerical simulation can only originate from under-resolution and can be expected to tend to zero when increasing the resolution. However, a dissipation rate decreasing extremely slowly under mesh refinement, e.g. as $1/\log\log(1/h)$, might then (erroneously) indicate grid-convergence to a dissipative solution, although the true solution is actually energy-conserving. Against this background, the present numerical investigations according to

the indirect approach are *consistent with* or *suggestive of* finite-time singularities, but do not conclusively *provide evidence* of finite-time singularities.

7.4 Conclusion and outlook

Hunting for finite-time singularities in incompressible Euler flows is a challenging discipline. Searching for singular behavior in visualizations of three-dimensional simulation results evokes the picture of finding a needle in a haystack, as it can be expected that singularities are very localized, will never be resolved by a numerical scheme, and the amount of data for large-scale three-dimensional simulations soon becomes cumbersome. For this reason, most previous studies focused on local quantities such as the maximum vorticity, the analyticity strip width, fitting energy spectra to power law behavior, and related blow-up criteria. The present work focuses on global quantities such as the temporal evolution of the kinetic energy, avoiding geometrical complexities in visualization and striving for a clearer indication of singular behavior given that it might be computationally less demanding to resolve the kinetic energy than the vorticity in numerical simulations. This approach is called *indirect* since it exploits the connection between singular behavior and anomalous energy dissipation according to Onsager's conjecture. A decisive point is that this technique requires suitable discretization schemes that remain robust in the presence of singularities and provide mechanisms of dissipation in case no viscous dissipation is present, which is a challenge in itself. Then, the idea is that observing an energy dissipating behavior for a sequence of mesh refinement levels provides insight into the physical dissipation behavior of the problem under investigation.

This technique has been applied to one-, two-, and three-dimensional problems. Results consistent with theory have been obtained in one and two space dimensions. For the complex three-dimensional inviscid Taylor–Green problem, an energy dissipating behavior consistent with the high Reynolds number limit of viscous simulations available in the literature has been observed. The present work measures grid-convergence to a dissipative, fine-resolution numerical solution for the three-dimensional inviscid Taylor–Green problem with a measured relative L^2-error of 0.27% for the kinetic energy and 3.52% for the kinetic energy dissipation rate. The results for the temporal evolution of the kinetic energy might therefore be considered as grid-converged and serve as a reference solution for future studies. Regarding the temporal evolution of the maximum vorticity and the enstrophy, an increase of almost four orders of magnitude could be resolved for both quantities. Confidence is put into the reliability of the numerical results for this challenging problem by the circumstance that the numerical method applied here is a robust discretization scheme that remains numerically stable in the inviscid limit for all spatial resolutions that have been investigated. This is an important prerequisite to draw conclusions about a potential physical blow-up. In contrast, a numerical simulation that blows up in finite time does not allow any conclusions, since the observed blow-up is due to numerical instabilities. In other words, a physical blow-up does not imply a numerical blow-up for a finite-resolution numerical simulation.

In summary, this work wants to raise the questions (i) to which extent these results are related to weak dissipative solutions of the incompressible Euler equations, (ii) to which extent these results can then be interpreted as a numerical confirmation of the energy dissipation anomaly, and (iii) whether these results would imply finite-time singularities according to Onsager's con-

jecture. It should be emphasized that the present results do not formally prove convergence of the discrete velocity fields $\boldsymbol{u}_h$ to a weak Euler solution $\boldsymbol{u}$. Even if convergence to a weak Euler solution was obtained for a subsequence h_j, another subsequence h_k might converge to a different weak Euler solution. Moreover, it can not be excluded that the observed "dissipation anomaly" is of numerical origin in the sense that the spatial resolutions considered here might be too coarse to adequately resolve an energy-conserving Euler solution. Nevertheless, the present study might complement theoretical and experimental works on anomalous energy dissipation and the related implications on finite-time singularities for three-dimensional incompressible Euler flows according to Onsager's conjecture. As part of future work, the so-called $4/5^{\text{th}}$-law (Duchon and Robert 2000) should be investigated numerically in order to substantiate the hypothesis of having found a dissipative anomaly. Finally, the present work would gain further theoretical support by a proof of convergence to generalized weak Euler solutions for DG discretization schemes of the Euler equations in the limit $h \to 0$ (along subsequences).

Given the challenges of the proposed indirect approach, it appears to be natural to raise the question whether this technique has advantages over well-known direct techniques. Below, some ideas are shared why the indirect approach might be an attractive technique for the exploration of finite-time singularities. A key motivation for the indirect approach is that it might not be necessary to resolve the smallest scales of the flow in order to resolve the temporal evolution of the kinetic energy. This is based on the observation that resolving the vorticity field (direct approach) typically requires significantly finer resolutions than resolving the kinetic energy (indirect approach). When studying singularities by the direct approach, one needs to realize that resolving the smallest scales is not possible for finite spatial resolutions. The vorticity will take finite[5] values at all times for a stable discretization scheme, so that results of a single simulation do not serve as numerical evidence of a finite-time singularity. A key element of the indirect approach is a mesh refinement study where the quantities of interest (kinetic energy and its dissipation rate) do not blow up. This circumstance is expected to support cross-solver validation very naturally. Assume for example that different spatial discretization schemes would converge to the same dissipative, numerical solution (measured by some error norms as done in the present work). A cross-solver validation of the kinetic energy evolution or dissipation rate (indirect approach) might give confidence that the obtained dissipative behavior is not artificial (why should entirely different PDE solvers produce "the same" artificial result with artificial dissipation?). On the contrary, it would be difficult for each of these simulations to demonstrate a blow-up of vorticity in order to provide evidence of finite-time singularities by direct techniques. All these solvers might show an increase in vorticity by a large factor, but it is still not clear whether a finite-time singularity $\|\boldsymbol{\omega}\|_\infty \to \infty$ occurs. Of course, there is currently not sufficient data available in the literature to substantiate this assumption regarding a cross-solver validation of the present results. One aspect that the present work wants to initiate is exactly such a cross-solver validation based on space-averaged results or other suitable statistical results. In summary, the argument in favor of the indirect approach would then be a clearer *indication* of finite-time singularities due to relaxed resolution requirements as compared to the direct approach.

Applicability of high-order DG discretizations to large-scale problems in turbulence research, for which spectral methods are currently the state-of-the-art due to their accuracy and computa-

[5] In this context, the term *finite* means that the quantity of interest takes values much smaller than the "infinite" value defined by the maximum floating point number that can be represented on a computer.

tional efficiency, has been demonstrated. Finally, the present work gives indications in terms of what a promising large-eddy simulation strategy might be, and contributes to the long-lasting and difficult discussion on explicit versus implicit sub-grid scale models. High-order discretizations that can be described as implicit LES have shown very promising results for moderate Reynolds number flows, but it is often argued that such techniques can be expected to finally need explicit sub-grid scale modeling once they are applied in the limit $\mathrm{Re} \to \infty$. The present work contributes to this discussion by investigating a high-order discontinuous Galerkin discretization without explicit model in the inviscid limit. Assuming that such an implicit approach gives physical results in the inviscid limit, e.g., consistent with Onsager's conjecture on anomalous energy dissipation, it makes one more confident that such a method is able to naturally account for more complex physical mechanisms in turbulence beyond K41 theory (Dubrulle 2019). Taking as an alternative an energy-conserving numerical method with explicit sub-grid scale model, the anomalous energy dissipation has to be realized by the sub-grid model. It could therefore be an interesting future research direction to take the results shown here for the inviscid Taylor–Green problem as a reference for further validation or to perform comparative studies between explicit and implicit LES techniques for the highest Reynolds number case, the inviscid limit.

8 Extension to moving meshes: arbitrary Lagrangian–Eulerian techniques

Chapter 2 dealing with discretization methods for the incompressible Navier–Stokes equations has assumed that the computational domain does not change over time. However, many problems of practical interest involve time-varying domains such as free-surface, multiphase, and fluid–structure interaction problems. A variety of numerical methods has been proposed to deal with moving domains, which can be classified into fitted and non-fitted methods. Another nomenclature widely used distinguishes between interface-tracking and interface-capturing methods. This chapter discusses discretization methods for the incompressible Navier–Stokes equations based on the arbitrary Lagrangian–Eulerian technique, an approach belonging to the first category of methods mentioned above. Representatives of the second category would be immersed boundary, cut, and level-set methods. In the context of fluid–structure interaction problems, an overview is given in the inter-methodological article by Wall et al. (2006). Hybrid techniques combining fitted and non-fitted methods have also been proposed, see for example Schott et al. (2019) and references therein. The goal of this chapter is to extend the methodology presented in Chapter 2 towards moving domains as a preparation for FSI problems that are the subject of Chapter 9. The content of this chapter is based on work that has already been published in Fehn et al. (2021a). First steps towards moving meshes for the present DG discretization have been made in Heinz (2019).

The outline of this chapter is as follows. Section 8.1 provides a short review of literature on ALE methods, summarizes previous contributions in the field of ALE-DG methods, and explains novel contributions of the present work. Section 8.2 briefly presents the model problem of the incompressible Navier–Stokes equations in ALE form. Aspects related to the temporal discretization are discussed in Section 8.3, where important peculiarities related to the imposition of boundary conditions for projection-type Navier–Stokes solvers are addressed in detail. The spatial discretization is subject of Section 8.4, where fulfillment of the so-called geometric conservation law is discussed as well as energy stability of the present stabilized DG formulation. The goal is not only to maintain optimal rates of convergence as well as appealing stability properties in the ALE case, but also to provide a comprehensive and simple formulation that can easily be included in existing flow solvers and yields computationally efficient solution algorithms. Detailed numerical results are presented in Section 8.5, and a summary is provided in Section 8.6.

8.1 Introduction

The arbitrary Lagrangian–Eulerian (ALE) continuum mechanics description is the basis of many methods to capture flow problems on deforming domains. A very prominent class of applications are fluid–structure interaction problems with moderate deformations of the structure, where moderate means that a mesh moving or mesh smoothing algorithm is able to handle the mesh deformation of the fluid mesh following the deformations imposed at the fluid–structure interface, as opposed to very large deformations and topological changes that require other, geometrically more flexible techniques.

ALE methods have a long tradition and have first been developed for finite difference methods in Hirt et al. (1974), see also Donea et al. (1982) for a review on the early development of this methodology and Donea et al. (2017) for a survey of ALE methods. They have later been developed for finite element discretizations of the compressible Navier–Stokes equations in Donea et al. (1977, 1982), of the incompressible Navier–Stokes equations using linear elements in Hughes et al. (1981) and spectral element discretizations in Beskok and Warburton (2001), Ho and Patera (1990), and also for finite volume discretizations, see for example Lesoinne and Farhat (1996). In the context of discontinuous Galerkin (DG) discretizations, this technique has first been applied to the compressible Navier–Stokes equations, being solved on the deforming domain (Lomtev et al. 1999, Mavriplis and Nastase 2011, Nguyen 2010), or solving transformed equations on a reference domain (Persson et al. 2009, Schnücke et al. 2018). For the incompressible Navier–Stokes equations, ALE-DG methods have first been presented in Ferrer and Willden (2012) and later in Wang et al. (2018). Since ALE-DG methods for incompressible flows are the central topic of this chapter, the state-of-the-art is discussed in more detail below.

An ALE method satisfying the geometric conservation law (GCL) (Thomas and Lombard 1979) is able to preserve a constant flow state on moving meshes. The GCL has been extensively discussed in the context of finite volume discretizations of the compressible Navier–Stokes equations, see for example the review article by Farhat and Geuzaine (2004) and references therein. Here, the reader is especially referred to the works by Étienne et al. (2009), Förster et al. (2006) addressing the solution of incompressible flow problems and being particularly relevant for the present work. Following the design described in Förster et al. (2006) one can easily construct incompressible flow solvers that automatically fulfill the geometric conservation law.

8.1.1 State-of-the-art

ALE formulations for the incompressible Navier–Stokes equations using DG discretizations have been presented in Ferrer and Willden (2012), Wang et al. (2018). Both methods are based on the dual splitting projection scheme proposed by Karniadakis et al. (1991), Orszag et al. (1986) and use a discontinuous Galerkin discretization of the velocity–pressure coupling terms without integration by parts as originally proposed in Hesthaven and Warburton (2007), that has been shown to be unstable for small time step sizes, see Fehn et al. (2017). Both works (Ferrer and Willden 2012, Wang et al. 2018) use equal-order polynomials for velocity and pressure, but it was shown in Fehn et al. (2017) that the dual splitting scheme per se is not inf–sup stable as is sometimes believed. Moreover, the works by Ferrer and Willden (2012), Wang et al. (2018) enforce the important aspect of mass conservation and energy stability discussed in Chapter 2.4

neither exactly nor weakly, so that the robustness of these methods remains unclear. Fulfillment of the geometric conservation law has neither been demonstrated theoretically nor by numerical experiments. Second-order convergence in time is shown in Ferrer and Willden (2012) for a rotating, non-deforming mesh, and a temporal convergence test revealing second-order accuracy is shown in Wang et al. (2018) for a FSI problem by comparing the error against the solution for the smallest time step size. It still needs to be investigated whether third-order accuracy can be achieved for the dual splitting scheme in the ALE case, which has for example been shown in Krank et al. (2017) for a DG method solving the Eulerian form of the equations.

In the field of hybridizable discontinuous Galerkin discretizations, ALE-based monolithic FSI solvers are presented in Sheldon et al. (2016) using an HDG discretization and in Neunteufel and Schöberl (2021) using an $H(\mathrm{div})$-conforming HDG discretization. An ALE-HDG scheme for moving meshes and two-phase flows is presented in Fu (2020). These works do also not demonstrate optimal convergence in time (and space) of the ALE formulation and fulfillment of the geometric conservation law. Since two-dimensional problems are solved, robustness for under-resolved turbulent flow problems remains unclear, even though robustness may be expected theoretically for the approach in Neunteufel and Schöberl (2021) due to $H(\mathrm{div})$-conformity and an exactly divergence-free velocity via a Piola transformation.

More sophisticated convergence tests are presented in Rhebergen and Cockburn (2012) for a space–time HDG method on deforming domains, which satisfies the geometric conservation law and achieves high-order accuracy in space and time. An energy-stable version of such a space–time HDG approach has been proposed recently in Horváth and Rhebergen (2019) achieved through a velocity field that is $H(\mathrm{div})$-conforming and exactly divergence-free.

8.1.2 Novel contributions of the present work

Open questions remain from previous works on ALE-(H)DG methods for the incompressible Navier–Stokes equations regarding their numerical properties in terms of stability (small time steps, inf–sup problem, and under-resolved turbulence or energy stability), fulfillment of the geometric conservation law, and temporal convergence rates. Schemes fulfilling these properties have so far only been presented for (space–time) HDG discretizations. However, the use of matrix-based implementations renders these methods computationally expensive, especially for three-dimensional problems. A recent study by Kronbichler and Wall (2018) has shown that the computational efficiency of matrix-based HDG solvers significantly trails behind fast matrix-free DG implementations for tensor-product elements (Kronbichler and Kormann 2019) on modern CPU hardware, due to the large amount of data that needs to be transferred through the memory hierarchy of the CPU, see also Chapter 4. It has long been internalized by the HPC community that matrix-free algorithms are the method of choice for performant PDE solvers, independently of the type of discretization used. In the context of ALE formulations, the geometry of the problem and, hence, the metric terms in the weak form change continuously over time, so that the expensive step of assembling the local matrices and subsequently removing inner degrees of freedom by condensation in case of HDG methods has to be done in each time step. The use of space–time approaches can be expected to increase computational costs compared to fast projection-type solution techniques for incompressible flows. With a view to the current state-of-the-art, these reasons manifest themselves in a lack of fast solvers for HDG-type discretizations

for incompressible flows achieving a performance similar to matrix-free DG implementations published in Fehn et al. (2018a) and discussed in Chapter 6.

The main novelty of this chapter is to develop ALE-DG methods that combine (i) fast Navier–Stokes solution algorithms based on the method-of-lines approach (Fehn et al. 2017, 2018b), (ii) desirable discretization properties in terms of optimal convergence rates in time and space, the geometric conservation law, and robustness for turbulent flows, with (iii) computationally efficient matrix-free DG implementations (Kronbichler and Kormann 2019). As an alternative to space–time formulations, fulfillment of the GCL and discretization methods that formally exhibit arbitrarily high order of accuracy in space can also be achieved with a classical method-of-lines approach. Although high-order accuracy can be observed for simple analytical test cases with smooth solution, it should be emphasized that such a theoretically optimal convergence behavior is rarely observed in complex applications. Many engineering problems of practical interest are turbulent in nature, and in this field most applications are operating in the pre-asymptotic regime where resolving spatial scales of the flow prior to entering the asymptotic regime is already a challenge from the point of view of computational costs. In such a setting, it would be unrealistic to expect an improvement by orders of magnitude from high-order discretizations. Instead, a main motivation for the use of high-order DG discretizations actually stems from their dissipation/dispersion properties, where improved resolution capabilities per degree of freedom have been reported for high-order methods (Gassner and Beck 2013, Moura et al. 2017a). Furthermore, such an approach is also attractive since robustness and good accuracy can be achieved for under-resolved turbulence without the use of explicit, often parameter-dependent turbulence models (Fehn et al. 2018b, 2019a). This motivates to apply this type of discretization also to problems on moving domains. Following the methodology in Chapter 2, this work does not strive for arbitrarily high order of accuracy in time, but relies on the pragmatism that second or third-order time integration schemes are sufficient in terms of accuracy for practical problems, especially if the time step size is restricted according to the CFL condition when treating the convective term explicitly in time. This solution technique is for example used by some of the most sophisticated and computationally efficient high-order CFD solvers, such as `Nektar++`(Cantwell et al. 2015) and `Nek5000` (Fischer et al. 2020a).

The ALE-DG methodology proposed here is characterized by the following design choices: The ALE equations are solved on the deformed geometry, storing one instance of the mesh that is updated from one time step to the next. Since high-order parametric mappings are considered, this implies updating the coordinates of all nodal points. The ALE equations are introduced on the level of differential equations, subsequently discretized in time and space. This way, it is straight-forward to satisfy the geometric conservation law automatically (Förster et al. 2006), independently of the mesh motion and its numerical approximation. Although this chapter considers analytical mesh motions, the methods are formulated with fluid–structure interaction problems in mind, i.e., the ALE formulations are implemented in a way that they only require knowledge about the coordinates of all grid nodes at discrete instants of time. Then, the grid velocity is computed from these grid coordinates in a way that the formal order of accuracy of the time integration schemes is maintained on moving meshes. The fact that metric terms in the weak form change over time does not introduce additional complexity, since this is an aspect that a generic finite element library (such as `deal.II` (Arndt et al. 2020a) used here) can take care of automatically. Hence, the present design allows to reuse existing DG implementations for static meshes with modifications stemming only from the additional ALE transport term.

This is in contrast to other ALE-DG formulations that transform the equations to a reference frame accounting for metric terms prior to discretization. The ALE-DG methods discussed here are embedded into the unified framework of Navier–Stokes solvers presented in Chapter 2, supporting both monolithic solvers and widely used projection-type solvers, implicit and explicit formulations of the convective term, as well as the option for adaptive time-stepping.

8.2 Incompressible Navier–Stokes equations in ALE formulation

This section briefly derives the ALE form of the incompressible Navier–Stokes equations and the reader is referred to Donea and Huerta (2003) for more detailed information. To derive the incompressible Navier–Stokes equation in arbitrary Lagrangian–Eulerian form, consider the Eulerian formulation of the incompressible Navier–Stokes equations in a domain $\Omega \subset \mathbb{R}^d$

$$\left.\frac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{x}} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \nabla\cdot\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) + \nabla p = \boldsymbol{f}\ , \tag{8.1}$$

$$\nabla\cdot\boldsymbol{u} = 0\ . \tag{8.2}$$

As introduced in Chapter 2, $\boldsymbol{u} = (u_1, ..., u_d)^{\mathsf{T}}$ is the velocity vector, p the kinematic pressure, and $\boldsymbol{f} = (f_1, ..., f_d)^{\mathsf{T}}$ the body force vector. Spatial derivatives are defined w.r.t. the Eulerian coordinates $\boldsymbol{x} = (x_1, ..., x_d)^{\mathsf{T}}$, $\nabla(\cdot) = \frac{\partial(\cdot)}{\partial \boldsymbol{x}}$, and $|_{\boldsymbol{x}}$ denotes the time derivative at a fixed point $\boldsymbol{x}$ in Eulerian coordinates. The convective term is written in convective formulation $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$ instead of the conservative formulation $\nabla\cdot(\boldsymbol{u}\otimes\boldsymbol{u})$, since the convective formulation is a natural form in the context of ALE, see Remark 8.2.

To obtain the ALE form of the above equations, the Eulerian time derivative is replaced by a time derivative with respect to a fixed point of the mesh (denoted as ALE time derivative in the following), which gives rise to an additional transport term with transport by the grid velocity $\boldsymbol{u}_{\mathrm{G}}$. The convective term then has the same structure as the convective term in Eulerian description (using the convective formulation), but with transport velocity $\boldsymbol{w} = \boldsymbol{u} - \boldsymbol{u}_{\mathrm{G}}$ instead of the fluid velocity $\boldsymbol{u}$. The motivation behind is to apply the time integration scheme with an update of the solution vectors just as in the Eulerian case, thereby automatically obtaining the solution coefficients on the new mesh. By this technique, expensive transformations of the solution vector (containing the degrees of freedom of the finite element expansion) from one mesh at a previous time instant onto another one at the current time instant is avoided. Apart from the Eulerian coordinates $\boldsymbol{x}$ describing an (arbitrary) point in Euclidean space, the mesh coordinates $\boldsymbol{\chi}$ are introduced describing a fixed point of the mesh. For transient problems, $\boldsymbol{x}(\boldsymbol{\chi}, t)$ describes the trajectory of a fixed point of the mesh in the Eulerian coordinates $\boldsymbol{x}$ as a function of time. The grid velocity is therefore given as

$$\boldsymbol{u}_{\mathrm{G}} = \left.\frac{\partial \boldsymbol{x}}{\partial t}\right|_{\boldsymbol{\chi}}\ . \tag{8.3}$$

The time derivative of an arbitrary quantity ϕ w.r.t. the mesh reference frame $\boldsymbol{\chi}$ gives the desired relation between the Eulerian and ALE time derivatives

$$\left.\frac{\partial \phi\left(\boldsymbol{x}(\boldsymbol{\chi},t),t\right)}{\partial t}\right|_{\boldsymbol{\chi}} = \left.\frac{\partial \phi}{\partial t}\right|_{\boldsymbol{x}} + \frac{\partial \phi}{\partial \boldsymbol{x}} \cdot \underbrace{\left.\frac{\partial \boldsymbol{x}}{\partial t}\right|_{\boldsymbol{\chi}}}_{=\boldsymbol{u}_{\mathrm{G}}} = \left.\frac{\partial \phi}{\partial t}\right|_{\boldsymbol{x}} + \left(\boldsymbol{u}_{\mathrm{G}} \cdot \nabla\right) \phi \, . \tag{8.4}$$

Using equation (8.4) to substitute the Eulerian time derivative in equation (8.1), one arrives at the incompressible Navier–Stokes equations in ALE formulation

$$\left.\frac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{\chi}} + \left((\boldsymbol{u}-\boldsymbol{u}_{\mathrm{G}}) \cdot \nabla\right)\boldsymbol{u} - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}) + \nabla p = \boldsymbol{f} \;\; \text{in } \Omega(t) \times [0,T] \;\; , \tag{8.5}$$

$$\nabla \cdot \boldsymbol{u} = 0 \;\; \text{in } \Omega(t) \times [0,T] \;\; . \tag{8.6}$$

As described in detail in Section 2.2, initial conditions are prescribed in $\Omega(t=0)$ and boundary conditions on $\Gamma(t) = \Gamma^{\mathrm{D}}(t) \cup \Gamma^{\mathrm{N}}(t)$ with $\Gamma^{\mathrm{D}}(t) \cap \Gamma^{\mathrm{N}}(t) = \emptyset$, with the difference that $\Gamma^{\mathrm{D}}(t), \Gamma^{\mathrm{N}}(t)$ are time dependent in general.

Remark 8.1 *The term arbitrary Lagrangian–Eulerian can be explained by the fact that the Lagrangian description is recovered for $\boldsymbol{\chi} = \boldsymbol{X}$ (material coordinates) yielding $\boldsymbol{u}_{\mathrm{G}} = \boldsymbol{u}$, and the Eulerian description for $\boldsymbol{\chi} = \boldsymbol{x}$ yielding $\boldsymbol{u}_{\mathrm{G}} = \boldsymbol{0}$.*

Remark 8.2 *Note that the formulation chosen as a starting point for discretization in time and space has important implications regarding compliance with the geometric conservation law. Using the above differential formulation with the convective term written in non-conservative form to derive temporal and spatial discretizations allows to satisfy the GCL automatically (Förster et al. 2006). Alternative conservative formulations with time derivative in front of the integral over a temporally changing domain contain an additional term in which the divergence of the mesh velocity occurs, and fulfilling the GCL is more complicated in this case (Étienne et al. 2009, Förster et al. 2006). The study by Étienne et al. (2009) is inconclusive in the sense that the non-conservative formulation is not restricted to first-order accuracy in time as implied in that work. As demonstrated in Förster et al. (2006) and in the present work, high-order accuracy in time can be achieved with the non-conservative formulation.*

The motion of the domain $\Omega(t)$ is described by a function $\boldsymbol{f}_{\mathrm{G}} = \boldsymbol{f}_{\mathrm{G}}(\boldsymbol{\chi}, t)$

$$\boldsymbol{f}_{\mathrm{G}} : \begin{cases} \Omega_0 \times [0,T] \rightarrow \Omega(t), \; \Omega_0, \Omega(t) \subset \mathbb{R}^d \, , \\ (\boldsymbol{\chi}, t) \mapsto \boldsymbol{x}\left(\boldsymbol{\chi}, t\right) \, . \end{cases} \tag{8.7}$$

With respect to the argument $\boldsymbol{\chi}$, the map $\boldsymbol{f}_{\mathrm{G}}$ is a homeomorphism for all times, and the argument t describes a continuous deformation over time. In this chapter, $\boldsymbol{f}_{\mathrm{G}}$ will be an analytically defined, smooth function in space and time. An illustration is shown in Figure 8.1. Without loss of generality, assume that $\boldsymbol{x}\left(\boldsymbol{\chi}, t=0\right) = \boldsymbol{\chi}$, and therefore $\Omega(t=0) = \Omega_0$. To demonstrate high order of accuracy of the multistep BDF time integration schemes that are used in this work and that require a starting procedure to demonstrate the formal order of accuracy, it is essential that the mesh motion is continuously differentiable in time. In the context of fluid–structure interaction, the mesh motion is defined by the deformation of the fluid–structure interface according to the structural displacements and a mesh smoothing algorithm calculating the mesh deformation in the interior of the fluid domain.

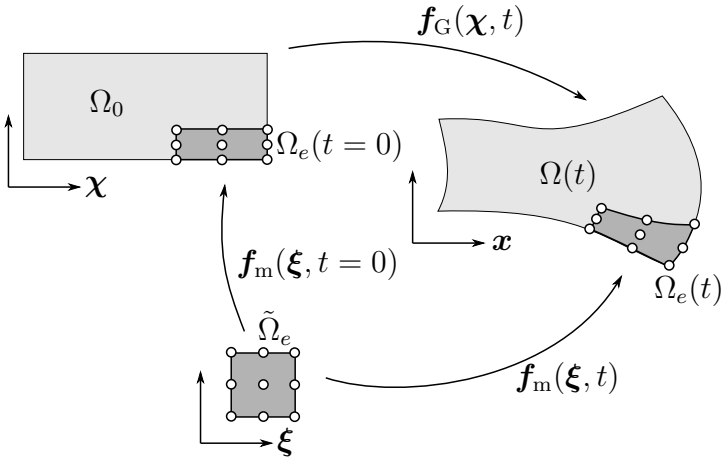


Figure 8.1: Illustration of coordinate systems $\boldsymbol{\chi}$ and $\boldsymbol{x}$ with mesh deformation $\boldsymbol{f}_\mathrm{G}$, and reference coordinates $\boldsymbol{\xi}$ with finite element mapping $\boldsymbol{f}_\mathrm{m}$ of polynomial degree $k_\mathrm{m} = 2$.

8.3 Discretization in time

This section builds upon Section 2.3, where different solution techniques are discussed for the incompressible Navier–Stokes equations in Eulerian form, including monolithic solvers and operator splitting techniques. The goal is to derive consistent ALE formulations maintaining the high-order accuracy of these schemes also on moving meshes. Hence, the focus of this section is on the aspects relevant to ALE and in particular boundary conditions, while the reader is referred to Section 2.3 for an introduction of these time integration schemes and the related notation.

8.3.1 Coupled solution approach

Applying the BDF time integration scheme to equations (8.5) and (8.6) and using a fully implicit formulation yields

$$\left.\frac{\gamma_0^n \boldsymbol{u}^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n}\right|_{\boldsymbol{\chi}} + \left(\left(\boldsymbol{u}^{n+1} - \boldsymbol{u}_\mathrm{G}^{n+1}\right) \cdot \nabla\right) \boldsymbol{u}^{n+1} \\ -\nabla \cdot \boldsymbol{F}_\mathrm{v}(\boldsymbol{u}^{n+1}) + \nabla p^{n+1} = \boldsymbol{f}\left(t_{n+1}\right) \ , \tag{8.8}$$

$$\nabla \cdot \boldsymbol{u}^{n+1} = 0 \ , \tag{8.9}$$

where $|_{\boldsymbol{\chi}}$ means that all terms of the BDF sum are evaluated at constant $\boldsymbol{\chi}$, i.e., an ALE-type time derivative has to be considered. The boundary conditions are

$$\boldsymbol{u}^{n+1} = \boldsymbol{g}_u^{n+1} \ \text{ on } \Gamma^\mathrm{D} \ , \tag{8.10}$$

$$\left(\boldsymbol{F}_\mathrm{v}(\boldsymbol{u}^{n+1}) - p^{n+1}\boldsymbol{I}\right) \cdot \boldsymbol{n} = \boldsymbol{h}^{n+1} \ \text{ on } \Gamma^\mathrm{N} \ . \tag{8.11}$$

As an alternative formulation, an explicit formulation of convective term is considered, discretized in time via an extrapolation scheme of order J

$$\begin{aligned}\left.\frac{\gamma_0^n \boldsymbol{u}^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n}\right|_{\boldsymbol{\chi}} + \sum_{i=0}^{J-1} \beta_i^n \left(\left(\boldsymbol{u}^{n-i} - \boldsymbol{u}_{\mathrm{G}}^{n+1}\right) \cdot \nabla\right) \boldsymbol{u}^{n-i} \\ -\nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{n+1}) + \nabla p^{n+1} = \boldsymbol{f}\left(t_{n+1}\right) \ ,\end{aligned} \tag{8.12}$$

where the following boundary condition is used in the convective term

$$\boldsymbol{u}^{n-i} = \boldsymbol{g}_u^{n+1} \ \text{ on } \Gamma^{\mathrm{D}} \ . \tag{8.13}$$

For an explicit treatment of the convective term, the time step size is restricted according to the CFL condition, equations (2.209) and (2.210), with the difference that the velocity $\boldsymbol{u}_h$ has to be replaced by the transport velocity $\boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{G},h}$ in the ALE case.

Remark 8.3 *While the boundary condition $\boldsymbol{u}^{n-i} = \boldsymbol{g}_u^{n-i}$ might be considered a natural formulation as well, it is interesting to study whether equation* (8.13) *also preserves optimal rates of convergence. The advantage of this formulation is that only one version of the boundary condition is required at any one time of the solution of the transient problem, therefore easing implementation without the need to store and keep track of previous versions of the boundary condition for fluid–structure interaction problems. For the projection methods discussed below, one can not fully maintain this goal as these methods are more involved regarding the formulation of boundary conditions.*

8.3.2 Dual splitting scheme

In ALE formulation, the high-order dual splitting scheme (Karniadakis et al. 1991) consists of the following four sub-steps to be solved in each time step

$$\begin{aligned}\left.\frac{\gamma_0^n \hat{\boldsymbol{u}} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}^{n-i}}{\Delta t_n}\right|_{\boldsymbol{\chi}} = \\ -\sum_{i=0}^{J-1} \beta_i^n \left(\left(\boldsymbol{u}^{n-i} - \boldsymbol{u}_{\mathrm{G}}^{n+1}\right) \cdot \nabla\right) \boldsymbol{u}^{n-i} + \boldsymbol{f}\left(t_{n+1}\right), \\ \boldsymbol{u}^{n-i} = \boldsymbol{g}_u^{n+1} \quad \text{on } \Gamma^{\mathrm{D}},\end{aligned} \tag{8.14}$$

$$\begin{aligned}-\nabla^2 p^{n+1} = -\frac{\gamma_0^n}{\Delta t_n} \nabla \cdot \hat{\boldsymbol{u}}, \\ \nabla p^{n+1} \cdot \boldsymbol{n} = h_p^{n+1} \quad \text{on } \Gamma^{\mathrm{D}}, \\ p^{n+1} = g_p^{n+1} \quad \text{on } \Gamma^{\mathrm{N}}, \\ \hat{\boldsymbol{u}} = \boldsymbol{g}_{\hat{u}}^{n+1} \quad \text{on } \Gamma^{\mathrm{D}},\end{aligned} \tag{8.15}$$

$$\begin{aligned} \hat{\hat{\boldsymbol{u}}} &= \hat{\boldsymbol{u}} - \frac{\Delta t_n}{\gamma_0^n} \nabla p^{n+1}, \\ p^{n+1} &= g_p^{n+1} \quad \text{on } \Gamma^{\mathrm{N}}, \end{aligned} \tag{8.16}$$

$$\begin{aligned} \frac{\gamma_0^n}{\Delta t_n} \boldsymbol{u}^{n+1} - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}\left(\boldsymbol{u}^{n+1}\right) &= \frac{\gamma_0^n}{\Delta t_n} \hat{\hat{\boldsymbol{u}}}, \\ \boldsymbol{u}^{n+1} &= \boldsymbol{g}_u^{n+1} \quad \text{on } \Gamma^{\mathrm{D}}, \\ \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{n+1}) \cdot \boldsymbol{n} &= \boldsymbol{h}_u^{n+1} \quad \text{on } \Gamma^{\mathrm{N}}. \end{aligned} \tag{8.17}$$

The pressure Neumann boundary condition h_p and the Dirichlet boundary condition $\boldsymbol{g}_{\hat{u}}$ for the intermediate velocity need to be described in ALE formulation as well. For a derivation of these boundary conditions, the reader is referred to Section 2.3.3. In case of the ALE form of the incompressible Navier–Stokes equations, the consistent Neumann boundary condition h_p for the pressure reads

$$\begin{aligned} h_p\left(t_{n+1}\right) = &- \left[\left. \frac{\gamma_0^n \boldsymbol{g}_u^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{g}_u^{n-i}}{\Delta t_n} \right|_{\boldsymbol{\chi}} - \boldsymbol{f}\left(t_{n+1}\right) \right] \cdot \boldsymbol{n}^{n+1} \\ &- \left[\sum_{i=0}^{J_p-1} \beta_i^n \left(\left(\left(\boldsymbol{u}^{n-i} - \boldsymbol{u}_{\mathrm{G}}^{n+1} \right) \cdot \nabla \right) \boldsymbol{u}^{n-i} + \nu \nabla \times \boldsymbol{\omega}^{n-i} \right) \right] \cdot \boldsymbol{n}^{n+1} \, . \end{aligned} \tag{8.18}$$

As already noted in Section 2.3.3, the acceleration term with exact derivative $\partial \boldsymbol{g}_u / \partial t$ is replaced by a discrete BDF time derivative in the ALE or fluid–structure interaction case where the boundary condition is only known at discrete times. Hence, one has to record the history of Dirichlet boundary values $\boldsymbol{g}_u$ in case of the dual splitting scheme. Since the time derivative is of ALE-type at constant $\boldsymbol{\chi}$, the convective term needs to be formulated in ALE form as well.

The velocity Dirichlet boundary condition for the intermediate velocity $\hat{\boldsymbol{u}}$, equation (2.42) for the Eulerian case, reads in ALE form

$$\boldsymbol{g}_{\hat{u}}\left(\boldsymbol{\chi}, t_{n+1}\right) = \sum_{i=0}^{J-1} \frac{\alpha_i^n}{\gamma_0^n} \boldsymbol{g}_u(\boldsymbol{\chi}, t_{n-i}) - \frac{\Delta t_n}{\gamma_0^n} \sum_{i=0}^{J-1} \beta_i^n \left(\left(\boldsymbol{u}^{n-i} - \boldsymbol{u}_{\mathrm{G}}^{n+1} \right) \cdot \nabla \right) \boldsymbol{u}^{n-i} + \frac{\Delta t_n}{\gamma_0^n} \boldsymbol{f}\left(t_{n+1}\right) \, . \tag{8.19}$$

As indicated in the above equation, the history of the boundary condition $\boldsymbol{g}_u$ is evaluated in grid coordinates $\boldsymbol{\chi}$ following the moving mesh from one time instant to the next, so that the ALE form of the convective term is required in this boundary condition similar to the pressure Neumann boundary condition (8.18).

8.3.3 Pressure-correction scheme

Extending the formulation of pressure-correction schemes from Section 2.3.4 to the ALE formulation of the incompressible Navier–Stokes equations, the pressure-correction schemes can

be summarized as follows

$$\begin{aligned}\left.\frac{\gamma_0^n\hat{\boldsymbol{u}}-\sum_{i=0}^{J-1}\alpha_i^n\boldsymbol{u}^{n-i}}{\Delta t_n}\right|_{\boldsymbol{\chi}}+\left(\left(\hat{\boldsymbol{u}}-\boldsymbol{u}_{\mathrm{G}}^{n+1}\right)\cdot\nabla\right)\hat{\boldsymbol{u}}-\nabla\cdot\boldsymbol{F}_{\mathrm{v}}\left(\hat{\boldsymbol{u}}\right)=\\ -\sum_{i=0}^{J_p-1}\beta_i^n\nabla p^{n-i}+\boldsymbol{f}\left(t_{n+1}\right),\\ \hat{\boldsymbol{u}}=\boldsymbol{g}_u^{n+1}\quad\text{on }\Gamma^{\mathrm{D}},\\ \boldsymbol{F}_{\mathrm{v}}(\hat{\boldsymbol{u}})\cdot\boldsymbol{n}=\boldsymbol{h}_u^{n+1}\quad\text{on }\Gamma^{\mathrm{N}},\\ p^{n-i}=g_p^{n-i}\quad\text{on }\Gamma^{\mathrm{N}},\end{aligned}\tag{8.20}$$

$$\begin{aligned}-\nabla^2\phi^{n+1}=-\frac{\gamma_0^n}{\Delta t_n}\nabla\cdot\hat{\boldsymbol{u}},\\ \nabla\phi^{n+1}\cdot\boldsymbol{n}=h_\phi^{n+1}=0\quad\text{on }\Gamma^{\mathrm{D}},\\ \phi^{n+1}=g_\phi^{n+1}\quad\text{on }\Gamma^{\mathrm{N}},\\ \hat{\boldsymbol{u}}=\boldsymbol{g}_u^{n+1}\quad\text{on }\Gamma^{\mathrm{D}},\end{aligned}\tag{8.21}$$

$$\begin{aligned}p^{n+1}=\phi^{n+1}+\sum_{i=0}^{J_p-1}\beta_i^n p^{n-i}-\chi\nu\nabla\cdot\hat{\boldsymbol{u}},\\ \hat{\boldsymbol{u}}=\boldsymbol{g}_u^{n+1}\quad\text{on }\Gamma^{\mathrm{D}},\end{aligned}\tag{8.22}$$

$$\begin{aligned}\boldsymbol{u}^{n+1}=\hat{\boldsymbol{u}}-\frac{\Delta t_n}{\gamma_0^n}\nabla\phi^{n+1},\\ \phi^{n+1}=g_\phi^{n+1}\quad\text{on }\Gamma^{\mathrm{N}}.\end{aligned}\tag{8.23}$$

In the above equations, the convective term is formulated implicitly. As for the monolithic solver, an alternative formulation with explicit treatment of the convective term can be used, resulting in the following momentum equation in the first sub-step

$$\begin{aligned}\left.\frac{\gamma_0^n\hat{\boldsymbol{u}}-\sum_{i=0}^{J-1}\alpha_i^n\boldsymbol{u}^{n-i}}{\Delta t_n}\right|_{\boldsymbol{\chi}}-\nabla\cdot\boldsymbol{F}_{\mathrm{v}}\left(\hat{\boldsymbol{u}}\right)=-\sum_{i=0}^{J-1}\beta_i^n\left(\left(\boldsymbol{u}^{n-i}-\boldsymbol{u}_{\mathrm{G}}^{n+1}\right)\cdot\nabla\right)\boldsymbol{u}^{n-i}\\ -\sum_{i=0}^{J_p-1}\beta_i^n\nabla p^{n-i}+\boldsymbol{f}\left(t_{n+1}\right)\ ,\end{aligned}\tag{8.24}$$

where the following boundary condition is imposed for the convective term

$$\boldsymbol{u}^{n-i}=\boldsymbol{g}_u^{n+1}\ \text{ on }\Gamma^{\mathrm{D}}\ .\tag{8.25}$$

The pressure Poisson equation (8.21) is subject to the pressure Dirichlet boundary condition

$$g_\phi(\boldsymbol{\chi},t_{n+1})=g_p\left(\boldsymbol{\chi},t_{n+1}\right)-\sum_{i=0}^{J_p-1}\beta_i^n g_p\left(\boldsymbol{\chi},t_{n-i}\right)\ .\tag{8.26}$$

Remark 8.4 *Note that $p^{n-i} = g_p^{n-i}$ is prescribed in equation* (8.20) *to be consistent with boundary condition* (8.26)*. Otherwise, sub-optimal rates of convergence have been observed when prescribing $p^{n-i} = g_p^{n+1}$. Hence, a history of pressure Dirichlet boundary values on Γ^{N} has to be stored for the pressure-correction scheme for higher-order schemes with $J_p \geq 1$, which can be seen as a consequence of the operator splitting as compared to the monolithic solver described in Section 8.3.1.*

Remark 8.5 *ALE formulations as described here for the incompressible Navier–Stokes equations have also been implemented for the scalar transport solver and the coupled flow–transport problems described in Chapter 3. A detailed description is omitted here since the procedure is very similar to the incompressible Navier–Stokes solvers described above. In the ALE case, the convective term in the scalar transport equation is written in convective form. Then, the convective term in equations* (3.8) *and* (3.9) *becomes $(\boldsymbol{w}^{n+1} \cdot \nabla)\, \theta^{n+1}$ and $\sum_{i=0}^{J-1} \beta_i^n \, (\boldsymbol{w}^{n+1} \cdot \nabla)\, \theta^{n-i}$ with $\boldsymbol{w} = \boldsymbol{u} - \boldsymbol{u}_{\mathrm{G}}$, respectively, if an arbitrary Lagrangian–Eulerian formulation is used.*

8.4 Discretization in space

8.4.1 Time-dependent mapping and grid velocity

This work chooses a formulation of ALE-DG methods that solves the equations on the deformed geometry. In the finite element context, the deformation of the geometry is described by the high-order mapping $\boldsymbol{x}^e(\boldsymbol{\xi}, t) : \tilde{\Omega}_e \times [0, T] \rightarrow \Omega_e(t)$ from reference space to physical space

$$\boldsymbol{f}_{\mathrm{m}}^e : \begin{cases} \tilde{\Omega}_e \times [0,T] \rightarrow \Omega_e(t),\, \tilde{\Omega}_e = [0,1]^d, \Omega_e(t) \subset \mathbb{R}^d, \\ (\boldsymbol{\xi}, t) \mapsto \boldsymbol{x}^e\,(\boldsymbol{\xi}, t)\ . \end{cases} \tag{8.27}$$

Introducing this time dependency into equation (2.71) yields

$$\boldsymbol{x}^e(\boldsymbol{\xi}, t) = \sum_{i_1,\ldots,i_d=0}^{k_{\mathrm{m}}} \ell_{i_1\ldots i_d}^{k_{\mathrm{m}}}(\boldsymbol{\xi}) \boldsymbol{x}_{i_1\ldots i_d}^e(t)\ . \tag{8.28}$$

The mapping can be seen in analogy to the function $\boldsymbol{f}_{\mathrm{G}}$ describing the topological changes of the domain Ω. While $\boldsymbol{f}_{\mathrm{G}}$ is defined globally for the whole domain and in a spatially continuous way, the finite element mapping describes the mesh motion for each element of the mesh in the discrete setting and is of finite dimension. For the following derivations, it is important to realize that a point with constant $\boldsymbol{\chi}$ can be thought of as a point with fixed $\boldsymbol{\xi}$ coordinates within one element, i.e., there exists a bijective map between $\boldsymbol{\chi}$ and $\boldsymbol{\xi}$ for each element. An illustration is given in Figure 8.1. The present work is restricted to problems for which the topology of the mesh does not change, i.e., that do not require remeshing. Hence, the data structures do not have to be adjusted dynamically when moving the mesh. Updating the mesh means updating the $d(k_{\mathrm{m}}+1)^d$ mapping degrees of freedom per element, i.e.,

$$\boldsymbol{x}_{i_1\ldots i_d}^e(t_{n+1}) = \boldsymbol{f}_{\mathrm{G}}(\boldsymbol{x}_{i_1\ldots i_d}^e(t=0), t_{n+1})\ . \tag{8.29}$$

Moderately large deformations are possible as long as $\boldsymbol{f}_{\mathrm{G}}$ remains invertible, and invalid elements with invalid mapping or Jacobian will otherwise occur in the discrete setting. The numerical examples shown in this chapter use a high-order mapping with $k_{\mathrm{m}} = k_u$.

Due to the ansatz (2.68) with a separation of space and time, applying the temporal discretization to the spatially discretized equations in ALE form becomes trivial in the sense that the structure of the equations is equivalent to the Eulerian case. Consider the time derivative term in equation (8.5) multiplied by test functions $\boldsymbol{v}_h$, integrated over element $\Omega_e(t = t_{n+1})$, and to be discretized in time

$$
\begin{aligned}
\left(\boldsymbol{v}_h, \left. \frac{\partial \boldsymbol{u}_h(t)}{\partial t} \right|_{\boldsymbol{\chi}} \right)_{\Omega_e^{n+1}} &= \left(\boldsymbol{v}_h, \left. \frac{\partial \sum_{i_1,\ldots,i_d=0}^{k_u} \ell_{i_1\ldots i_d}^{k_u}(\boldsymbol{\xi}) \boldsymbol{u}_{i_1\ldots i_d}^{e}(t)}{\partial t} \right|_{\boldsymbol{\xi}} \right)_{\Omega_e^{n+1}} \\
&= \left(\boldsymbol{v}_h, \sum_{i_1,\ldots,i_d=0}^{k_u} \ell_{i_1\ldots i_d}^{k_u}(\boldsymbol{\xi}) \frac{\partial \boldsymbol{u}_{i_1\ldots i_d}^{e}(t)}{\partial t} \right)_{\Omega_e^{n+1}} \\
&\approx \left(\boldsymbol{v}_h, \sum_{i_1,\ldots,i_d=0}^{k_u} \ell_{i_1\ldots i_d}^{k_u}(\boldsymbol{\xi}) \frac{\gamma_0^n \boldsymbol{u}_{i_1\ldots i_d}^{e,n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}_{i_1\ldots i_d}^{e,n-i}}{\Delta t_n} \right)_{\Omega_e^{n+1}} \\
&= \left(\boldsymbol{v}_h, \left. \frac{\gamma_0^n \boldsymbol{u}_h^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}_h^{n-i}}{\Delta t_n} \right|_{\boldsymbol{\xi}} \right)_{\Omega_e^{n+1}} .
\end{aligned}
\tag{8.30}
$$

The BDF rule introduced in the third row of the above equation approximates the acceleration at time t_{n+1} consistently with the integral over Ω_e taken at the same instant of time. In the following, the label $|_{\boldsymbol{\xi}}$ is skipped for simplicity, as it is clear from the above derivation that the BDF rule is simply applied to the global solution vector containing the unknown degrees of freedom and that all terms of the BDF sum use the same mass matrix at time t_{n+1}. As explained in more detail in Section 8.4.5, using the same mass matrix for all solution vectors is important in order to satisfy the geometric conservation law (Förster et al. 2006). The above equation highlights that discretization in space and time commute, meaning that the last term of the above equation would have also been obtained by discretizing equation (8.8) in space. The projection-type solution methods considered in this work are already formulated in a time-discrete manner, since the splitting is performed on the level of differential operators. For this reason, the derivation of DG formulations shown in the following starts from the time-discrete problems stated in Section 8.3.

Following Förster et al. (2006), the grid velocity is computed in the same way via a BDF time derivative of the nodal grid coordinates $\boldsymbol{x}_{i_1\ldots i_d}^{e}$ in the time discrete case in order to achieve high-order temporal convergence on moving meshes

$$
\boldsymbol{u}_{\mathrm{G},h}^{n+1} = \left. \frac{\partial \boldsymbol{x}_h}{\partial t}(t_{n+1}) \right|_{\boldsymbol{\chi}} \approx \left. \frac{\gamma_0^n \boldsymbol{x}_h^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{x}_h^{n-i}}{\Delta t_n} \right|_{\boldsymbol{\xi}} . \tag{8.31}
$$

This procedure is different from Beskok and Warburton (2001), Ho and Patera (1990), Hughes et al. (1981) where the grid coordinates are updated by integrating the mesh velocity forward in time.

8.4.2 Coupled solution approach

Beginning with the monolithic solution approach, the weak discontinuous Galerkin formulation of the fully discrete problem with implicit formulation of the convective term can be summarized as follows: Find $\boldsymbol{u}_h^{n+1} \in \mathcal{V}_h^u$, $p_h^{n+1} \in \mathcal{V}_h^p$ such that

$$\begin{aligned} m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \frac{\gamma_0^n \boldsymbol{u}_h^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}_h^{n-i}}{\Delta t_n}\right) + c_h^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}, \boldsymbol{u}_{\mathrm{G},h}^{n+1}; \boldsymbol{g}_u^{n+1}\right) \\ + v_h^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}; \boldsymbol{g}_u^{n+1}, \boldsymbol{h}_u^{n+1}\right) + g_h^{e,n+1}\left(\boldsymbol{v}_h, p_h^{n+1}; g_p^{n+1}\right) \\ + a_{\mathrm{D},h}^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}\right) + a_{\mathrm{C},h}^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}; \boldsymbol{g}_u^{n+1}\right) - b_h^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{f}(t_{n+1})\right) = 0 \,, \qquad (8.32) \\ - d_h^{e,n+1}\left(q_h, \boldsymbol{u}_h^{n+1}; \boldsymbol{g}_u^{n+1}\right) = 0 \,, \qquad (8.33) \end{aligned}$$

for all $(\boldsymbol{v}_h, q_h) \in \mathcal{V}_{h,e}^u \times \mathcal{V}_{h,e}^p$ and for all elements $e = 1, ..., N_{\mathrm{el}}$. The time label $n+1$, e.g. in $m_{h,u}^{e,n+1}$, indicates that the integral is evaluated on the domain Ω_e^{n+1}. When formulating the convective term explicitly, the convective term in the discretized momentum equation (8.32) is replaced by

$$c_h^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}, \boldsymbol{u}_{\mathrm{G},h}^{n+1}; \boldsymbol{g}_u^{n+1}\right) \rightarrow \sum_{i=0}^{J-1} \beta_i^n c_h^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n-i}, \boldsymbol{u}_{\mathrm{G},h}^{n+1}; \boldsymbol{g}_u^{n+1}\right) \,. \qquad (8.34)$$

As discussed in Section 2.5.1, a computational efficient variant is to apply the divergence and continuity penalty terms in a postprocessing step, see equation (2.191). For the numerical results studied in this chapter, the formulation shown in equations (8.32) and (8.33) is used with penalty terms added to the momentum equation.

The individual terms in the weak form have been derived in Section 2.4. Only the convective term, equation (2.94), needs to be adapted due to the ALE transport term

$$\begin{aligned} c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_h, \boldsymbol{u}_{\mathrm{G},h}; \boldsymbol{g}_u\right) = \left(\boldsymbol{v}_h, \left(\nabla \boldsymbol{u}_h\right) \cdot \left(\boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{G},h}\right)\right)_{\Omega_e} - \left(\boldsymbol{v}_h, \left(\left(\{\{\boldsymbol{u}_h\}\} - \boldsymbol{u}_{\mathrm{G},h}\right) \cdot \boldsymbol{n}\right) \boldsymbol{u}_h\right)_{\partial\Omega_e} \\ + \Big(\boldsymbol{v}_h, \underbrace{\left(\left(\{\{\boldsymbol{u}_h\}\} - \boldsymbol{u}_{\mathrm{G},h}\right) \cdot \boldsymbol{n}\right) \{\{\boldsymbol{u}_h\}\} + \frac{1}{2} \left| \left(\{\{\boldsymbol{u}_h\}\} - \boldsymbol{u}_{\mathrm{G},h}\right) \cdot \boldsymbol{n} \right| [\boldsymbol{u}_h]}_{\text{upwind flux}}\Big)_{\partial\Omega_e} \,. \end{aligned} \qquad (8.35)$$

Finally, the linearization of the convective term is specified. Following the procedure in Section 2.4.2 yields

$$\begin{aligned} c_{h,\mathrm{lin}}^e(\boldsymbol{v}_h, \boldsymbol{u}_{h,\mathrm{lin}}, \boldsymbol{u}_{\mathrm{G},h}, \Delta\boldsymbol{u}_h) = \left(\boldsymbol{v}_h, \left(\nabla \boldsymbol{u}_{h,\mathrm{lin}}\right) \cdot \Delta\boldsymbol{u}_h\right)_{\Omega_e} + \left(\boldsymbol{v}_h, \left(\nabla(\Delta\boldsymbol{u}_h)\right) \cdot \left(\boldsymbol{u}_{h,\mathrm{lin}} - \boldsymbol{u}_{\mathrm{G},h}\right)\right)_{\Omega_e} \\ - \left(\boldsymbol{v}_h, \left(\left(\{\{\boldsymbol{u}_{h,\mathrm{lin}}\}\} - \boldsymbol{u}_{\mathrm{G},h}\right) \cdot \boldsymbol{n}\right) \Delta\boldsymbol{u}_h\right)_{\partial\Omega_e} - \left(\boldsymbol{v}_h, \left(\{\{\Delta\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right) \boldsymbol{u}_{h,\mathrm{lin}}\right)_{\partial\Omega_e} \\ + \left(\boldsymbol{v}_h, \left(\left(\{\{\boldsymbol{u}_{h,\mathrm{lin}}\}\} - \boldsymbol{u}_{\mathrm{G},h}\right) \cdot \boldsymbol{n}\right) \{\{\Delta\boldsymbol{u}_h\}\} + \left(\{\{\Delta\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right) \{\{\boldsymbol{u}_{h,\mathrm{lin}}\}\}\right)_{\partial\Omega_e} \\ + \left(\boldsymbol{v}_h, \frac{1}{2} \left| \left(\{\{\boldsymbol{u}_{h,\mathrm{lin}}\}\} - \boldsymbol{u}_{\mathrm{G},h}\right) \cdot \boldsymbol{n} \right| [\Delta\boldsymbol{u}_h]\right)_{\partial\Omega_e} \,. \end{aligned} \qquad (8.36)$$

8.4.3 Dual splitting scheme

For the dual splitting projection scheme, the variational formulation can be summarized as follows: Find $\hat{\boldsymbol{u}}_h, \hat{\hat{\boldsymbol{u}}}_h, \hat{\hat{\hat{\boldsymbol{u}}}}_h, \boldsymbol{u}_h^{n+1} \in \mathcal{V}_h^u$ and $p_h^{n+1} \in \mathcal{V}_h^p$ such that for all $\boldsymbol{v}_h \in \mathcal{V}_{h,e}^u$, $q_h \in \mathcal{V}_{h,e}^p$ and for all elements $e = 1, ..., N_{\text{el}}$

$$\begin{aligned} m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \frac{\gamma_0^n \hat{\boldsymbol{u}}_h - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}_h^{n-i}}{\Delta t_n}\right) = \\ -\sum_{i=0}^{J-1} \beta_i^n c_h^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n-i}, \boldsymbol{u}_{\mathrm{G},h}^{n+1}; \boldsymbol{g}_u^{n+1}\right) + b_h^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{f}(t_{n+1})\right) \ , \end{aligned} \tag{8.37}$$

$$l_{h,\text{hom}}^{e,n+1}\left(q_h, p_h^{n+1}\right) = -\frac{\gamma_0^n}{\Delta t_n} d_h^{e,n+1}\left(q_h, \hat{\boldsymbol{u}}_h; \boldsymbol{g}_{\hat{u}}^{n+1}\right) - l_{h,\text{inhom}}^{e,n+1}\left(q_h; g_p^{n+1}, h_p^{n+1}\right) \ , \tag{8.38}$$

$$m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \hat{\hat{\boldsymbol{u}}}_h\right) = m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h\right) - \frac{\Delta t_n}{\gamma_0^n} g_h^{e,n+1}\left(\boldsymbol{v}_h, p_h^{n+1}; g_p^{n+1}\right) \ , \tag{8.39}$$

$$\begin{aligned} m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \frac{\gamma_0^n}{\Delta t_n}\hat{\hat{\boldsymbol{u}}}_h\right) + v_{h,\text{hom}}^{e,n+1}\left(\boldsymbol{v}_h, \hat{\hat{\hat{\boldsymbol{u}}}}_h\right) = \\ m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \frac{\gamma_0^n}{\Delta t_n}\hat{\hat{\boldsymbol{u}}}_h\right) - v_{h,\text{inhom}}^{e,n+1}\left(\boldsymbol{v}_h; \boldsymbol{g}_u^{n+1}, \boldsymbol{h}_u^{n+1}\right) \ , \end{aligned} \tag{8.40}$$

$$\begin{aligned} m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}\right) + a_{\mathrm{D},h}^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}\right)\Delta t_n + a_{\mathrm{C},h,\text{hom}}^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}\right)\Delta t_n = \\ m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \hat{\hat{\hat{\boldsymbol{u}}}}_h\right) - a_{\mathrm{C},h,\text{inhom}}^{e,n+1}\left(\boldsymbol{v}_h; \boldsymbol{g}_u^{n+1}\right)\Delta t_n \ . \end{aligned} \tag{8.41}$$

8.4.4 Pressure-correction scheme

For the class of pressure-correction methods, the variational formulation can be summarized as follows: Find $\hat{\boldsymbol{u}}_h$, $\hat{\hat{\boldsymbol{u}}}_h$, $\boldsymbol{u}_h^{n+1} \in \mathcal{V}_h^u$ and $\phi_h^{n+1}, p_h^{n+1} \in \mathcal{V}_h^p$ such that for all $\boldsymbol{v}_h \in \mathcal{V}_{h,e}^u$, $q_h \in \mathcal{V}_{h,e}^p$ and for all elements $e = 1, ..., N_{\text{el}}$

$$\begin{aligned} m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \frac{\gamma_0^n \hat{\boldsymbol{u}}_h - \sum_{i=0}^{J-1} \alpha_i^n \boldsymbol{u}_h^{n-i}}{\Delta t_n}\right) \\ +c_h^{e,n+1}\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h, \boldsymbol{u}_{\mathrm{G},h}^{n+1}; \boldsymbol{g}_u^{n+1}\right) + v_h^{e,n+1}\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h; \boldsymbol{g}_u^{n+1}, \boldsymbol{h}_u^{n+1}\right) = \\ -\sum_{i=0}^{J_p-1} \beta_i^n g_h^{e,n+1}\left(\boldsymbol{v}_h, p_h^{n-i}; g_p^{n-i}\right) + b_h^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{f}(t_{n+1})\right) \ , \end{aligned} \tag{8.42}$$

$$l_{h,\text{hom}}^{e,n+1}\left(q_h, \phi_h^{n+1}\right) = -\frac{\gamma_0^n}{\Delta t_n} d_h^{e,n+1}\left(q_h, \hat{\boldsymbol{u}}_h; \boldsymbol{g}_u^{n+1}\right) - l_{h,\text{inhom}}^{e,n+1}\left(q_h; g_\phi^{n+1}, h_\phi^{n+1}\right) \ , \tag{8.43}$$

$$m_{h,p}^{e,n+1}\left(q_h, p_h^{n+1}\right) = m_{h,p}^{e,n+1}\left(q_h, \phi_h^{n+1} + \sum_{i=0}^{J_p-1} \beta_i^n p_h^{n-i}\right) - \chi\nu \ d_h^{e,n+1}\left(q_h, \hat{\boldsymbol{u}}_h; \boldsymbol{g}_u^{n+1}\right) \ , \tag{8.44}$$

$$m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \hat{\hat{\boldsymbol{u}}}_h\right) = m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h\right) - \frac{\Delta t_n}{\gamma_0^n} g_h^{e,n+1}\left(\boldsymbol{v}_h, \phi_h^{n+1}; g_\phi^{n+1}\right) \ , \tag{8.45}$$

$$m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}\right) + a_{\mathrm{D},h}^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}\right)\Delta t_n + a_{\mathrm{C},h,\mathrm{hom}}^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n+1}\right)\Delta t_n = m_{h,u}^{e,n+1}\left(\boldsymbol{v}_h, \hat{\hat{\boldsymbol{u}}}_h\right) - a_{\mathrm{C},h,\mathrm{inhom}}^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{g}_u^{n+1}\right)\Delta t_n \ . \tag{8.46}$$

As for the monolithic solver, an alternative formulation treating the convective term explicitly is obtained by replacing

$$c_h^{e,n+1}\left(\boldsymbol{v}_h, \hat{\boldsymbol{u}}_h, \boldsymbol{u}_{\mathrm{G},h}^{n+1}; \boldsymbol{g}_u^{n+1}\right) \rightarrow \sum_{i=0}^{J-1} \beta_i^n c_h^{e,n+1}\left(\boldsymbol{v}_h, \boldsymbol{u}_h^{n-i}, \boldsymbol{u}_{\mathrm{G},h}^{n+1}; \boldsymbol{g}_u^{n+1}\right) \ . \tag{8.47}$$

8.4.5 Geometric conservation law

It can be proven that the fully discrete ALE-DG methods derived above satisfy the geometric conservation law, i.e., they are able to preserve a constant flow field (Thomas and Lombard 1979). In other words, the constant solution $\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}_0$, $p(\boldsymbol{x}, t) = p_0$ is a solution of the fully discrete formulations for vanishing body forces, $\boldsymbol{f} = \boldsymbol{0}$.

Theorem 8.1 *Assume the solution at time instant t_n is given as $\boldsymbol{u}_h^n = \boldsymbol{u}_0$, $p_h^n = p_0$ where $u_{i,0} = C_i$, $i = 1, ..., d$, and $p_0 = C$ (and similarly for previous time instants in case of high-order schemes), and further assume $\boldsymbol{f} = \boldsymbol{0}$ and exact numerical integration of the velocity divergence term and pressure gradient term. Then, the fully discrete ALE-DG incompressible Navier–Stokes solvers introduced in Sections 8.4.2, 8.4.3, and 8.4.4 preserve a constant solution and yield $\boldsymbol{u}_h^{n+1} = \boldsymbol{u}_0$, $p_h^{n+1} = p_0$ at time $t_{n+1} = t_n + \Delta t_n$, independently of the order of the time integration and extrapolation schemes, and for arbitrary mesh velocities.*

Proof For the monolithic solver, insert the assumptions of Theorem 8.1 along with $\boldsymbol{u}_h^{n+1} = \boldsymbol{u}_0$, $p_h^{n+1} = p_0$ into equations (8.32) and (8.33) and show that the weak forms of all individual terms evaluate to zero in order to complete the proof. Obviously, the time derivative term becomes zero since it holds $\gamma_0^n = \sum_{i=0}^{J-1} \alpha_i^n$. For the other terms, it holds

$$\begin{aligned} &c_h^e\left(\boldsymbol{v}_h, \boldsymbol{u}_0, \boldsymbol{u}_{\mathrm{G},h}\right) = 0 \ , \ v_h^e(\boldsymbol{v}_h, \boldsymbol{u}_0) = 0 \ , \ d_{h,\mathrm{strong}}^e\left(q_h, \boldsymbol{u}_0\right) = 0 \ , \ g_{h,\mathrm{strong}}^e\left(\boldsymbol{v}_h, p_0\right) = 0 \ , \\ &a_{\mathrm{D},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_0) = 0 \ , \ a_{\mathrm{C},h}^e(\boldsymbol{v}_h, \boldsymbol{u}_0) = 0 \ . \end{aligned} \tag{8.48}$$

The volume term of the convective operator contains the gradient, and the face terms vanish due to the consistency of the numerical flux. Note that it is enough to consider interior faces in this context, as boundary faces behave the same when evaluating the boundary values according to Table 2.3 for a constant solution. The SIPG discretization of the viscous term also vanishes, as each form either contains gradients or jumps of the solution. The strong formulations of the velocity divergence term, equation (2.98), and pressure gradient term, equation (2.103), vanish for constant solutions, since the volume integrals contain derivatives of the solution and the face integrals contain jumps. These terms do not vanish exactly for the weak formulations of the velocity–pressure coupling terms, equations (2.97) and (2.102), when applying the quadrature rules from Section 2.5.5 and for arbitrarily deformed elements. The divergence and continuity penalty terms vanish for a constant solution, since they evaluate the divergence inside the element or the jump over faces. The GCL is also fulfilled when applying these terms in a postprocessing step, equation (2.191), since the remaining mass matrix system implies an identity.

Regarding the dual splitting scheme, it follows from the above argumentation that the first sub-step in equation (8.37) yields $\hat{\boldsymbol{u}}_h = \boldsymbol{u}_0$. Particular attention has to be paid to the boundary conditions h_p in equation (8.18) and $\boldsymbol{g}_{\hat{u}}$ in equation (8.19). The pressure Neumann boundary condition vanishes for $\boldsymbol{f} = \boldsymbol{0}$, as it contains derivatives in either space or time, and one obtains $\boldsymbol{g}_{\hat{u}} = \boldsymbol{g}_u$ in the case of a constant solution. Hence, the divergence operator on the right-hand side of the pressure Poisson equation (8.38) vanishes, and the pressure Poisson equation is satisfied for a constant solution p_0. It immediately follows from the argumentation for the monolithic solver that the remaining sub-steps, equations (8.39), (8.40), and (8.41), yield $\hat{\hat{\boldsymbol{u}}}_h = \boldsymbol{u}_0$, $\hat{\hat{\hat{\boldsymbol{u}}}}_h = \boldsymbol{u}_0$, and $\boldsymbol{u}_h^{n+1} = \boldsymbol{u}_0$. Thus, a constant flow state is preserved.

Turning to the pressure-correction scheme, the momentum equation (8.42) results in $\hat{\boldsymbol{u}}_h = \boldsymbol{u}_0$ for reasons explained above. The pressure boundary condition g_ϕ becomes zero due to $\sum_{i=0}^{J_p-1} \beta_i^n = 1$, so that $\phi_h^{n+1} = 0$ is a solution of the pressure Poisson equation (8.43). The pressure update equation (8.44) results in $p_h^{n+1} = p_0$, since the divergence term on the right-hand side becomes zero. The same holds for the pressure gradient term in equation (8.45), since its arguments are $\phi_h^{n+1} = 0$, $g_\phi^{n+1} = 0$, so that the constant solution $\hat{\hat{\boldsymbol{u}}}_h = \boldsymbol{u}_0$ and $\boldsymbol{u}_h^{n+1} = \boldsymbol{u}_0$ is recovered in the last sub-steps (8.45) and (8.46). □

Remark 8.6 *Note that no assumption has been made regarding the mesh velocity or how it is computed numerically to show compliance with the GCL. This is a direct consequence of the fact that the differential form of the ALE equations in convective formulation, equation* (8.5)*, is discretized, where the time derivative is applied to the velocity only, as opposed to formulations that apply the time derivative to an integral quantity, see also Förster et al. (2006). In Section 8.5, it is demonstrated numerically that the geometric conservation law is fulfilled exactly for the strong formulations, and that it is not fulfilled exactly down to rounding errors for the weak formulations of the velocity–pressure coupling terms in general. However, since the temporal discretization and spatial discretization are designed to satisfy the GCL, no relevant difference is expected for practical problems with non-constant solution due to this variational crime. For example, fulfilling discrete energy stability exactly with respect to the velocity–pressure coupling terms requires that one term is formulated in weak form and the other one in strong form, so that the formulation becomes symmetric independently of integration errors, see Section 2.4.5. Previous works have shown that fulfilling the GCL is neither a necessary nor a sufficient condition for the time integrator to preserve its high-order accuracy on moving meshes (Étienne et al. 2009, Geuzaine et al. 2003). Therefore, the temporal convergence behavior of the present ALE schemes is carefully investigated in Section 8.5, where it is demonstrated that the high-order accuracy of the Navier–Stokes solvers on fixed meshes is preserved on moving meshes when using definition* (8.31) *to calculate the mesh velocity.*

8.4.6 Energy stability

This section extends Section 2.4.5 to the case of moving meshes. The aim is to investigate the energy stability of the semi-discrete formulation for moving meshes and the influence that the additional ALE transport term has on energy stability. Hence, the assumption of a time-invariant domain from Section 2.4.5 is dropped here, while the additional assumption of a periodic motion of the domain with a periodic grid velocity is introduced. Moreover, the viscous term is omitted for this analysis since it has a dissipative character and since the critical case is to investigate

whether a numerical method is energy stable in the absence of viscous dissipation, see also Section 2.4.5. In a first step, the spatially continuous problem is studied in order to derive an equation describing energy-conservation for moving domains. In a second step, energy stability is investigated for the spatially discretized problem. For the continuous-in-space problem, begin with

$$\begin{aligned}\int_{\Omega(t)} \left.\frac{\partial \frac{1}{2}\boldsymbol{u}\cdot\boldsymbol{u}}{\partial t}\right|_{\boldsymbol{\chi}} \mathrm{d}\Omega &= \int_{\Omega(t)} \boldsymbol{u}\cdot\left.\frac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{\chi}} \mathrm{d}\Omega \\ &= -\int_{\Omega(t)} \boldsymbol{u}\cdot(((\boldsymbol{u}-\boldsymbol{u}_{\mathrm{G}})\cdot\nabla)\boldsymbol{u}+\nabla p)\,\mathrm{d}\Omega\,,\end{aligned} \tag{8.49}$$

where the momentum equation in ALE form, equation (8.5), has been inserted in the second step. Next, the convective term is reformulated by making use of the identity $\boldsymbol{u}\cdot\nabla\boldsymbol{u}\cdot\boldsymbol{w} = -\frac{1}{2}(\boldsymbol{u}\cdot\boldsymbol{u})\nabla\cdot\boldsymbol{w} + \frac{1}{2}\nabla\cdot(\boldsymbol{w}(\boldsymbol{u}\cdot\boldsymbol{u}))$ with $\boldsymbol{w} = \boldsymbol{u}-\boldsymbol{u}_{\mathrm{G}}$, and by applying Gauss' divergence theorem for the second term

$$\begin{aligned}\int_{\Omega(t)} \boldsymbol{u}\cdot\nabla\boldsymbol{u}\cdot(\boldsymbol{u}-\boldsymbol{u}_{\mathrm{G}})\;\mathrm{d}\Omega &= -\frac{1}{2}\int_{\Omega(t)} (\boldsymbol{u}\cdot\boldsymbol{u})\nabla\cdot(\boldsymbol{u}-\boldsymbol{u}_{\mathrm{G}})\,\mathrm{d}\Omega \\ &\quad + \frac{1}{2}\int_{\partial\Omega(t)} (\boldsymbol{u}\cdot\boldsymbol{u})(\boldsymbol{u}-\boldsymbol{u}_{\mathrm{G}})\cdot\boldsymbol{n}\,\mathrm{d}\Gamma \\ &= +\frac{1}{2}\int_{\Omega(t)} (\boldsymbol{u}\cdot\boldsymbol{u})\nabla\cdot\boldsymbol{u}_{\mathrm{G}}\,\mathrm{d}\Omega\,.\end{aligned} \tag{8.50}$$

In the second step, it has been exploited that the surface integral vanishes due to the assumption of periodicity, and that $\nabla\cdot\boldsymbol{u} = 0$ holds in the continuous case. For the pressure gradient term, integration by parts yields

$$\int_{\Omega(t)} \boldsymbol{u}\nabla p\,\mathrm{d}\Omega = -\int_{\Omega(t)} \underbrace{\nabla\cdot\boldsymbol{u}}_{=0}\; p\,\mathrm{d}\Omega + \int_{\partial\Omega(t)} p\;\boldsymbol{u}\cdot\boldsymbol{n}\,\mathrm{d}\Gamma = 0\,, \tag{8.51}$$

where the volume integral vanished due to $\nabla\cdot\boldsymbol{u} = 0$ and the surface integral due to periodic boundaries. Inserting equations (8.50) and (8.51) into equation (8.49) yields an equation describing energy-conservation on moving domains

$$\int_{\Omega(t)} \left.\frac{\partial \frac{1}{2}\boldsymbol{u}\cdot\boldsymbol{u}}{\partial t}\right|_{\boldsymbol{\chi}} \mathrm{d}\Omega + \int_{\Omega(t)} \frac{1}{2}(\boldsymbol{u}\cdot\boldsymbol{u})\nabla\cdot\boldsymbol{u}_{\mathrm{G}}\,\mathrm{d}\Omega = 0\,. \tag{8.52}$$

Then, the spatially discretized ALE-DG formulation

$$\begin{aligned} m_h^e\left(\boldsymbol{v}_h,\left.\frac{\partial \boldsymbol{u}_h}{\partial t}\right|_{\boldsymbol{\chi}}\right) + c_h^e(\boldsymbol{v}_h,\boldsymbol{u}_h,\boldsymbol{u}_{\mathrm{G},h}) \\ +a_{\mathrm{D},h}^e(\boldsymbol{v}_h,\boldsymbol{u}_h) + a_{\mathrm{C},h}^e(\boldsymbol{v}_h,\boldsymbol{u}_h) + g_h^e(\boldsymbol{v}_h,p_h) = \boldsymbol{0}\,,\end{aligned} \tag{8.53}$$

$$-d_h^e(q_h,\boldsymbol{u}_h) = 0\,, \tag{8.54}$$

is called energy-stable if it fulfills the following discrete analogy

$$\int_{\Omega_h(t)} \left.\frac{\partial \frac{1}{2}\boldsymbol{u}_h\cdot\boldsymbol{u}_h}{\partial t}\right|_{\boldsymbol{\chi}} \mathrm{d}\Omega + \sum_{e=1}^{N_{\mathrm{el}}}\int_{\Omega_e(t)} \frac{1}{2}(\boldsymbol{u}_h\cdot\boldsymbol{u}_h)\nabla\cdot\boldsymbol{u}_{\mathrm{G},h}\mathrm{d}\Omega \leq 0\,. \tag{8.55}$$

To begin with, the left term in equation (8.55) is reformulated so that the semi-discrete momentum equation (8.53) can be inserted

$$\begin{aligned}\int_{\Omega_h(t)} \left.\frac{\partial \frac{1}{2}\boldsymbol{u}_h \cdot \boldsymbol{u}_h}{\partial t}\right|_{\boldsymbol{\chi}} \mathrm{d}\Omega = \int_{\Omega_h(t)} \boldsymbol{u}_h \cdot \left.\frac{\partial \boldsymbol{u}_h}{\partial t}\right|_{\boldsymbol{\chi}} \mathrm{d}\Omega = \sum_{e=1}^{N_\mathrm{el}} m_h^e \left(\boldsymbol{u}_h, \left.\frac{\partial \boldsymbol{u}_h}{\partial t}\right|_{\boldsymbol{\chi}}\right) \\ = -\sum_{e=1}^{N_\mathrm{el}} \left(c_h^e\left(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{u}_{\mathrm{G},h}\right) + g_h^e\left(\boldsymbol{u}_h, p_h\right) + a_{\mathrm{D},h}^e(\boldsymbol{u}_h, \boldsymbol{u}_h) + a_{\mathrm{C},h}^e(\boldsymbol{u}_h, \boldsymbol{u}_h)\right) .\end{aligned} \tag{8.56}$$

As explained in Section 2.4.5, the pressure gradient term drops out due to its symmetry with the velocity divergence term (under the assumption of exact integration) and due to the discrete continuity equation. The divergence and continuity penalty terms are symmetric positive semi-definite, so that it remains to investigate the convective term, which differs from the analysis on static meshes due to an additional ALE transport term. Inserting the identity

$$\boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h \cdot \boldsymbol{w}_h = -\frac{1}{2}\nabla \cdot \boldsymbol{w}_h \left(\boldsymbol{u}_h \cdot \boldsymbol{u}_h\right) + \frac{1}{2}\nabla \cdot \left(\boldsymbol{w}_h \left(\boldsymbol{u}_h \cdot \boldsymbol{u}_h\right)\right) , \tag{8.57}$$

with $\boldsymbol{w}_h = \boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{G},h}$ into equation (8.35), and applying Gauss' divergence theorem yields

$$\begin{aligned}c_h^e\left(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{u}_{\mathrm{G},h}\right) = &-\frac{1}{2}\left(\nabla \cdot (\boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{G},h}), \boldsymbol{u}_h \cdot \boldsymbol{u}_h\right)_{\Omega_e} + \frac{1}{2}\left(\boldsymbol{u}_h \cdot \boldsymbol{u}_h, (\boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{G},h}) \cdot \boldsymbol{n}\right)_{\partial\Omega_e} \\ &- \left(\boldsymbol{u}_h, \left(\left(\{\{\boldsymbol{u}_h\}\} - \boldsymbol{u}_{\mathrm{G},h}\right) \cdot \boldsymbol{n}\right) \frac{1}{2}\left[\boldsymbol{u}_h\right]\right)_{\partial\Omega_e} \\ &+ \left(\boldsymbol{u}_h, \frac{1}{2}\left|\left(\{\{\boldsymbol{u}_h\}\} - \boldsymbol{u}_{\mathrm{G},h}\right) \cdot \boldsymbol{n}\right| \left[\boldsymbol{u}_h\right]\right)_{\partial\Omega_e} .\end{aligned} \tag{8.58}$$

Next, summation over all elements is performed and the face integrals with the grid velocity are written separately

$$\begin{aligned}c_h\left(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{u}_{\mathrm{G},h}\right) = &-\frac{1}{2}\left(\nabla \cdot (\boldsymbol{u}_h - \boldsymbol{u}_{\mathrm{G},h}), \boldsymbol{u}_h \cdot \boldsymbol{u}_h\right)_{\Omega_h} \\ &\underbrace{-\frac{1}{2}\left(\left[\boldsymbol{u}_h \cdot \boldsymbol{u}_h\right], \boldsymbol{u}_{\mathrm{G},h} \cdot \boldsymbol{n}\right)_{\Gamma_h^\mathrm{int}} + \left(\left[\boldsymbol{u}_h\right] \cdot \{\{\boldsymbol{u}_h\}\}, \boldsymbol{u}_{\mathrm{G},h} \cdot \boldsymbol{n}\right)_{\Gamma_h^\mathrm{int}}}_{=0 \text{ (linear transport)}} \\ &+ \frac{1}{2}\left(\boldsymbol{u}_h^- \cdot \boldsymbol{u}_h^-, \boldsymbol{u}_h^- \cdot \boldsymbol{n}^-\right)_{\Gamma_h^\mathrm{int}} + \frac{1}{2}\left(\boldsymbol{u}_h^+ \cdot \boldsymbol{u}_h^+, \boldsymbol{u}_h^+ \cdot \boldsymbol{n}^+\right)_{\Gamma_h^\mathrm{int}} \\ &- \left(\left[\boldsymbol{u}_h\right] \cdot \{\{\boldsymbol{u}_h\}\}, \{\{\boldsymbol{u}_h\}\} \cdot \boldsymbol{n}\right)_{\Gamma_h^\mathrm{int}} \\ &+ \left(\left[\boldsymbol{u}_h\right], \frac{1}{2}\left|\left(\{\{\boldsymbol{u}_h\}\} - \boldsymbol{u}_{\mathrm{G},h}\right) \cdot \boldsymbol{n}\right| \left[\boldsymbol{u}_h\right]\right)_{\Gamma_h^\mathrm{int}} .\end{aligned} \tag{8.59}$$

The first term on the right-hand side contains the divergence of the velocity, i.e., a residual of the incompressible Navier–Stokes equations, and the divergence of the grid velocity. The volume integral of the ALE transport term does not drop out since the grid velocity is not divergence-free. However, one can see that this is exactly the second term in equation (8.55). The key point is

that the face integrals related to the moving mesh drop out, since the ALE term describes a linear transport term. Other terms of the above equation can be further simplified as shown in equation (2.185) for the case with static meshes. Then, inserting equation (8.59) into equation (8.56) yields the result

$$\begin{aligned}\int_{\Omega_h(t)} \left.\frac{\partial \frac{1}{2}\boldsymbol{u}_h\cdot\boldsymbol{u}_h}{\partial t}\right|_{\boldsymbol{\chi}} \mathrm{d}\Omega + \sum_{e=1}^{N_{\mathrm{el}}} \int_{\Omega_e(t)} \frac{1}{2}\left(\boldsymbol{u}_h\cdot\boldsymbol{u}_h\right)\nabla\cdot\boldsymbol{u}_{\mathrm{G},h}\mathrm{d}\Omega = \\ = +\frac{1}{2}\left(\nabla\cdot\boldsymbol{u}_h,\boldsymbol{u}_h\cdot\boldsymbol{u}_h\right)_{\Omega_h} - a_{\mathrm{D},h}(\boldsymbol{u}_h,\boldsymbol{u}_h) \\ -\frac{1}{2}\left([\boldsymbol{u}_h]\cdot\boldsymbol{n},\{\{\boldsymbol{u}_h\cdot\boldsymbol{u}_h\}\}\right)_{\Gamma_h^{\mathrm{int}}} - a_{\mathrm{C},h}(\boldsymbol{u}_h,\boldsymbol{u}_h) \\ -\left([\boldsymbol{u}_h],\frac{1}{2}|\left(\{\{\boldsymbol{u}_h\}\}-\boldsymbol{u}_{\mathrm{G},h}\right)\cdot\boldsymbol{n}|\,[\boldsymbol{u}_h]\right)_{\Gamma_h^{\mathrm{int}}}.\end{aligned} \tag{8.60}$$

It is interesting to realize that the ALE formulation does not introduce new terms on the right-hand side as compared to the Eulerian case, equation (2.186), a consequence of the fact that the additional ALE term is a linear transport term. One might therefore argue that energy stability for the Eulerian case translates into energy stability for the ALE case with moving meshes. The reader is referred to Section 2.4.5 for a general discussion of the above energy estimate. An aspect not reflected by the above energy estimate is that of inexact integration, e.g. due to deformed elements, which can be expected to be negligible for smooth problems but might have an influence for under-resolved problems and strongly-deformed elements. Another aspect concerns the assumption of a smooth solution implying energy-conservation in the absence of viscosity in the continuous case, which might not be fulfilled due to singularities developing in finite time for the incompressible Euler equations with the occurrence of anomalous energy dissipation, see Chapter 7 for a detailed discussion of this phenomenon.

8.5 Numerical results

The aim of this section is to display the numerical discretization properties of the proposed ALE-DG incompressible Navier–Stokes solvers. A set of academic test cases that address different aspects of ALE solvers is selected, with the goal to obtain a picture as complete as possible. In detail, the geometric conservation property is studied by the example of the free stream preservation test in Section 8.5.1. The convergence behavior in terms of temporal and spatial convergence rates is investigated in Section 8.5.2 by the example of a two-dimensional vortex problem with moving Dirichlet and Neumann boundaries. In this section, also the robustness of the different incompressible Navier–Stokes solvers is tested in the limit of large grid deformations. Finally, the robustness of the proposed discretization methods for under-resolved turbulent flows is studied in Section 8.5.3 by considering the three-dimensional Taylor–Green vortex problem.

The Laplace formulation of the viscous term is used for the results shown in this chapter. Convergence rates and relative L^2-errors for problems with known analytical solution are computed as defined in Fehn et al. (2017). Solver tolerances are selected to not spoil accuracy, e.g., by choosing relative solver tolerances of 10^{-6} and absolute solver tolerances of 10^{-12}. In case

of pure Dirichlet boundary conditions, the pressure level is undefined and can be fixed, e.g., by setting the mean value of the pressure DoF vector to zero.

8.5.1 Geometric conservation law – free stream preservation test

The free stream preservation test is studied to investigate whether the ALE formulations derived above fulfill the geometric conservation law. Satisfying the geometric conservation law means that a constant flow field is not disturbed by a moving mesh, i.e., the solver is able to preserve the free stream flow conditions exactly and independently of the mesh deformation. The analytical solution of the free stream preservation test is therefore the constant flow state

$$\boldsymbol{u}(\boldsymbol{x},t) = (1,\ldots,1)^{\mathsf{T}}\ ,\ p(\boldsymbol{x},t) = 1\ , \tag{8.61}$$

where pure Dirichlet boundary conditions are prescribed, $\boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{g}_u(\boldsymbol{x},t)$ on $\Gamma_h^{\mathrm{D}}(t) = \Gamma_h(t)$. The computational domain at initial time is $\Omega_0 = \Omega(t=0) = [-L/2, L/2]^2$ with $L = 1$. The simulation is run over a time interval of $0 \leq t \leq T = 10$. The following analytical mesh movement with sine functions in both time and space is prescribed in two space dimensions

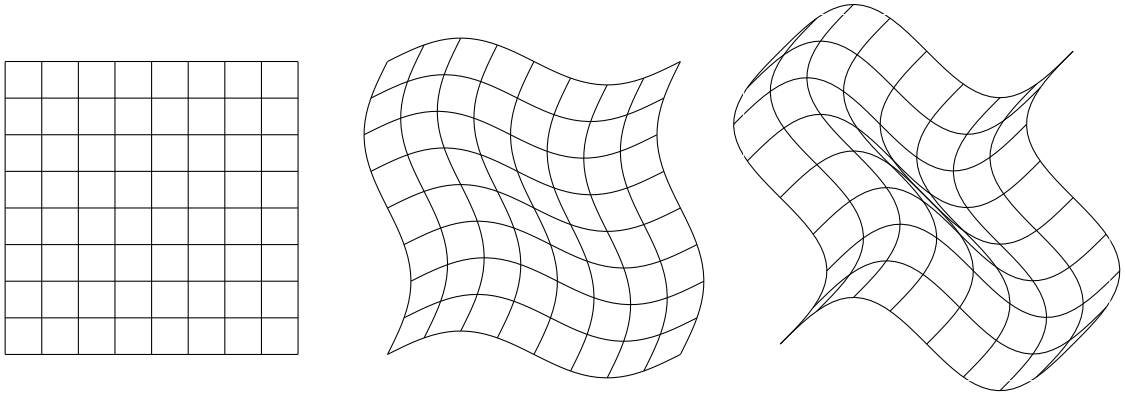
$$\boldsymbol{x}(\boldsymbol{\chi},t) = \boldsymbol{\chi} + A\sin\left(2\pi\frac{t}{T_{\mathrm{G}}}\right)\begin{pmatrix}\sin\left(2\pi\frac{\chi_2+L/2}{L}\right)\\ \sin\left(2\pi\frac{\chi_1+L/2}{L}\right)\end{pmatrix}\ , \tag{8.62}$$

and in three space dimensions

$$\boldsymbol{x}(\boldsymbol{\chi},t) = \boldsymbol{\chi} + A\sin\left(2\pi\frac{t}{T_{\mathrm{G}}}\right)\begin{pmatrix}\sin\left(2\pi\frac{\chi_2+L/2}{L}\right)\sin\left(2\pi\frac{\chi_3+L/2}{L}\right)\\ \sin\left(2\pi\frac{\chi_1+L/2}{L}\right)\sin\left(2\pi\frac{\chi_3+L/2}{L}\right)\\ \sin\left(2\pi\frac{\chi_1+L/2}{L}\right)\sin\left(2\pi\frac{\chi_2+L/2}{L}\right)\end{pmatrix}\ , \tag{8.63}$$

where the amplitude is set to $A = 0.08$ resulting in a strongly deformed mesh. The period length of the grid motion is set to $T_{\mathrm{G}} = T/10$ and the wavenumber in space is chosen such that the length and height of the domain are exactly one period. In Figure 8.2(b), the mesh deformation is illustrated for $d = 2$, where the initial, undeformed mesh is a uniform Cartesian grid. The mesh reaches its maximum deformation at times $t = T_{\mathrm{G}}/4 + i\,T_{\mathrm{G}}/2, i = 0, 1, 2, \ldots$. The viscosity is set to $\nu = 0.025$. Adaptive time-stepping is used where the time step size is adjusted dynamically according to the CFL condition (2.210) using $\mathrm{Cr} = 0.25$. A mesh with 8^d elements is used as in Figure 8.2 and the polynomial degree is $k = 3$. Both absolute and relative solver tolerances are set to a small value of 10^{-14}. Table 8.1 reports relative errors for velocity and pressure for BDF schemes of order 1 to 3 and the three different incompressible Navier–Stokes solvers considered in this work. For the dual splitting scheme, $J_p = \min(2, J)$ is used but the simulations have also been stable for the choice $J_p = J$ for the high-order BDF scheme $J = 3$. For the pressure-correction scheme, $J_p = \min(2, J)-1$ is used for all J and the rotational formulation with $\chi = 1$. Here, the choice $J_p = J - 1$ leads to instabilities for the high-order scheme $J = 3$ in agreement with theory.

The results in Table 8.1 reveal that all schemes fulfill the geometric conservation law for $d = 2$, and in particular also for the weak formulation of the velocity–pressure coupling terms.



(a) undeformed mesh (b) deformed mesh ($A/L = 0.08$) (c) theoretical limit $A_{\max}/L = \frac{1}{2\pi}$

Figure 8.2: Illustration of sine-like mesh motion in two space dimensions according to equation (8.62).

For $d = 3$, the geometric conservation law is fulfilled exactly only when using the strong formulations $d^e_{h,\text{strong}}$ and $g^e_{h,\text{strong}}$ as expected theoretically, see Section 8.4.5. Errors larger than the solver tolerances are observed for the weak formulations for $d = 3$. For all variants studied here, similar results are obtained when using an implicit formulation of the convective term for the coupled solver and the pressure-correction scheme, where the solver tolerances had to be relaxed slightly to 10^{-12} to ensure convergence of the Newton solver. Hence, it remains to explain why the weak formulation of velocity–pressure coupling terms fulfills the GCL exactly for $d = 2$. As the relevant aspect in this context is the exact evaluation of integrals, see Section 8.4.5, it can be conjectured that integrals of the velocity–pressure coupling terms are indeed evaluated exactly for $d = 2$ for constant solutions on deformed elements. However, this is a special case that only occurs for the free stream preservation test due to a solution of lowest polynomial degree, and this does not hold for general non-constant solutions and arbitrarily deformed elements. Hence, no further attention is paid to this point. In a similar direction, the results show that the weak formulations are very accurate as well, and will therefore be used for the subsequent examples. The reason for this choice is that the conclusion is even stronger when being able to demonstrate optimal convergence behavior for a formulation that appears to be sub-optimal regarding the free stream preservation test.

8.5.2 Temporal and spatial convergence behavior

Next, the convergence behavior of the ALE-DG methods is analyzed to study whether optimal rates of convergence observed in the Eulerian case carry over to moving meshes. For this purpose, the two-dimensional Taylor–Green vortex problem is considered, which has already been analyzed in Chapter 2 for the case of non-moving meshes. A description of the test case including the analytical solution is given in Section 2.6.5.1. The viscosity is set to $\nu = 0.025$, and the problem is simulated over the time interval $0 \leq t \leq T = 1$. The computational domain at start time is $\Omega_0 = [-L/2, L/2]^2$ with length $L = 1$, and is deformed according to the two-dimensional mesh movement function (8.62) with parameters $A = 0.08$ and $T_\text{G} = 4T$ (maximum deforma-

Table 8.1: Numerical results for free stream preservation test for both strong and weak formulations of velocity–pressure coupling terms and two- and three-dimensional problems: $J_p = \min(2, J)$ is used for the dual splitting scheme, and $J_p = \min(2, J) - 1$ for the pressure-correction scheme in rotational formulation. An explicit formulation of the convective term is used for all three solvers.

(a) two-dimensional problem ($d = 2$), weak formulations $d^e_{h,\text{weak}}$ and $g^e_{h,\text{weak}}$

	relative L^2-error $\boldsymbol{u}_h$			relative L^2-error p_h		
	BDF1	BDF2	BDF3	BDF1	BDF2	BDF3
coupled	1.8E–16	1.8E–15	2.1E–15	1.3E–16	6.0E–13	2.9E–13
dual splitting	1.2E–15	1.6E–15	1.8E–15	2.5E–13	7.5E–14	6.1E–13
pressure-correction	1.6E–15	2.7E–15	3.0E–15	1.8E–13	9.5E–14	4.6E–13

(b) two-dimensional problem ($d = 2$), strong formulations $d^e_{h,\text{strong}}$ and $g^e_{h,\text{strong}}$

	relative L^2-error $\boldsymbol{u}_h$			relative L^2-error p_h		
	BDF1	BDF2	BDF3	BDF1	BDF2	BDF3
coupled	1.8E–16	1.1E–15	1.8E–15	1.3E–16	2.1E–13	2.4E–13
dual splitting	1.0E–15	1.3E–15	1.4E–15	1.5E–13	3.8E–13	6.5E–13
pressure-correction	1.0E–15	1.6E–15	1.8E–15	2.3E–13	2.3E–13	4.7E–13

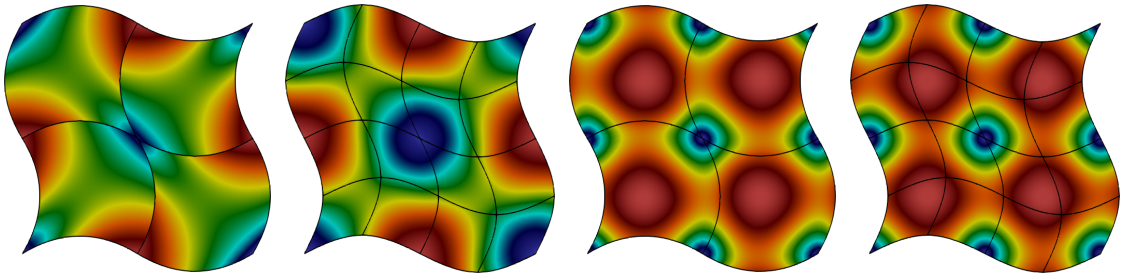
(c) three-dimensional problem ($d = 3$), weak formulations $d^e_{h,\text{weak}}$ and $g^e_{h,\text{weak}}$

	relative L^2-error $\boldsymbol{u}_h$			relative L^2-error p_h		
	BDF1	BDF2	BDF3	BDF1	BDF2	BDF3
coupled	1.3E–08	1.3E–08	1.3E–08	8.7E–09	8.2E–09	8.2E–09
dual splitting	1.1E–08	1.2E–08	1.2E–08	6.4E–09	6.6E–09	6.8E–09
pressure-correction	1.2E–08	1.3E–08	1.3E–08	7.0E–09	7.1E–08	6.1E–08

(d) three-dimensional problem ($d = 3$), strong formulations $d^e_{h,\text{strong}}$ and $g^e_{h,\text{strong}}$

	relative L^2-error $\boldsymbol{u}_h$			relative L^2-error p_h		
	BDF1	BDF2	BDF3	BDF1	BDF2	BDF3
coupled	3.5E–16	1.4E–14	1.2E–14	1.7E–16	8.5E–13	8.4E–13
dual splitting	3.5E–16	1.8E–15	1.1E–15	1.4E–13	7.3E–13	1.3E–12
pressure-correction	1.9E–15	1.9E–15	2.3E–15	3.3E–13	7.8E–13	1.1E–12

tion reached at end time $t = T$ is shown in Figure 8.2(b)) unless specified otherwise. Again, a mesh as depicted in Figure 8.2 with refinement level l is used. Sine-like mesh deformations are commonly used to verify high-order ALE-DG implementations, see for example Mavriplis and Nastase (2011), Nguyen (2010), Schnücke et al. (2018). In these works, however, the sine func-



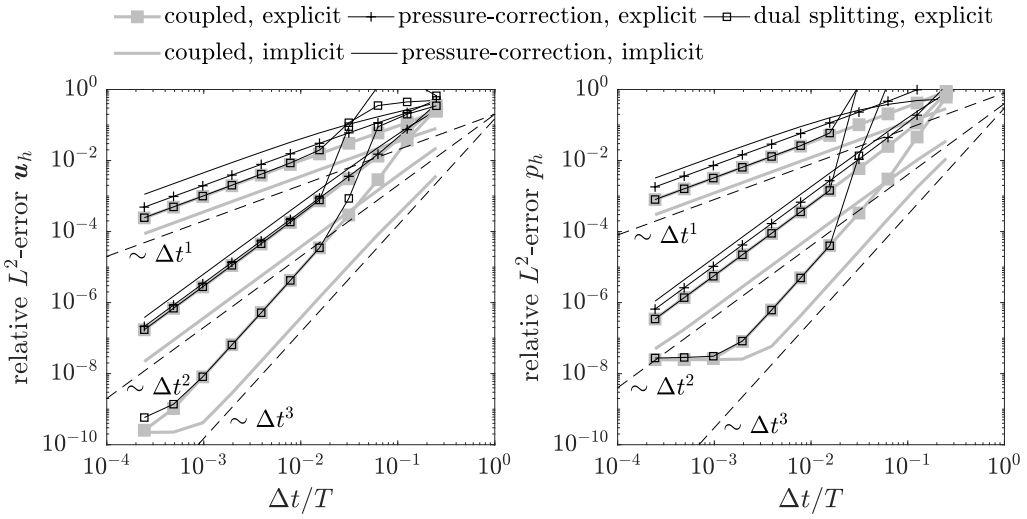
(a) $p(\boldsymbol{x}, T)$ $(l = 1)$ (b) $p(\boldsymbol{x}, T)$ $(l = 2)$ (c) $\|\boldsymbol{u}(\boldsymbol{x}, T)\|$ $(l = 1)$ (d) $\|\boldsymbol{u}(\boldsymbol{x}, T)\|$ $(l = 2)$

Figure 8.3: Vortex problem: visualization of solution at final time $t = T$ for two different mesh resolutions, $l = 1$ and $l = 2$, with polynomial degree $k = 3$ for the velocity and 2 for the pressure (red indicates high values and blue low values). The amplitude of the mesh deformation is $A = 0.08$.

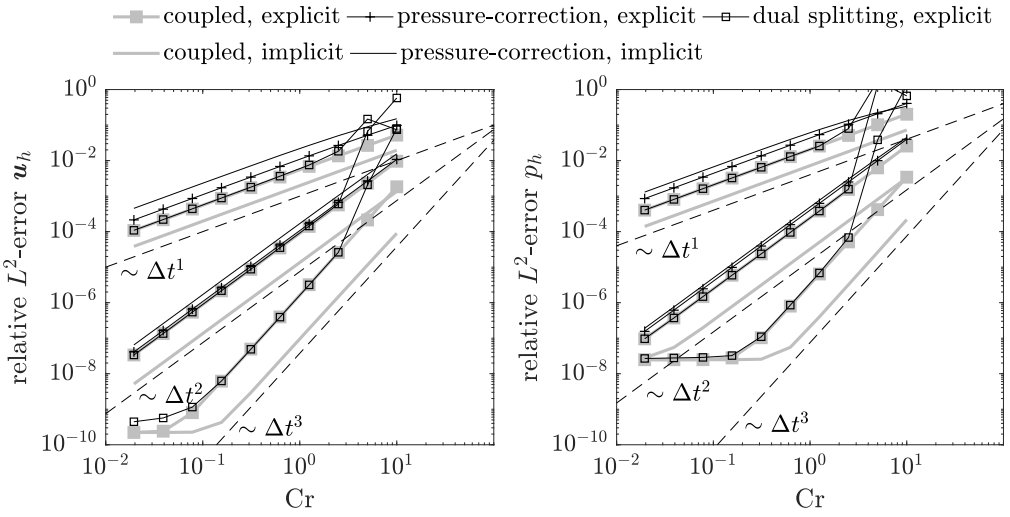
tions are defined in a way that the boundaries are not moving. Instead, a setup is chosen here for which the boundaries are moving since the goal is to test all parts of the algorithm relevant for FSI. The verification of boundary conditions is particularly relevant for the splitting-type solvers and some effects might not be visible if the boundaries are fixed. For example, if the boundaries are non-moving, the normal vector in equation (8.18) would not change over time and the ALE transport term would simply drop out since $\boldsymbol{u}_{\mathrm{G}} = \boldsymbol{0}$ on the boundary. According to the setup in Hesthaven and Warburton (2007), each of the four sides of the square is split into a Dirichlet boundary and a Neumann boundary according to the inflow and outflow sections, respectively. Note also that the chosen mesh deformation is in compliance with these boundary conditions.

Figure 8.3 shows a visualization of the solution at the time of maximal mesh deformation $t = T$ using polynomial shape functions of degree $k = 3$ and considering the two lowest refinement levels of $l = 1, 2$ (the grid has to consist of at least 2^2 elements due to the type of boundary conditions prescribed with each face of the rectangular domain cut into a Dirichlet part and a Neumann part). While the velocity field is already well resolved on the coarsest mesh with $l = 1$, the pressure field of polynomial degree 2 is approximated poorly for refinement level $l = 1$ with distinct discontinuities between the elements. For $l = 2$, the pressure solution appears to be visually converged with only minor differences as compared to the solution on even finer meshes. In the following, the convergence is studied quantitatively in terms of errors against the analytical solution as well as convergence rates measured in space and time.

A first set of experiments tests the temporal convergence behavior for both constant and adaptive time step sizes. The chosen spatial resolution is fine, $l = 3$ and $k = 8$, to make sure that errors are dominated by temporal discretization errors. Recalling from Section 2.6.5 that the CFL condition did not show up in this example for the chosen parameters, this test case is very interesting since it allows to measure temporal convergence rates of an incompressible Navier–Stokes solver with explicit formulation of the convective term. Note that in the regular case with $\mathrm{Cr} < \mathrm{Cr}_{\mathrm{crit}}$ this could be very difficult, since one is often operating in a regime where temporal discretization errors are already negligible as compared to spatial errors for higher-order time integration schemes. Figure 8.4 shows results of a temporal convergence study for BDF schemes of order $J = 1, 2, 3$ using constant and adaptive time step sizes. All types of



(a) constant Δt



(b) adaptive Δt_n

Figure 8.4: Vortex problem: temporal convergence tests for ALE incompressible Navier–Stokes solvers for BDF schemes of order $J = 1, ..., 3$ with $J_p = \min(2, J)$ for the dual splitting scheme and $J_p = \min(2, J) - 1, J \leq 2$ for the pressure-correction scheme.

solvers converge with optimal rates of convergence on the moving mesh. Compared to additional simulations performed for a static Cartesian mesh, the errors are almost the same and only slightly larger. The lowest errors are obtained for the coupled solver with implicit formulation of the convective term. The dual splitting scheme and the coupled solver with explicit convective term yield similar errors, and the errors are again slightly larger for both explicit and implicit pressure-correction formulations. For very small time step sizes and the BDF3 scheme, the spatial error becomes dominant at some point. Note that BDF3 schemes are not considered for the pressure-correction scheme since $J_p = 1$ (required for stability) limits convergence rates

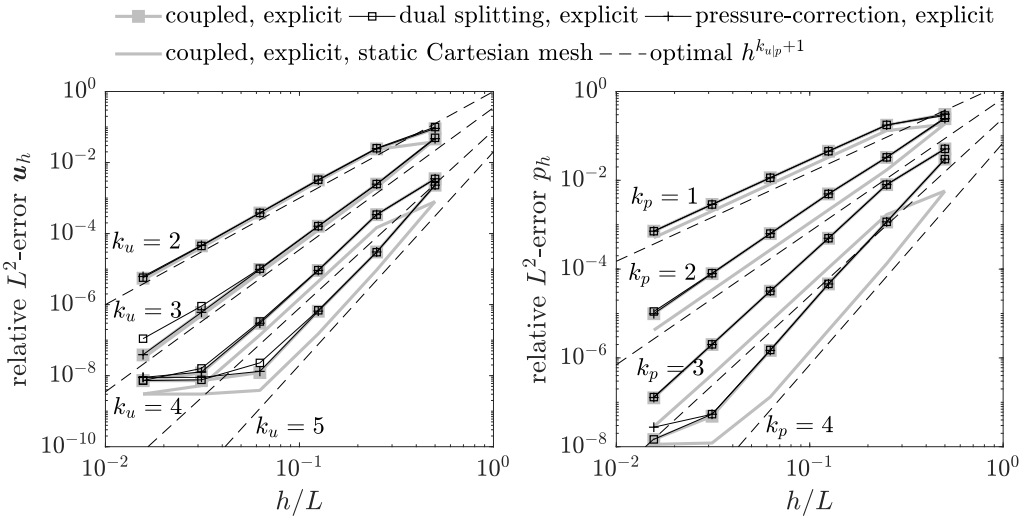


Figure 8.5: Vortex problem: spatial convergence tests for ALE incompressible Navier–Stokes solvers for polynomial degrees $k = 2, 3, 4, 5$ and comparison to static Cartesian mesh.

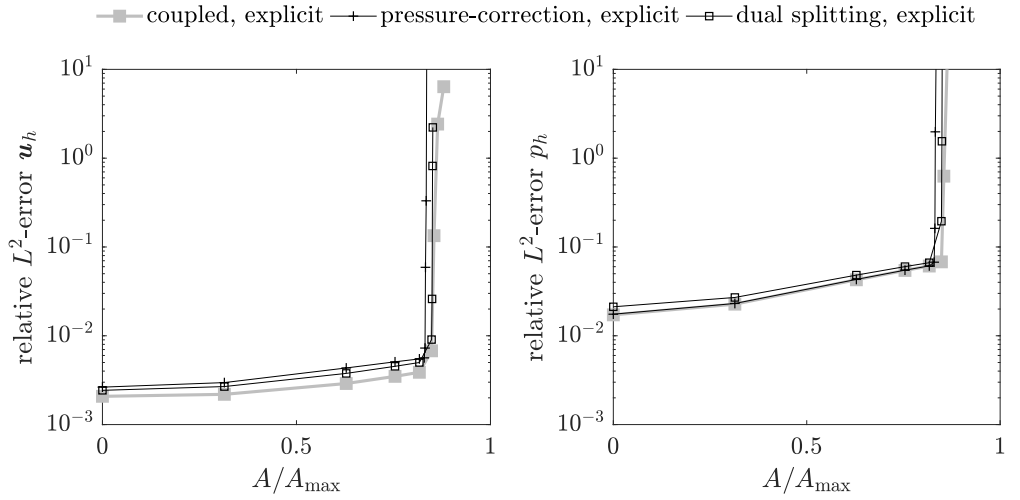


Figure 8.6: Vortex problem: robustness test of ALE incompressible Navier–Stokes solvers on a mesh with 4^2 elements and polynomial degree $k = 3$.

to second-order in that case. For the dual splitting scheme, large errors occur for Courant numbers $\mathrm{Cr} > 1$. This effect does not show up for the non-moving mesh and it is conjectured that this effect originates from the CFL condition. For a regular Navier–Stokes problem with explicit formulation of the convective term, no stability can be expected in this range of Courant numbers. While a sharp CFL bound is not visible for the chosen parameters, all solvers become unstable for $\mathrm{Cr} > 1$ (for $J = 2$) when using a smaller viscosity of $\nu = 10^{-3}$ (higher Reynolds number), showing the usual sharp CFL bound.

A second set of experiments studies the spatial convergence behavior for polynomial degrees $k = 2, 3, 4, 5$ by a mesh refinement study, considering refine levels $l = 1, ..., 6$. The BDF2 time integration scheme with a small, constant time step of $\Delta t = 5 \cdot 10^{-5}$ is used to obtain small temporal errors. Figure 8.5 shows results for the three different solver types with an explicit formulation of the convective term. Since this is a spatial convergence test, the results would be indistinguishable when using an implicit formulation of the convective term and are therefore not shown explicitly. For comparison, results are also shown for a static Cartesian mesh for the coupled solution approach. Overall, all variants converge with optimal rates of convergence for all polynomial degrees until the temporal discretization error is reached. Compared to the static mesh, the errors are slightly larger on the moving mesh, and the gap between moving and static meshes increases for increasing polynomial degree.

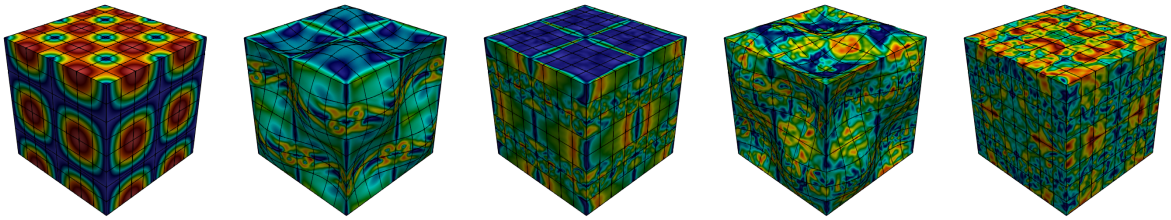
Finally, the robustness w.r.t. large mesh deformations is tested by increasing the amplitude A of the mesh movement function to its theoretical limit, i.e., the value at which the aspect ratio tends to infinity or at which inverted elements occur, in order to characterize the point where the proposed ALE-DG methods will break down. This theoretical limit is reached when the lower left corner of the domain becomes an arbitrarily thin needle, see Figure 8.2(c). Mathematically, this limit is reached when the slope of the lower domain boundary reaches a value of 1 in the lower left corner at the time of maximum deformation, i.e.,

$$\left.\frac{\partial x_2(\chi_1, \chi_2 = -L/2, t = T_\mathrm{G}/4)}{\partial \chi_1}\right|_{\chi_1 = -L/2} = A \cos\left(2\pi \frac{\chi_1 + L/2}{L}\right)\Bigg|_{\chi_1 = -L/2} \frac{2\pi}{L} \stackrel{!}{=} 1 \,, \quad (8.64)$$

from which it follows that $A_\mathrm{max} = \frac{L}{2\pi}$. Figure 8.6 plots the relative errors of velocity and pressure for a coarse mesh with 4^2 elements and polynomial degree $k = 3$ as a function of A/A_max. Adaptive time-stepping, equation (2.210), with $\mathrm{Cr} = 0.2$ is used. The error increases moderately for small amplitudes of the mesh deformation, and a rapid increase in errors can be observed around 85% of the theoretically maximum amplitude for this problem.

8.5.3 Robustness for under-resolved turbulent flows

In a last example, applicability of the present ALE-DG solvers to transitional and turbulent flows is studied by the example of the three-dimensional Taylor–Green vortex (TGV) problem (Taylor and Green 1937). The reader is referred to Section 2.6.7.2 for a description of the test case and the initial conditions. Both the standard setting $\mathrm{Re} = 1600$ and the inviscid limit $\mathrm{Re} \to \infty$ have been investigated for this test case. The simulated time interval is $0 \leq t \leq T = 20$. At start time, the domain $\Omega_0 = [-L/2, L/2]^3 = [-\pi, \pi]^3$ is a Cartesian box that deforms over time according to the mesh motion described in equation (8.63), with an amplitude of $A = \pi/6$ and varying mesh velocities characterized by period times decreasing from $T_\mathrm{G} = 20$ to $T_\mathrm{G} = 1$. The standard setup with periodic boundaries in all coordinate directions is used. The domain boundaries are moving for the given mesh motion, but the mesh deformation is defined periodically in order to ensure consistency with the periodic boundary conditions. An illustration of the mesh deformation for the above parameters is given in Figure 8.7. The mesh is originally Cartesian with $N_\mathrm{el} = (2^l)^3$ elements, where l denotes the level of refinement, and the effective mesh resolution follows the definition from Section 2.6.7.2. A BDF2 time integration scheme along with an explicit treatment of the convective term is used for all solver types, with $J_p = 2$ for the dual splitting



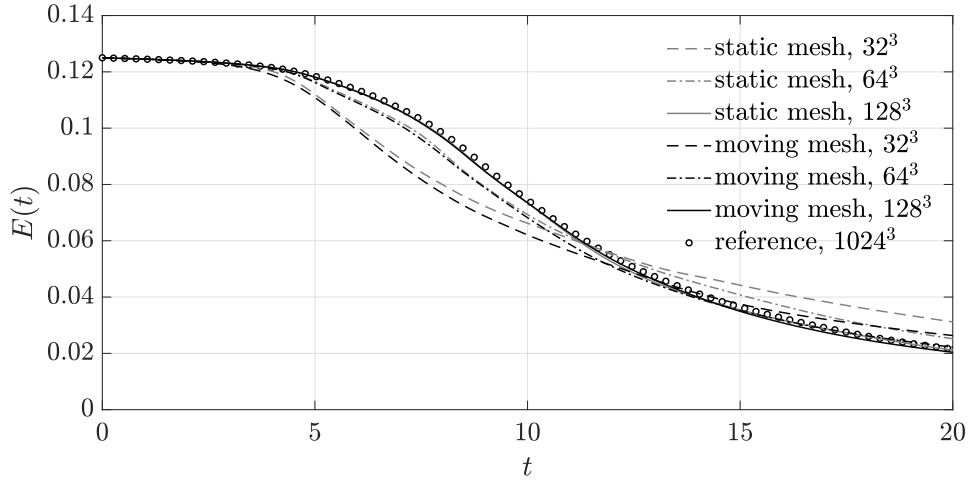
(a) $\|\boldsymbol{u}(\boldsymbol{x}, t = 0)\|$ (b) $\|\boldsymbol{u}(\boldsymbol{x}, t = 5)\|$ (c) $\|\boldsymbol{u}(\boldsymbol{x}, t = 10)\|$ (d) $\|\boldsymbol{u}(\boldsymbol{x}, t = 15)\|$ (e) $\|\boldsymbol{u}(\boldsymbol{x}, t = 20)\|$

Figure 8.7: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: visualization of velocity magnitude at different times for a spatial resolution with $l = 3$ and polynomial degree $k = 7$. The parameters of the mesh deformation are $A = \pi/6$ and $T_\mathrm{G} = 20$. The color map has been rescaled for each time instant, where red indicates high velocity and blue low velocity. The results shown have been simulated with the dual splitting scheme.

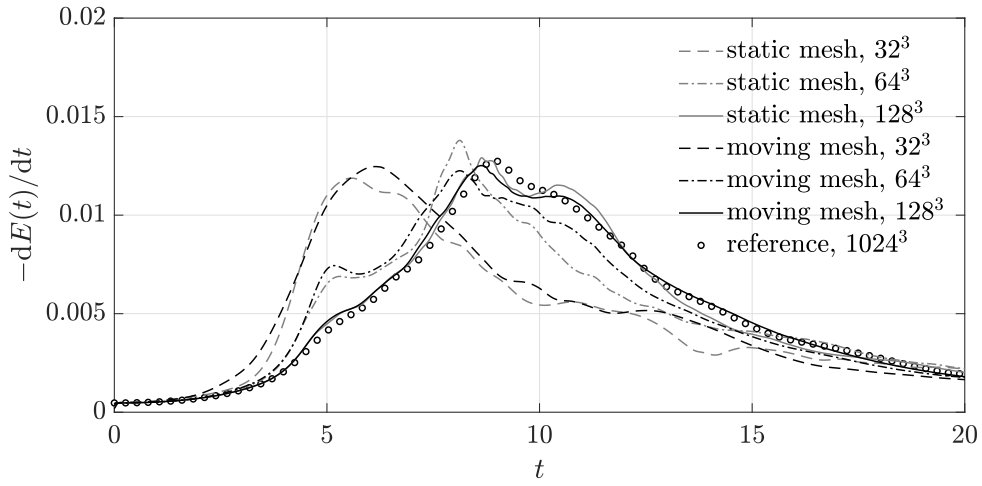
scheme and $J_p = 1$ for the pressure-correction scheme unless specified otherwise. Moreover, adaptive time-stepping, equation (2.210), with a Courant number of $\mathrm{Cr} = 0.2$ is used.

In a first set of experiments, the viscous case at $\mathrm{Re} = 1600$ is investigated. In Figure 8.8, results for the kinetic energy $E_\mathrm{k} = \int_{\Omega_h} \frac{1}{2}\boldsymbol{u}_h \cdot \boldsymbol{u}_h \mathrm{d}\Omega / \int_{\Omega_h} \mathrm{d}\Omega$ and the dissipation rate of the kinetic energy obtained on the moving mesh are compared to results on a static Cartesian mesh. A mesh refinement study for degree $k = 3$ is performed using a rather slow mesh motion with a period time of $T_\mathrm{G} = T$. The results converge towards the accurate DNS reference solution under mesh refinement, and the solution quality is comparable for static and moving meshes. Repeating this experiment for the strong formulation of the velocity–pressure coupling terms yields virtually the same results. In Figure 8.9, robustness w.r.t. the grid velocity is tested for the 64^3 mesh resolution by decreasing the period of the mesh motion down to $T_\mathrm{G} = T/20 = 1$, resulting in a very fast mesh motion. With increasing mesh velocity, an oscillating behavior can be observed in the kinetic energy dissipation rate where the frequency of these oscillations follows the mesh motion. However, the temporal evolution of the kinetic energy is almost indistinguishable for the different mesh velocities. It can be observed that the oscillations in the dissipation rate are very small in the beginning of the simulation where the solution is smooth, while the oscillations grow once the transition to a turbulent state took place. In Section 8.4.6, it was noted that the ALE transport term does not contribute to the energy evolution apart from the upwind stabilization term. However, this only holds under the assumption of exact numerical integration, which is not fulfilled on generally deformed geometries. A possible explanation for the results in Figure 8.9 could therefore be that integration errors (which are larger for non-smooth solutions or under-resolved scenarios) are amplified if the period of the mesh motion T_G tends to zero and the mesh velocity tends to infinity. This is supported by the observation that the oscillations are larger on coarser meshes. It should be mentioned that this experiment is performed here to test the robustness of the solver and that such a scenario (increasing the mesh velocity for a fixed fluid velocity) is not representative of a fluid–structure interaction problem for which the fluid has to follow the motion of the fluid–structure interface due to no-slip conditions.

In a second set of experiments, the inviscid Taylor–Green vortex problem is studied, which is considered one of the most challenging benchmark examples to test the robustness of a flow



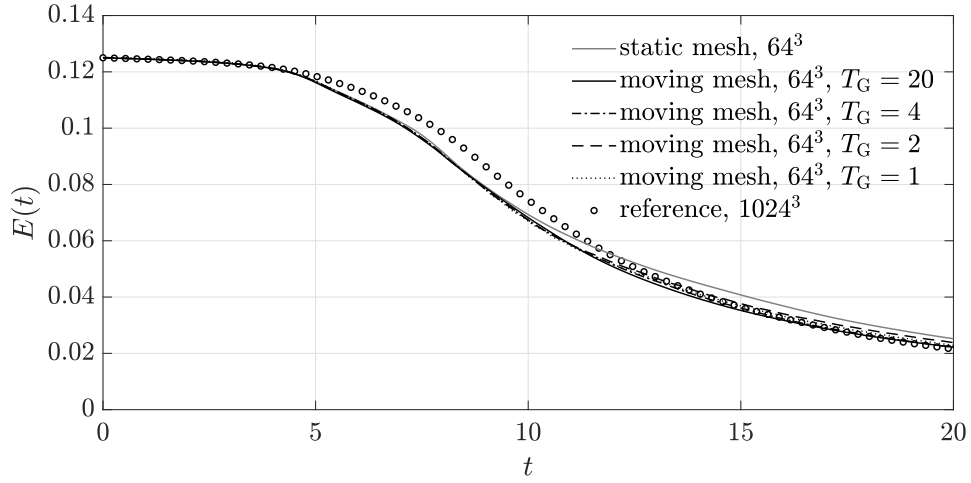
(a) Kinetic energy.



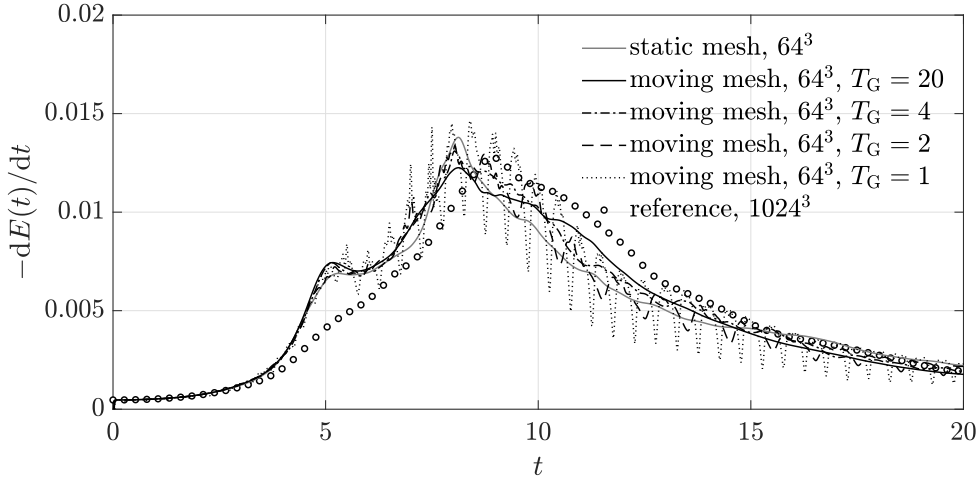
(b) Kinetic energy dissipation rate.

Figure 8.8: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: comparison between static and moving meshes for polynomial degree $k = 3$ for increasing effective mesh resolution of 32^3, 64^3, and 128^3. The results shown have been simulated with the dual splitting scheme.

solver for turbulent flows due to the absence of viscous dissipation as discussed in Chapter 7. Although the test case is academic, it can be expected that if a numerical method is robust for the inviscid Taylor–Green problem, it can also be successfully applied to practical, engineering problems. The results of detailed robustness tests for the inviscid TGV problem on moving meshes are reported in Fehn et al. (2021a), where it is found that the present stabilized DG approach achieves robustness also on moving meshes. These results are encouraging in the sense that the stabilized DG approach developed in Chapter 2 appears to be well designed, meaning that robustness carries over from static to moving meshes without having to adjust discretization



(a) Kinetic energy.



(b) Kinetic energy dissipation rate.

Figure 8.9: Taylor–Green vortex problem at $\mathrm{Re} = 1600$: comparison between static and moving meshes for polynomial degree $k = 3$ and effective resolution of 64^3 for increasing mesh velocity (decreasing period times of $T_\mathrm{G} = 20, 4, 2, 1$). The results shown have been simulated with the dual splitting scheme.

parameters. At the same time, it is emphasized again that it is unclear to which extent energy stability can be guaranteed theoretically for the present stabilized DG approach.

8.6 Conclusion and outlook

This chapter has presented ALE-DG methods for the incompressible Navier–Stokes equations that are up to third-order accurate in time and arbitrarily high-order accurate in space for sufficiently smooth problems. Moving mesh formulations are derived for both monolithic and split-

ting approaches based on a method-of-lines approach, considering both implicit and explicit formulations of the convective term. The time integration framework relies on BDF and extrapolation schemes and extends naturally to adaptive time-stepping. Stable and high-order accurate boundary conditions are derived for the splitting-type approaches. The ALE methods are designed to automatically fulfill the geometric conservation law. A key ingredient is the use of consistent divergence and continuity penalty terms to stabilize the method for under-resolved turbulent flows. Numerical results demonstrate optimality in terms of convergence rates and the geometric conservation law. Robustness and accuracy of the proposed methods have been investigated for under-resolved turbulent flows showing convincing properties for high-Reynolds-number flows. An important aspect is that the proposed methods are simple to implement since the equations are solved on the deformed geometry, i.e., the generic finite element software takes care of the mapping and the geometry terms, and only one instance of the mesh is stored at a time. To obtain computationally efficient algorithms, fast matrix-free evaluation techniques are applied for all parts of the Navier–Stokes solvers as in the Eulerian case. Even though not discussed here, ALE formulations have been implemented for the coupled flow–transport problem that has been the subject of Chapter 3. Chapter 9 shows applicability of the moving mesh approach developed here to problems of fluid–structure interaction. Another prominent field of applications that could be studied as part of future work are free surface flows.

9 Multiphysics application: fluid–structure interaction

This chapter develops a novel matrix-free partitioned FSI solver based on a three-field formulation, using an ALE-DG method for the fluid sub-problem modeled as an incompressible Newtonian fluid and a standard H^1-conforming finite element method for the structural problem in Lagrangian description modeled as a d-dimensional continuum with large deformations and hyperelastic material behavior. A standard Poisson-type or elasticity-type approach is used for the mesh deformation problem of the fluid domain. The main novelity of this approach is (i) the use of efficient matrix-free implementation techniques for all fields involved in the FSI problem, and (ii) the flexibility of this new formulation supporting non-matching meshes at the fluid–structure interface and the use of polynomial degrees that can be chosen independently for the fluid, solid, and mesh motion problems. The primary aim of this chapter is to demonstrate applicability of the incompressible Navier–Stokes DG solver developed in this thesis to multiphysics problems such as fluid–structure interaction, rather than a holistic discussion of fluid–structure interaction solvers. For this reason, the focus is on so-called black-box partitioned FSI solvers in the present work, in a fashion similar to how other external libraries for multiphysics problems would apply the incompressible fluid solver developed in the course of this thesis. In a similar direction, the aim of this chapter is to outline the algorithmic and computational properties of the FSI solver rather than extensive validation.

9.1 Motivation

9.1.1 State-of-the-art

Methods for the numerical solution of fluid–structure interaction problems can be categorized into monolithic and partitioned approaches. Monolithic approaches solve one global system of nonlinear equations for all involved unknowns, typically by a Newton–Krylov approach with suitable preconditioner for the linearized problem, e.g., by exploiting the structure of the equations and using multigrid preconditioners for the diagonal blocks, which then form the building blocks of the preconditioner for the monolithic system. Different multigrid variants and block preconditioners are possible, and the reader is referred to Aulisa et al. (2018), Barker and Cai (2010), Forti et al. (2017), Gee et al. (2011), Heil (2004), Hron and Turek (2006), Jodlbauer et al. (2019), Kong and Cai (2018), Langer and Yang (2016), Mayr et al. (2015, 2020), Muddle et al. (2012), Richter (2015) and references therein for a discussion of monolithic solution approaches.

A competing approach is the partitioned solution of fluid–structure interaction problems, where the fluid and solid sub-problems are solved separately and where the equilibrium (interface coupling conditions) between the fluid and solid domains is realized by appropriate

boundary conditions on the fluid–structure interface. The partitioned technique is attractive for several reasons, most importantly the natural support for modular code development, the reduction of complexity through a separation of concerns, flexibility of plugging together those software tools most efficient for a specific sub-task through the black-box principle, and the associated agility in adapting to new developments (discretization techniques, numerical linear algebra, hardware and implementation techniques). The most common partitioned strategy is the Dirichlet–Neumann partitioning scheme that imposes Dirichlet boundary conditions for the fluid and Neumann boundary conditions for the solid at the fluid–structure interface. From an algebraic point of view, this solution technique can be interpreted as a block Gauss–Seidel iteration. Due to the well-known artificial added-mass effect (Causin et al. 2005, Förster et al. 2007, van Brummelen 2009) and its associated instabilities for incompressible flows, it is necessary to drive the coupled FSI problem into a converged state by an (outer) iterative procedure around the single-field solvers. A simple Gauss–Seidel iteration without relaxation or other techniques for convergence-acceleration may not converge, so that several different techniques have been formulated as a result of major research efforts over the last two decades. One can distinguish between fixed-point iterations with constant relaxation (Le Tallec and Mouro 2001) or dynamic relaxation (Küttler and Wall 2008, Mok and Wall 2001), vector-extrapolation methods (Küttler and Wall 2009), block-Newton methods (Matthies and Steindorf 2002), interface Newton methods with exact Jacobian (Fernández and Moubachir 2005), interface Jacobians approximated by finite-differences or by simplified physical models (Gerbeau and Vidrascu 2003, Gerbeau et al. 2005), and the category of black-box quasi-Newton methods (Bogaers et al. 2014, Degroote et al. 2009, Haelterman et al. 2016, Lindner et al. 2015, Spenke et al. 2020) primarily motivated from a numerical linear algebra perspective. In the present work, the focus is on relaxation-type and quasi-Newton partitioned black-box FSI solvers. While the above approaches are used most often, other techniques for convergence acceleration of strongly-coupled partitioned FSI problems have been examined as well in the literature. From a structural perspective, fictitious mass and damping has been used in Baek and Karniadakis (2012) in combination with Aitken relaxation. From a fluid perspective, the beneficial effect of (weak) compressibility on iteration counts in combination with Aitken relexation has been shown in La Spina et al. (2020a).

A class of methods that share characteristics of both monolithic schemes and partitioned schemes are so called semi-implicit (or projection/splitting) schemes. The idea is to solve only those terms of the equations in a strongly-coupled way that are relevant to prevent artificial added-mass instabilities, and to solve other terms (preferably expensive nonlinear terms) only once per time step. In this context, a projection scheme has been proposed in Fernández et al. (2007) that treats the mesh motion problem and the momentum equation of the fluid explicitly and the pressure-structure coupling implicitly. The splitting is introduced on the level of differential equations in Fernández et al. (2007), and similar algebraic splitting schemes are presented in Badia et al. (2008b). While the method of Fernández et al. (2007) naturally includes nonlinearities of the structure, the work by Badia et al. (2008b) assumes a linear structural problem and an extension to the non-linear case remains unclear. The pressure-structure problem can be solved in a monolithic way or in a strongly-coupled partitioned way. However, these schemes can not be realized in a black-box environment, and are therefore not considered here.

To overcome convergence problems of the classical Dirichlet–Neumann partitioning for problems with strong artificial added-mass effect, alternative Robin–Neumann and Robin–Robin partitioning schemes have been developed (Badia et al. 2008a, Nobile and Vergara 2008), which

have been reported to converge without under-relaxation and to be robust w.r.t. the density ratio. These works also discuss the use of semi-implicit versus implicit schemes mentioned above. The Robin–Neumann method is applied in Wang et al. (2018) in the context of discontinous Galerkin discretizations. In Serino et al. (2019), a partitioned scheme for incompressible flows interacting with a linear structure (integrated explicitly in time) is constructed by the use of special Robin-type boundary conditions and it is stated that this solver is stable without sub-iterations and independently of the density ratio. While these methods exhibit interesting stability properties, the required boundary conditions are in general not available for black-box field solvers, which is why this approach is also not considered in the present work.

9.1.2 Partitioned versus monolithic approaches

The aspects of robustness and computational efficiency of partitioned approaches when compared to the monolithic approach are discussed controversially in the literature. Several previous works already aimed at comparative studies (Badia et al. 2008c, Degroote et al. 2009, Gee et al. 2011, Heil et al. 2008, Küttler et al. 2010, Sheldon et al. 2014), mostly with the outcome that monolithic solvers are not only more robust in general, but also computationally more efficient especially in the strongly-coupled regime of large density ratios $\rho^{\mathcal{F}}/\rho^{\mathcal{S}}$. However, it should be emphasized that a conclusion regarding computational efficiency is still not within reach to the best of the author's knowledge. On the one hand, this is due to the continuous developments made in various fields in combination with the snapshot character of comparative studies. On the other hand, these comparative studies usually apply the same (potentially computationally expensive, matrix-based) preconditioners or linear solvers for the single-field problems in partitioned solvers that also form the building blocks of the preconditioner or linear solver for the monolithic approach. However, this understanding of a "fair comparison" leads to a conservative selection of solvers dictated by the most complex algebraic system of equations (in this case the monolithic system that is highly ill-conditioned, see Richter (2015)). Some comparative studies even use direct solvers. Note that this strategy might render comparative studies inconclusive since it does not exploit the full optimization potential of partitioned schemes. In fact, a main motivation for the use of partitioned approaches is that they can exploit those techniques most suitable and efficient for a certain sub-task. This includes the use of fast preconditioners tailored to the different single-field problems. Furthermore, partitioned approaches naturally allow to apply the partitioned paradigm in a nested way, e.g., by solving the fluid sub-problem with fast projection-type solvers and mixed implicit/explicit time-stepping, an approach that is considered computationally efficient especially for high-Reynolds-number turbulent flows. Partitioned FSI solvers currently appear to be the predominant solution strategy in the application field of turbulent flows, see e.g. Breuer et al. (2012), Lorentzon and Revstedt (2020). In this context, different iteration procedures have been proposed (Badia et al. 2008b, Breuer et al. 2012, Fernández et al. 2007) with the goal to solve expensive parts such as the ALE mesh motion or the fluid convective and viscous terms only once per time step. As already introduced above, these methods are known as semi-implicit schemes, originally proposed by Fernández et al. (2007). They solve some terms of the equations in a weakly-coupled (or explicit) sense and others, which are relevant for the added-mass effect, in a strongly-coupled (or implicit) sense, involving partitioned solvers not only on the FSI coupling level but also on the fluid level.

Table 9.1: Performance of state-of-the-art monolithic fluid–structure interaction solvers in terms of throughput per time step in $\frac{\text{DoF}}{\text{s·core}}$ measured for the 3D pressure wave benchmark problem (see Section 9.7.2) and published within the last decade. The table additionally lists the range of problem sizes (in total number of degrees of freedom) and the time step size considered in each study. All studies used low-order finite element discretizations of degree one or two and matrix-based preconditioners.

Study	N_{DoFs}	Δt in s	Throughput in $\frac{\text{DoF}}{\text{s·core}}$
(Gee et al. 2011, Table 5)	0.024M–3.1M	10^{-4}	$350 - 700$
(Langer and Yang 2016, Figures 4–7)	0.023M–0.11M	$\{1.25, 2.5\} \cdot 10^{-4}$	$25 - 100$
(Forti et al. 2017, Figure 5.3)	12M–48M	10^{-4}	$600 - 700$
(Kong and Cai 2018, Tables 1–2)	7.8M–480M	10^{-3}	$800 - 1400$
(Mayr et al. 2020, Table 4)	0.12M–1.9M	10^{-4}	$350 - 1500$

In the following, an attempt is made to get an impression of the computational efficiency of state-of-the-art matrix-based monolithic FSI solvers. For this purpose, data is collected from the literature and summarized in Table 9.1. The efficiency metric is the throughput in degrees of freedom solved per second of wall-time and per compute core in one time step. To ensure comparability of the results published for different FSI solvers, all throughput numbers listed in Table 9.1 refer to the pressure wave benchmark which is also studied in the present work in Section 9.7.2. It should be emphasized that the throughput further depends on discretization parameters such as the mesh size h (or problem size in degrees of freedom) and the time step size Δt. Moreover, the hardware under consideration and the chosen solver tolerances have an impact on throughput. Nevertheless, it might be possible to extract some trends from these numbers. In particular, these numbers suggest that state-of-the-art monolithic solvers are currently not able to significantly exceed a throughput of approximately $10^3 \frac{\text{DoF}}{\text{s·core}}$ per time step for the considered benchmark problem.

Given the multitude of partitioned schemes, numerous comparative studies on the relative performance of partitioned schemes can be found in the literature, see Bogaers et al. (2014), Degroote et al. (2010), Küttler and Wall (2008, 2009), Küttler et al. (2010), Lindner et al. (2015), Spenke et al. (2020). Unfortunately, it has not been possible to extract numbers from the literature with a similar level of detail as for the monolithic schemes listed in Table 9.1. Many works only report normalized or relative costs, see for example Bogaers et al. (2014), Degroote et al. (2010), Spenke et al. (2020), or only report iteration counts, see for example Lindner et al. (2015), Mehl et al. (2016). Hence, the performance of partitioned schemes is estimated from a different perspective in the following. For single-field problems, previous chapters have shown that a throughput of $10^5 - 10^6 \frac{\text{DoF}}{\text{s·core}}$ can be reached for high-order DG discretizations on modern hardware, see also Fehn et al. (2020) for a Poisson-type model problem and Arndt et al. (2020b) for an incompressible, turbulent flow problem. The most effective partitioned FSI solvers are reported to typically converge in $5 - 10$ outer iterations for strongly-coupled FSI problems (Bogaers et al. 2014, Spenke et al. 2020) and lower iteration counts for weakly-coupled problems

with $\rho^{\mathcal{F}}/\rho^{\mathcal{S}} \ll 1$. Assuming that partitioned schemes require $\mathcal{O}(10)$ outer FSI iterations with single-field solvers called in each of them, one might expect a throughput of $10^4 - 10^5 \frac{\text{DoF}}{\text{s}\cdot\text{core}}$ for the FSI problem. This would imply a huge gap in throughput of at least one order of magnitude when comparing these estimated values for partitioned schemes to the performance of monolithic solvers listed in Table 9.1.[1] Based on these considerations, one might expect that it is possible to implement partitioned FSI solvers that significantly improve the throughput compared to the monolithic solvers listed in Table 9.1, e.g., by the use of fast matrix-free solvers. These considerations serve as a main motivation for the present work, which aims at overcoming the seemingly stagnating performance of state-of-the-art monolithic solvers. While the present work exclusively considers partitioned solvers, it should be mentioned that alternative optimal-complexity matrix-free implementations of monolithic solvers might also improve the performance significantly. The development of fast matrix-free preconditioners for the monolithic system appears to be the key ingredient in this context, but this aspect is beyond the scope of the present work. The above performance numbers and estimates should therefore not be misunderstood as claiming a superior performance of partitioned over monolithic schemes. Instead, these numbers point to potential performance gains by the use of an efficient implementation, which are first explored for partitioned schemes in the present thesis, and could also be explored for monolithic schemes in the future. Of course, a discussion regarding computational efficiency becomes obsolete if a partitioned scheme is not robust enough to converge for a given problem.

9.1.3 Mesh movement techniques

For ALE-based FSI, a key ingredient is the technique used for the ALE mesh motion of the fluid domain, where various formulations can be used. The most simple technique is an explicit mapping or interpolation of the moving domain boundary at the fluid–structure interface into the interior of the fluid domain, e.g., by radial basis function interpolation (de Boer et al. 2007) or transfinite interpolation (Wang and Przekwas 1994), see also Froehle and Persson (2014), Heil (2004), Persson et al. (2007) for applications in the context of FSI. Another class of mesh motion techniques are spring models, see for example Batina (1991), Farhat et al. (1998). The most widely used approach for general fluid–structure interaction problems, however, is the solution of a PDE-type model problem for the mesh deformation, where the solution of a Laplace problem (harmonic extension) (Kanchi and Masud 2007, Löhner and Yang 1996), a linear elasticity problem (Johnson and Tezduyar 1994, Sackinger et al. 1996, Stein et al. 2003), a non-linear elasticity problem (Froehle and Persson 2015, Neunteufel and Schöberl 2021), or a biharmonic equation (Helenbrook 2003) form the main categories of such an approach. Comparative studies are for example conducted in Wick (2011), Yang and Mavriplis (2005), Yigit et al. (2008).

In order to obtain a generic and versatile FSI solver, the present work exclusively considers the PDE-type approach for mesh movement. The solution of a constant-coefficient Laplace equation is typically used in hemodynamics (with the simulation of pressure waves as the most prominent benchmark scenario), where the deformations are moderate and good-natured in the sense that the domain is mainly enlarged in the direction normal to the blood vessels without significantly

[1] Although a partitioned FSI solver involves several single-field problems, this does not necessarily imply a reduction in throughput, since the degrees of freedom of the single-field problems sum up to the overall number of unknowns of the FSI problem. A single-field problem with a very low throughput does not deteriorate overall throughput significantly as long as this single-field problem has a low share of the overall computational costs.

distorting the topology. However, for problems with large deformations of the structure, it is well known that the solution of a Laplace problem mainly introduces mesh deformation close to the interface and soon leads to inverted elements. As a means to smear the deformation imposed at the fluid–structure interface over a wider part of the domain, the common strategy to tackle this problem is the use of spatially varying coefficients with a diffusivity (in case of the heat equation model problem) or stiffness (in case of the elasticity model problem) that increases with decreasing distance to the structure. The spatial variation of the variable coefficient can be prescribed analytically based on the reference configuration or in a more generic way by determining the distance to the moving interface and using a purely distance-based description of the variable coefficient. Another option is to select the coefficient inversely proportional to the element volume, as used in Johnson and Tezduyar (1994), Stein et al. (2003) in the context of low-order finite element methods, assuming that elements are smaller close to the fluid–structure interface. However, there are scenarios conceivable where this assumption might not be satisfied or might be too restrictive. Especially for high-order spectral element methods, element size is no longer a good measure.

Remark 9.1 *Given that the use of a variable coefficient or stiffness appears to be the essential ingredient enabling large mesh deformations, it remains unclear from previous works whether and to which extent the more complex elasticity (pseudo-solid) model problems give an advantage over the more simple Laplacian (harmonic) model. It remains also unclear how the biharmonic model performs when compared to a second-order PDE with variable coefficient. These aspects could be part of future investigations.*

9.1.4 Discontinuous Galerkin methods for fluid–structure interaction

Application of discontinuous Galerkin discretization techniques to fluid–structure interaction problems with compressible fluid formulations have already been proposed in Froehle and Persson (2014), Persson et al. (2007), where explicit time stepping is used in Persson et al. (2007) and semi-implicit time stepping in Froehle and Persson (2014). An FSI formulation for weakly compressible flows with HDG discretization for the fluid problem and continuous Galerkin discretization for the structural problem is presented in La Spina et al. (2020b). In the context of the incompressible Navier–Stokes equations, the development of DG-FSI methods is still in an early stage, explainable by the fact that the single-field fluid problem required a lot of attention in providing robust and efficient numerical methods, where significant progress has been made over the last two decades, as discussed in Chapter 2. ALE-based monolithic FSI solvers are proposed in Neunteufel and Schöberl (2021), Sheldon et al. (2016), where hybridizable discontinuous Galerkin methods are used. While the work by Sheldon et al. (2016) uses an HDG formulation for all three fields, the work by Neunteufel and Schöberl (2021) uses an $H(\mathrm{div})$-conforming HDG discretization for the fluid and a standard H^1-conforming discretization for the nonlinear elasticity problems of the solid domain and the mesh deformation of the fluid domain. A partitioned FSI solver using L^2-conforming discontinuous Galerkin discretizations for the fluid and the harmonic mesh motion equation is presented in Wang et al. (2018). For the structure, shell models are considered which are again discretized by DG methods. The FSI coupling is realized by a strongly-coupled partitioned approach and the dual splitting projection scheme is used as

solution technique for the incompressible Navier–Stokes sub-problem. All of these methods use conforming meshes at the fluid–structure interface. Another major limitation of these previous works is that they all rely on matrix-based implementations, rendering the numerical method memory intensive and computationally expensive, especially for three-dimensional problems and high polynomial degrees.

9.1.5 Novel contributions and scope of this work

The FSI solvers developed in this work build upon the ALE-DG formulations for incompressible flows on moving domains developed in Chapter 8. The structure is modeled as d-dimensional non-linear elasticity problem with hyperelastic material model. PDE-type model problems are considered for the mesh movement problem on the fluid domain, allowing the use of a harmonic or (non-)linear elasticity mesh motion problem. The solid problem and mesh movement problem are discretized by standard H^1-conforming discretizations. The fluid–structure coupling is realized by strongly-coupled partitioned solution techniques with the goal to obtain computationally efficient solution techniques. The present work exclusively considers black-box coupling techniques that do not require access to individual field solvers apart from controlling the time step size (for adaptive time stepping) and querying or relaxing solution vectors. In this context, the present work relies on state-of-the-art methods such as a fixed-point iteration with Aitken relaxation or quasi-Newton algorithms for convergence acceleration. All fields of the three-field formulation are implemented with the intention to be able to use fast matrix-free evaluation techniques, but the FSI solver should of course make use of matrix-based techniques where reasonable. For reasons of flexibility, the FSI solvers developed in this work are designed to allow non-conforming meshes at the fluid–structure interface and polynomial degrees that can be selected independently for the fluid, mesh motion, and solid problems. This design naturally enables the use of different incompressible Navier–Stokes solution techniques and time integration schemes developed in Chapter 2 as well as adaptive time stepping. As discussed in Section 9.1.4, many state-of-the-art FSI approaches using DG formulations for the fluid problem do not provide this flexibility. To the best of the author's knowledge, the present work is the first that applies fast matrix-free DG implementations for incompressible flows in an FSI context. Performance comparisons to a state-of-the-art matrix-based monolithic FSI solver based on low-order continuous finite element discretizations (see Mayr et al. (2020)) indicate substantial performance improvements by the ingredients proposed in this thesis. The present implementation does currently not support hp-adaptivity in the individual fields or remeshing of the fluid mesh, where the latter might in general be necessary in case of large deformations of the structure.

9.2 Mathematical model in strong formulation

The domain $\Omega(t)$ consists of a fluid part $\Omega^{\mathcal{F}}(t)$ and a structure part $\Omega^{\mathcal{S}}(t)$ with $\Omega(t) = \Omega^{\mathcal{F}}(t) \cup \Omega^{\mathcal{S}}(t)$ and $\Omega^{\mathcal{F}}(t) \cap \Omega^{\mathcal{S}}(t) = \emptyset$. Both domains evolve as a function of time, and for ease of notation the time argument will also be skipped in the following. Both domains share a common interface $\Gamma^{\mathrm{I}}(t) = \partial\Omega^{\mathcal{F}}(t) \cap \partial\Omega^{\mathcal{S}}(t)$, the fluid–structure interface. The boundary of the fluid domain $\partial\Omega^{\mathcal{F}}(t)$ can then be written as $\Gamma^{\mathcal{F}}(t) = \Gamma^{\mathrm{D},\mathcal{F}}(t) \cup \Gamma^{\mathrm{N},\mathcal{F}}(t) \cup \Gamma^{\mathrm{I}}(t)$ (with no intersection between the Dirichlet, Neumann, and fluid–structure interface parts), and likewise for the

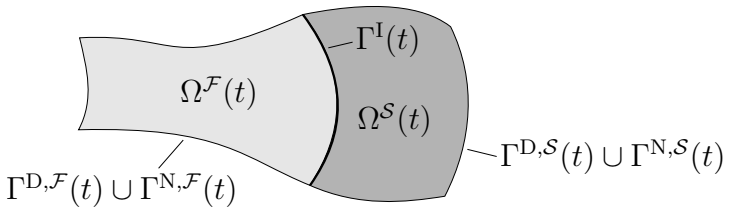


Figure 9.1: Illustration of domains and boundaries for fluid–structure interaction problems.

structure $\Gamma^{\mathcal{S}}(t) = \Gamma^{\mathrm{D},\mathcal{S}}(t) \cup \Gamma^{\mathrm{N},\mathcal{S}}(t) \cup \Gamma^{\mathrm{I}}(t)$. An illustration is given in Figure 9.1. The FSI problem can be formulated as the balance equations for the fluid and structure sub-problems, along with coupling conditions at the interface. The description below is restricted to an arbitrary Lagrangian–Eulerian formulation for the fluid ($\mathcal{F}$), and a Lagrangian formulation for the structure ($\mathcal{S}$). Due to the use of an ALE formulation for the fluid, an additional equation has to be solved for the mesh motion ($\mathcal{M}$) on the fluid domain, leading to a so-called three-field formulation ($\mathcal{M}, \mathcal{F}, \mathcal{S}$).

9.2.1 Fluid

The governing equations for the fluid domain are the incompressible Navier–Stokes equations in ALE formulation for a Newtonian fluid (see also Chapter 8)

$$\left.\frac{\partial \boldsymbol{u}^{\mathcal{F}}}{\partial t}\right|_{\boldsymbol{\chi}} + ((\boldsymbol{u}^{\mathcal{F}} - \boldsymbol{u}^{\mathcal{M}}) \cdot \nabla)\boldsymbol{u}^{\mathcal{F}} - \nabla \cdot \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{\mathcal{F}}) + \nabla p^{\mathcal{F}} = \boldsymbol{f}^{\mathcal{F}} \ \text{ in } \Omega^{\mathcal{F}} \times [0,T] \,, \tag{9.1}$$

$$\nabla \cdot \boldsymbol{u}^{\mathcal{F}} = 0 \quad \text{ in } \Omega^{\mathcal{F}} \times [0,T] \,, \tag{9.2}$$

where $\boldsymbol{u}^{\mathcal{F}}$ denotes the fluid velocity, $\boldsymbol{u}^{\mathcal{M}}$ the mesh velocity, $p^{\mathcal{F}}$ the kinematic pressure, $\boldsymbol{f}^{\mathcal{F}}$ the body force vector, $\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{\mathcal{F}})$ the viscous term written in Laplace or divergence formulation (see Chapter 2), and $\boldsymbol{\chi}$ the coordinates of the moving ALE frame as introduced in Chapter 8. The strong formulation is completed by the initial condition

$$\boldsymbol{u}^{\mathcal{F}}(\boldsymbol{x}, t=0) = \boldsymbol{u}_0^{\mathcal{F}}(\boldsymbol{x}) \ \text{ in } \Omega_0^{\mathcal{F}} \,, \tag{9.3}$$

and the boundary conditions

$$\boldsymbol{u}^{\mathcal{F}} = \boldsymbol{g}_u^{\mathcal{F}} \ \text{ on } \Gamma^{\mathrm{D},\mathcal{F}}(t) \,, \tag{9.4}$$

$$\left(\boldsymbol{F}_{\mathrm{v}}(\boldsymbol{u}^{\mathcal{F}}) - p^{\mathcal{F}}\boldsymbol{I}\right) \cdot \boldsymbol{n} = \boldsymbol{h}^{\mathcal{F}} \ \text{ on } \Gamma^{\mathrm{N},\mathcal{F}}(t) \,, \tag{9.5}$$

where $\boldsymbol{n}$ is the outward pointing unit normal vector and $\boldsymbol{I}$ the identity matrix. Dirichlet and Neumann boundary values are denoted by $\boldsymbol{g}_u^{\mathcal{F}}$ and $\boldsymbol{h}^{\mathcal{F}}$, respectively. The initial condition $\boldsymbol{u}_0^{\mathcal{F}}(\boldsymbol{x})$ is divergence-free and fulfills the velocity Dirichlet boundary condition.

9.2.2 Structure

For a general description of nonlinear solid mechanics the reader is referred to Holzapfel (2000). The structural problem is governed by the equations of non-linear elasticity in Lagrangian for-

mulation

$$\rho_0^{\mathcal{S}} \frac{\mathrm{d}^2 \boldsymbol{d}^{\mathcal{S}}}{\mathrm{d}t^2} - \nabla_0 \cdot \boldsymbol{P} = \boldsymbol{b}_0^{\mathcal{S}} \ \text{ in } \Omega_0^{\mathcal{S}} \times [0,T] \ , \tag{9.6}$$

where $\rho_0^{\mathcal{S}}$ denotes the density in the initial/reference configuration $\Omega_0^{\mathcal{S}} = \Omega^{\mathcal{S}}(t=0)$, $\boldsymbol{b}_0^{\mathcal{S}}$ the body force per unit undeformed volume, and $\boldsymbol{P}$ the first Piola–Kirchhoff stress tensor

$$\boldsymbol{P} = \boldsymbol{F} \cdot \boldsymbol{S} \ , \tag{9.7}$$

with deformation gradient $\boldsymbol{F}$

$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \boldsymbol{1} + \nabla_0 \boldsymbol{d}^{\mathcal{S}} \ , \tag{9.8}$$

and second Piola–Kirchhoff stress tensor $\boldsymbol{S}$. In the above equations, ∇_0 denotes the gradient with respect to the material coordinates $\boldsymbol{X}$, and $\boldsymbol{x}^{\mathcal{S}}(\boldsymbol{X},t) = \boldsymbol{X} + \boldsymbol{d}^{\mathcal{S}}(\boldsymbol{X},t)$ the location of a material point in deformed configuration where $\boldsymbol{d}^{\mathcal{S}}(\boldsymbol{X},t)$ is the displacement vector. The present work considers hyperelastic materials, for which the second Piola–Kirchhoff stress is given as $\boldsymbol{S} = \partial\Psi/\partial\boldsymbol{E}$ as a function of the strain energy function Ψ and the Green–Lagrange strain tensor $\boldsymbol{E}$. From this class of materials, the isotropic St. Venant–Kirchhoff constitutive relation is considered here, which is often used for problems with small strains

$$\boldsymbol{S} = \lambda^{\mathcal{S}} \operatorname{tr}(\boldsymbol{E})\, \boldsymbol{I} + 2\mu^{\mathcal{S}} \boldsymbol{E} \ , \tag{9.9}$$

where the Green–Lagrange strain tensor $\boldsymbol{E}$ is

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{F}^{\mathsf{T}} \cdot \boldsymbol{F} - \boldsymbol{I}\right) \ , \tag{9.10}$$

and where the Lamé coefficients $\lambda^{\mathcal{S}}$ and $\mu^{\mathcal{S}}$ are given as

$$\lambda^{\mathcal{S}} = \frac{\nu^{\mathcal{S}} E^{\mathcal{S}}}{(1+\nu^{\mathcal{S}})(1-2\nu^{\mathcal{S}})} \ , \ \mu^{\mathcal{S}} = \frac{E^{\mathcal{S}}}{2(1+\nu^{\mathcal{S}})} \ , \tag{9.11}$$

with Young's modulus $E^{\mathcal{S}}$ and Poisson's ratio $\nu^{\mathcal{S}} < \frac{1}{2}$. The St. Venant–Kirchhoff material law can be alternatively formulated as $\boldsymbol{S} = \boldsymbol{C} : \boldsymbol{E}$ with a fourth-order material tensor $\boldsymbol{C}$, highlighting the linear relationship between the second Piola–Kirchhoff stress tensor and the Green–Lagrange strain tensor. The strong formulation for the solid domain is completed by initial conditions for the displacement $\boldsymbol{d}^{\mathcal{S}}$ and the velocity $\boldsymbol{v}^{\mathcal{S}} = \mathrm{d}\boldsymbol{d}^{\mathcal{S}}/\mathrm{d}t$

$$\boldsymbol{d}^{\mathcal{S}}(\boldsymbol{x}, t=0) = \boldsymbol{d}_0^{\mathcal{S}}(\boldsymbol{x}) \ \text{ in } \Omega_0^{\mathcal{S}} \ , \tag{9.12}$$

$$\boldsymbol{v}^{\mathcal{S}}(\boldsymbol{x}, t=0) = \boldsymbol{v}_0^{\mathcal{S}}(\boldsymbol{x}) \ \text{ in } \Omega_0^{\mathcal{S}} \ , \tag{9.13}$$

and boundary conditions

$$\boldsymbol{d}^{\mathcal{S}} = \boldsymbol{g}^{\mathcal{S}} \ \text{ on } \Gamma^{\mathrm{D},\mathcal{S}}(t) \ , \tag{9.14}$$

$$\boldsymbol{P} \cdot \boldsymbol{N} = \boldsymbol{t}_0^{\mathcal{S}} \ \text{ on } \Gamma^{\mathrm{N},\mathcal{S}}(t) \ , \tag{9.15}$$

where $\boldsymbol{g}^{\mathcal{S}}$ denotes the displacement prescribed at the Dirichlet boundary, $\boldsymbol{N}$ the outward-pointing unit normal vector in reference configuration, and $\boldsymbol{t}_0^{\mathcal{S}}$ the traction force per unit undeformed area. The traction $\boldsymbol{t}_0^{\mathcal{S}}$ in Lagrangian description is related to the traction $\boldsymbol{t}^{\mathcal{S}}$ in the deformed configuration via

$$\boldsymbol{t}_0^{\mathcal{S}} = \frac{\mathrm{d}a}{\mathrm{d}A}\boldsymbol{t}^{\mathcal{S}} \ , \ \frac{\mathrm{d}a}{\mathrm{d}A} = \| \det(\boldsymbol{F}) \ \boldsymbol{F}^{-\mathsf{T}} \cdot \boldsymbol{N} \| \ . \tag{9.16}$$

Similarly, it holds for the body force $\boldsymbol{b}_0^{\mathcal{S}}$

$$\boldsymbol{b}_0^{\mathcal{S}} = \frac{\mathrm{d}v}{\mathrm{d}V} \ \boldsymbol{b}^{\mathcal{S}} \ , \ \frac{\mathrm{d}v}{\mathrm{d}V} = \det(\boldsymbol{F}) \ , \tag{9.17}$$

where $\boldsymbol{b}^{\mathcal{S}}$ is the body force per deformed volume. For example, one obtains $\boldsymbol{b}_0^{\mathcal{S}} = \rho_0^{\mathcal{S}}\boldsymbol{g}$ for the gravitational force.

9.2.3 Coupling conditions

Assuming no-slip conditions at the fluid–structure interface leads to the kinematic coupling condition

$$\boldsymbol{u}^{\mathcal{F}} = \frac{\mathrm{d}\boldsymbol{d}^{\mathcal{S}}}{\mathrm{d}t} \text{ on } \Gamma^{\mathrm{I}}(t) \ , \tag{9.18}$$

i.e., material points of the fluid and structure continua are glued together at the interface for all times. The kinetic (or dynamic) coupling condition enforces a balance of traction forces at the interface

$$\boldsymbol{\sigma}^{\mathcal{F}} \cdot \boldsymbol{n}^{\mathcal{F}} + \boldsymbol{\sigma}^{\mathcal{S}} \cdot \boldsymbol{n}^{\mathcal{S}} = \boldsymbol{0} \text{ on } \Gamma^{\mathrm{I}}(t) \ , \tag{9.19}$$

where $\boldsymbol{\sigma}$ denotes the Cauchy stress tensor and where the outward pointing unit normal vectors fulfill the relation $\boldsymbol{n}^{\mathcal{F}} = -\boldsymbol{n}^{\mathcal{S}}$.

9.2.4 Mesh motion

In Chapter 8, the mesh motion has been prescribed analytically by a map $\boldsymbol{f}_{\mathrm{G}}(\boldsymbol{\chi}, t)$ that can be described mathematically as a homeomorphism. In the context of FSI, the mesh motion problem can be formulated as an extension of the displacement prescribed at the fluid–structure interface into the interior of the fluid domain. This extension is in principle arbitrary, but must prevent mesh-folding in the fluid domain. Introducing an extension operator $\mathrm{Ext} : \Gamma_0^{\mathrm{I}} \mapsto \Omega_0^{\mathcal{F}}$ as in Fernández and Moubachir (2005), the fluid mesh displacement can be written in an abstract way as

$$\boldsymbol{d}^{\mathcal{M}} = \mathrm{Ext}\left(\boldsymbol{d}^{\mathcal{S}}|_{\Gamma_0^{\mathrm{I}}}\right) \ . \tag{9.20}$$

An alternative formulation used in Gerbeau et al. (2005) is $\boldsymbol{d}^{\mathcal{M}} = \mathrm{tr}^{-1}\left(\boldsymbol{d}^{\mathcal{S}}|_{\Gamma_0^{\mathrm{I}}}\right)$, where tr^{-1} is an inverse trace operator. The present work considers harmonic and pseudo-solid extensions described by a Laplace problem and a linear or nonlinear elasticity problem, respectively, which is

solved for the fluid mesh displacement $\boldsymbol{d}^{\mathcal{M}}$. The fluid mesh position is then given as $\boldsymbol{x}^{\mathcal{M}}(\boldsymbol{\chi}, t) = \boldsymbol{\chi} + \boldsymbol{d}^{\mathcal{M}}(\boldsymbol{\chi}, t)$. The mesh coordinates $\boldsymbol{\chi}$ are introduced in Chapter 8 and can be seen as the material coordinates $\boldsymbol{X}$ of the mesh field in terms of the pseudo-solid analogy. For the harmonic extension, the mesh motion problem is described by the following Laplace problem with spatially varying coefficient $\alpha(\boldsymbol{\chi})$ formulated w.r.t. the undeformed fluid mesh

$$-\nabla_{\boldsymbol{\chi}} \cdot \left(\alpha(\boldsymbol{\chi}) \nabla_{\boldsymbol{\chi}} \boldsymbol{d}^{\mathcal{M}}\right) = \boldsymbol{0} \;\text{ in } \Omega_0^{\mathcal{F}} \times [0, T] \,. \tag{9.21}$$

The description of boundary conditions is postponed to the end of this section. The pseudo-solid problem described by the equations of linear elasticity reads

$$-\nabla_{\boldsymbol{\chi}} \cdot \left(\lambda^{\mathcal{M}}(\boldsymbol{\chi}) \operatorname{tr}\left(\boldsymbol{\varepsilon}^{\mathcal{M}}\right) \boldsymbol{I} + 2\mu^{\mathcal{M}}(\boldsymbol{\chi}) \boldsymbol{\varepsilon}^{\mathcal{M}}\right) = \boldsymbol{0} \;\text{ in } \Omega_0^{\mathcal{F}} \times [0, T] \,, \tag{9.22}$$

where the strain tensor is defined as

$$\boldsymbol{\varepsilon}^{\mathcal{M}} = \frac{1}{2}\left(\nabla_{\boldsymbol{\chi}} \boldsymbol{d}^{\mathcal{M}} + \left(\nabla_{\boldsymbol{\chi}} \boldsymbol{d}^{\mathcal{M}}\right)^{\mathsf{T}}\right) \,, \tag{9.23}$$

and where $\lambda^{\mathcal{M}}, \mu^{\mathcal{M}}$ are pseudo material parameters related to Young's modulus $E^{\mathcal{M}}$ and Poisson's ratio $\nu^{\mathcal{M}}$ through equation (9.11). The notation with a distinction between $\boldsymbol{x}$ and $\boldsymbol{\chi}$ is used here for clarity, indicating that the displacements might indeed be large for the fluid mesh deformation despite the use of a linear elasticity approach. In case of using a non-linear elasticity model for the mesh motion problem, the formulation follows immediately from Section 9.2.2 by dropping the body force term and time derivative term in equation (9.6) and by replacing the material coordinates $\boldsymbol{X}$ by the mesh coordinates $\boldsymbol{\chi}$

$$-\nabla_{\boldsymbol{\chi}} \cdot \left(\boldsymbol{F}^{\mathcal{M}} \cdot \boldsymbol{S}^{\mathcal{M}}\right) = \boldsymbol{0} \;\text{ in } \Omega_0^{\mathcal{F}} \times [0, T] \,, \tag{9.24}$$

where the second Piola–Kirchhoff stress tensor is described in Section 9.2.2, using for example the St. Venant–Kirchhoff material law.

The fluid mesh movement is described as a PDE-type model problem, and therefore requires suitable boundary conditions. Note that the PDE for the mesh deformation is most often formulated as steady-state problem (as introduced above) with boundary conditions changing over time, while some works also consider dynamic formulations. At the fluid–structure interface, the fluid mesh follows the motion of the structure, which leads to the geometric coupling condition

$$\boldsymbol{d}^{\mathcal{M}} = \boldsymbol{d}^{\mathcal{S}} \text{ on } \Gamma_0^{\mathrm{I}} \,. \tag{9.25}$$

The other boundaries, $\Gamma^{\mathrm{D},\mathcal{F}}(t) \cup \Gamma^{\mathrm{N},\mathcal{F}}(t)$, are typically non-moving and a common choice is the use of homogeneous Dirichlet boundary conditions for the mesh motion, $\boldsymbol{d}^{\mathcal{M}} = \boldsymbol{0}$. As a means to keep fluid mesh deformations small, an alternative is to allow a sliding of the fluid mesh along the boundary and only prescribe the motion normal to the boundary $\boldsymbol{d}^{\mathcal{M}} \cdot \boldsymbol{n} = 0$ in order to adhere to the physical boundaries of the problem.

9.3 Discretization in space and time

This section discusses the discretization of the fluid–structure interaction problem in space and time. The discretization of the fluid problem relies on the high-order discontinuous Galerkin

solver with matrix-free implementation that has been introduced in previous chapters. Hence, the new ingredient in this section is the description of a structural mechanics solver based on a (potentially high-order) continuous finite element method for discretization in space, and the generalized-α method for discretization in time. Similarly, the mesh motion problem is also discretized by a high-order continuous finite element discretization and uses ingredients of the structural mechanics solver due to the pseudo-solid analogy of the mesh motion problem.

9.3.1 Fluid

The fluid solver is based on the methodology introduced in Chapter 2 for non-moving meshes, and extended to moving meshes via arbitrary Lagrangian–Eulerian techniques in Chapter 8. The fluid solver uses BDF time integration and high-order discontinuous Galerkin methods for discretization in space. The black-box design of the FSI solver followed in this chapter allows to take advantage of the flexibility of the fluid solver with respect to different Navier–Stokes solution techniques (coupled or monolithic solution versus projection methods), fully-implicit or mixed implicit/explicit time integration, and adaptive time stepping. As motivated in the introduction of this chapter, this is a key ingredient in order to obtain a versatile and computationally efficient FSI solver.

9.3.2 Structure

This section describes the spatial discretization and temporal discretization for the structural field. For ease of notation, this section omits the superscript $\mathcal{S}$ indicating the structural field.

9.3.2.1 Spatial discretization

The spatial discretization of the solid field is based on a standard displacement-based continuous Galerkin finite element discretization, where high-order polynomial shape functions can be used as a means to mitigate locking. The space of solution functions for tensor-product elements $\mathcal{Q}_k$ of degree k is given as

$$\mathcal{V}_h = \left\{ \boldsymbol{d}_h \in \left[H^1(\Omega_{h,0})\right]^d : \ \boldsymbol{d}_h\left(\boldsymbol{X}^e(\boldsymbol{\xi})\right)|_{\Omega_{e,0}} = \tilde{\boldsymbol{d}}_h^e(\boldsymbol{\xi})|_{\tilde{\Omega}_e} \in \left[\mathcal{Q}_k(\tilde{\Omega}_e)\right]^d \, \forall e, \ \boldsymbol{d}_h|_{\Gamma^{\mathrm{D}}_{h,0}} = \mathcal{I}_h\left(\boldsymbol{g}\right) \right\} \ , \tag{9.26}$$

where $H^1(\Omega_{h,0})$ is the space of functions in $L^2(\Omega_{h,0})$ with first derivative also in $L^2(\Omega_{h,0})$, and where $\mathcal{I}_h$ is an operator interpolating the prescribed Dirichlet boundary data. Due to the strong imposition of Dirichlet boundary conditions, the space of test functions is given as

$$\mathcal{V}_{h,0} = \left\{ \boldsymbol{d}_h \in \mathcal{V}_h : \ \boldsymbol{d}_h|_{\Gamma^{\mathrm{D}}_{h,0}} = \boldsymbol{0} \right\} \ . \tag{9.27}$$

The usual shorthand notation $(\cdot,\cdot)_{\Omega_h}$ is used to denote the L^2-product. Then, the weak formulation can be stated as follows: Find $\boldsymbol{d}_h \in \mathcal{V}_h$ such that it holds for all $\boldsymbol{v}_h \in \mathcal{V}_{h,0}$

$$\left(\boldsymbol{v}_h, \rho_h \frac{\mathrm{d}^2 \boldsymbol{d}_h}{\mathrm{d}t^2} \right)_{\Omega_{h,0}} + \left(\nabla_0 \boldsymbol{v}_h, \boldsymbol{P}_h\right)_{\Omega_{h,0}} - \left(\boldsymbol{v}_h, \boldsymbol{t}_0\right)_{\Gamma^{\mathrm{N}}_{h,0}} = \left(\boldsymbol{v}_h, \boldsymbol{b}_0\right)_{\Omega_{h,0}} \ , \tag{9.28}$$

with the first Piola–Kirchhoff stress tensor $\boldsymbol{P}_h(\boldsymbol{d}_h)$ as defined in Section 9.2.2. The above variational form is obtained by integration-by-parts of the nonlinear elasticity operator, exploiting that the test function vanishes on the Dirichlet part of the boundary, and replacing the traction in the Neumann boundary integral by the prescribed Neumann boundary condition $\boldsymbol{t}_0$. Here, the Neumann boundary $\Gamma_{h,0}^{\mathrm{N}}$ also includes the fluid–structure interface Γ_h^{I}, which will become clear from Section 9.5 discussing the Dirichlet–Neumann partitioning scheme that imposes a Neumann boundary condition for the structure on Γ_h^{I}. Note that due to the continuity of shape functions between elements, the variational form is not stated in an elementwise manner as done in the case of discontinuous Galerkin discretizations in previous chapters.

A solution of this nonlinear system of equations (after discretization in time) by Newton's method requires the linearization of the nonlinear terms. The Neumann boundary integral and body force term also depend nonlinearly on the displacement vector (see equation (9.16) and equation (9.17)), but these terms are neglected in the linearization. Hence, it remains to linearize the term $(\nabla_0 \boldsymbol{v}_h, \boldsymbol{P}_h)_{\Omega_{h,0}}$. Denoting by $\boldsymbol{d}_{h,\mathrm{lin}}$ the point of linearization and by $\Delta\boldsymbol{d}_h$ the solution increment, the linearization is given as

$$\begin{aligned}\mathrm{Lin}\left((\nabla_0 \boldsymbol{v}_h, \boldsymbol{P}_h)_{\Omega_{h,0}}\right) = &+ \left(\nabla_0 \boldsymbol{v}_h, \boldsymbol{P}(\boldsymbol{d}_{h,\mathrm{lin}})\right)_{\Omega_{h,0}} \\ &+ \left(\nabla_0 \boldsymbol{v}_h, \nabla_0\left(\Delta\boldsymbol{d}_h\right) \cdot \boldsymbol{S}(\boldsymbol{d}_{h,\mathrm{lin}})\right)_{\Omega_{h,0}} \\ &+ \left(\nabla_0 \boldsymbol{v}_h, \boldsymbol{F}(\boldsymbol{d}_{h,\mathrm{lin}}) \cdot \left(\boldsymbol{C} : \left(\boldsymbol{F}(\boldsymbol{d}_{h,\mathrm{lin}})^{\mathsf{T}} \cdot \nabla_0\left(\Delta\boldsymbol{d}_h\right)\right)\right)\right)_{\Omega_{h,0}} ,\end{aligned} \tag{9.29}$$

where the minor symmetries of the material tensor $\boldsymbol{C}$ have been exploited. As a result of the discretization in space, the following semi-discrete problem is obtained

$$\mathbf{M}\frac{\mathrm{d}^2\mathbf{d}(t)}{\mathrm{d}t^2} + \mathbf{e}\left(\mathbf{d}(t), t\right) - \mathbf{b}\left(\mathbf{d}(t), t\right) = \mathbf{0} . \tag{9.30}$$

where $\mathbf{M}$ denotes the mass matrix (scaled by density), $\mathbf{e}$ the discrete nonlinear elasticity operator, and $\mathbf{b}$ the discrete body force operator.

9.3.2.2 Temporal discretization

Discretization in time is based on the generalized-α time integration scheme (Chung and Hulbert 1993). The basic idea is to introduce two generalized mid-points $t_{n+1-\alpha_{\mathrm{m}}} = \alpha_{\mathrm{m}} t_n + (1-\alpha_{\mathrm{m}}) t_{n+1}$ and $t_{n+1-\alpha_{\mathrm{f}}} = \alpha_{\mathrm{f}} t_n + (1-\alpha_{\mathrm{f}}) t_{n+1}$ at which to simultaneously satisfy the equations of motion

$$\mathbf{M}\mathbf{a}^{n+1-\alpha_{\mathrm{m}}} + \mathbf{e}\left(\mathbf{d}^{n+1-\alpha_{\mathrm{f}}}, t_{n+1-\alpha_{\mathrm{f}}}\right) - \mathbf{b}\left(\mathbf{d}^{n+1-\alpha_{\mathrm{f}}}, t_{n+1-\alpha_{\mathrm{f}}}\right) = \mathbf{0} , \tag{9.31}$$

where the mid-point displacement and acceleration are

$$\mathbf{d}^{n+1-\alpha_{\mathrm{f}}} = \alpha_{\mathrm{f}}\mathbf{d}^n + (1-\alpha_{\mathrm{f}})\,\mathbf{d}^{n+1} , \tag{9.32}$$

$$\mathbf{a}^{n+1-\alpha_{\mathrm{m}}} = \alpha_{\mathrm{m}}\mathbf{a}^n + (1-\alpha_{\mathrm{m}})\,\mathbf{a}^{n+1} . \tag{9.33}$$

Using the following relations for the Newmark time integration scheme with parameters β, γ

$$\frac{\mathbf{d}^{n+1} - \mathbf{d}^n}{\Delta t_n} = \mathbf{v}^n + \frac{\Delta t_n}{2}\left(2\beta\mathbf{a}^{n+1} + (1-2\beta)\,\mathbf{a}^n\right) , \tag{9.34}$$

$$\frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta t_n} = \gamma\mathbf{a}^{n+1} + (1-\gamma)\,\mathbf{a}^n , \tag{9.35}$$

equation (9.31) can be formulated as an equation for the unknown displacements $\mathbf{d}^{n+1-\alpha_\mathrm{f}}$ only. This is done by inserting equations (9.32) and (9.34) into equation (9.33), and the resulting equation into equation (9.31) to replace $\mathbf{a}^{n+1-\alpha_\mathrm{m}}$

$$\begin{aligned}\frac{1-\alpha_\mathrm{m}}{(1-\alpha_\mathrm{f})\beta\Delta t_n^2}\mathbf{M}\mathbf{d}^{n+1-\alpha_\mathrm{f}} + \mathbf{e}\left(\mathbf{d}^{n+1-\alpha_\mathrm{f}}, t_{n+1-\alpha_\mathrm{f}}\right) - \mathbf{b}\left(\mathbf{d}^{n+1-\alpha_\mathrm{f}}, t_{n+1-\alpha_\mathrm{f}}\right) = \\ \mathbf{M}\left[\frac{1-\alpha_\mathrm{m}}{(1-\alpha_\mathrm{f})\beta\Delta t_n^2}\mathbf{d}^n + \frac{1-\alpha_\mathrm{m}}{\beta\Delta t_n}\mathbf{v}^n + \frac{1-\alpha_\mathrm{m}-2\beta}{2\beta}\mathbf{a}^n\right] .\end{aligned} \tag{9.36}$$

In the first time step $n = 0$, the displacement $\mathbf{d}^0$ and velocity $\mathbf{v}^0$ are given by interpolation of the initial conditions. The initial acceleration $\mathbf{a}^0$ is obtained as the solution of the mass matrix problem

$$\mathbf{M}\mathbf{a}^0 + \mathbf{e}\left(\mathbf{d}^0, t_0\right) - \mathbf{b}\left(\mathbf{d}^0, t_0\right) = \mathbf{0} . \tag{9.37}$$

Equation (9.36) can be solved for the displacement $\mathbf{d}^{n+1-\alpha_\mathrm{f}}$. Then the displacement, velocity, and acceleration at time t_{n+1} are obtained from equations (9.32), (9.34), (9.35) as follows

$$\mathbf{d}^{n+1} = \frac{\mathbf{d}^{n+1-\alpha_\mathrm{f}} - \alpha_\mathrm{f}\mathbf{d}^n}{1-\alpha_\mathrm{f}} , \tag{9.38}$$

$$\mathbf{v}^{n+1} = \frac{\gamma}{\beta\Delta t_n}\left(\mathbf{d}^{n+1} - \mathbf{d}^n\right) - \frac{\gamma-\beta}{\beta}\mathbf{v}^n - \frac{\gamma-2\beta}{2\beta}\Delta t_n\mathbf{a}^n , \tag{9.39}$$

$$\mathbf{a}^{n+1} = \frac{1}{\beta\Delta t_n^2}\left(\mathbf{d}^{n+1} - \mathbf{d}^n\right) - \frac{1}{\beta\Delta t_n}\mathbf{v}^n - \frac{1-2\beta}{2\beta}\mathbf{a}^n . \tag{9.40}$$

9.3.2.3 Implementation and iterative solution of discrete problem

The nonlinear system of equations (9.36) is solved by a Newton–Krylov approach. The nonlinear residual evaluation as well as the linearized operator can immediately be evaluated by matrix-free techniques with sum-factorization on tensor-product elements as discussed in Chapter 4. A matrix-free implementation for nonlinear elasticity problems with Neo-Hookean constitutive law has recently been proposed in Davydov et al. (2020), which uses the same implementation of matrix-free evaluation techniques, provided by the `deal.II` library, as the present work. For problems with complex constitutive law that does not result in a constant material tensor but instead depends on the current state of deformation, the decisive factor in achieving high performance concerns the question whether all quantities should be recomputed in every iteration of the linear solver or whether certain quantities should be stored[2] and then streamed from memory for every application of the matrix-vector product. The reason behind is that for simple constitutive laws the arithmetic work per quadrature point can often be hidden behind the transfer of the solution vectors and geometry data (on deformed meshes) from main memory. However, for constitutive laws that are more expensive to evaluate with significantly increased arithmetic work per quadrature point, precomputing certain information and storing this data for all quadrature

[2]Here, the term "cached" used in Davydov et al. (2020) is avoided to prevent misunderstandings in the sense that the data needs to be "cached" for all quadrature points for all elements and, therefore, can not reside in the cache but needs to be streamed from main memory.

points can be a better compromise with increased performance in the sense of a higher throughput, i.e., a faster evaluation of the matrix-vector product. These aspects are discussed in detail in Davydov et al. (2020). Here, all quantities are computed on-the-fly given the simplicity of the St. Venant–Kirchhoff constitutive behavior considered in this work.

To solve the linearized system of equations, the hybrid geometric multigrid preconditioner with AMG coarse grid solver as discussed in Chapter 5 can be used. To evaluate the operator on coarser multigrid levels, the displacement vector $\boldsymbol{d}_{h,\mathrm{lin}}$ describing the current point of linearization is restricted to coarser levels (using matrix-free techniques) in addition to the restriction of the solution vector. As a smoother on each level, Chebyshev smoothing is applied. It has been shown in Davydov et al. (2020) that such a setup significantly outperforms a matrix-based AMG solver especially for higher polynomial degrees, and achieves a throughput in terms of degrees of freedom solved per second that depends only mildly on the polynomial degree. For this reason, the present work follows the same strategy in order to obtain an FSI solver that exploits fast matrix-free techniques for all fields. Due to the fundamentally different complexity of matrix-based and matrix-free implementations, it can be expected that the use of a matrix-based approach in combination with a large polynomial degree k for one of the fields would otherwise limit the performance of the overall FSI solver. For low polynomial degrees, the use of an algebraic multigrid preconditioner might be an efficient alternative.

Remark 9.2 *An open question in the context of matrix-free solvers for high-order discretizations of elasticity problems as discussed in Davydov et al. (2020) appears to be the robustness of Chebyshev smoothing for large polynomial degrees k. While Chebyshev smoothing has been investigated in detail for simple Poisson-type model problems in Fehn et al. (2020), Kronbichler and Wall (2018), Sundar et al. (2015) for high-order (dis-)continuous Galerkin discretizations, it is unclear whether this robust behavior translates to elasticity problems that are characterized by a linearized operator which might be less diagonally-dominant as compared to the Laplace operator and which introduces a coupling of degrees of freedom associated to displacements in different coordinate directions due to the symmetric gradient occurring in the elasticity operator. This topic appears to be rarely explored in the literature in the context of Chebyshev smoothing for high-order methods. While an in-depth investigation of this aspect is beyond the scope of the present work, this aspect should be considered as part of future work.*

9.3.3 Mesh motion

For the mesh deformation problem, an approximation by a continuous finite element space of tensor degree $k^{\mathcal{M}}$ is used as for the structural field, with a strong imposition of Dirichlet boundary conditions. For brevity, the definition of finite element function spaces is therefore omitted here. In case of a harmonic extension, the weak form reads: Find $\boldsymbol{d}_h^{\mathcal{M}} \in \mathcal{V}_h$ such that it holds for all $\boldsymbol{v}_h \in \mathcal{V}_{h,0}$

$$\left(\nabla_{\boldsymbol{\chi}} \boldsymbol{v}_h, \alpha(\boldsymbol{\chi}) \nabla_{\boldsymbol{\chi}} \boldsymbol{d}^{\mathcal{M}}\right)_{\Omega_{h,0}^{\mathcal{F}}} = 0 \,, \tag{9.41}$$

where only the volume term remains due to the fact that Dirichlet boundaries are prescribed on the whole boundary. Analogously, the linear elasticity mesh deformation problem, equa-

tion (9.22), is given in variational form as follows

$$\left(\nabla_{\boldsymbol{\chi}} \boldsymbol{v}_h, \lambda^{\mathcal{M}}(\boldsymbol{\chi}) \operatorname{tr}\left(\boldsymbol{\varepsilon}_h^{\mathcal{M}}\right) \boldsymbol{I} + 2\mu^{\mathcal{M}}(\boldsymbol{\chi}) \boldsymbol{\varepsilon}_h^{\mathcal{M}}\right)_{\Omega_{h,0}^{\mathcal{F}}} = 0 \,, \tag{9.42}$$

with $\boldsymbol{\varepsilon}_h^{\mathcal{M}}(\boldsymbol{d}_h^{\mathcal{M}})$ as given in equation (9.23). The weak form for the mesh deformation described by a nonlinear elasticity problem, equation (9.24), is omitted here, as it is a simplified version of the problem described in Section 9.3.2.1 for the solid domain. The solution of the algebraic system of equations resulting from the mesh motion problem follows the multigrid techniques used for the fluid and solid problems, see also Chapter 5.

9.3.4 Time step calculation

The partitioned FSI solvers discussed here require the individual fields to be advanced with the same time step size. Hence, the minimum of the time step size from the fluid field and structure field is taken as a global time step size. In case of adaptive time stepping, the implementation allows the fluid field to update the time step size adaptively after each time step (for example when using an explicit treatment of the convective term with a time step restriction according to the CFL condition), while no time step adaptation is implemented for the structure. In case of adaptive time stepping, the new time step size then needs to be synchronized between the fluid domain and solid domain after each time step. This strategy originates from the fact that a fully implicit formulation is used for the structure, while mixed explicit/implicit formulations are supported for the fluid domain for reasons of flexibility and computational efficiency. Hence, this type of time step adaptation assumes that the optimal time step size is driven by stability rather than accuracy considerations, see also Chapter 6. A different philosophy is for example followed in Mayr et al. (2015, 2020), where the time step size is selected by accuracy considerations in the context of a fully implicit monolithic solution approach.

9.4 Interface coupling

For fluid–structure interaction problems, information needs to be exchanged at the fluid–structure interface. For partitioned FSI schemes, this information exchange has an explicit and directional (from-to) character, as opposed to monolithic solvers where the coupling is contained implicitly in the monolithic system of equations. The present work does not want to make any assumptions regarding the conformity of meshes at the interface, or the type of discretization used for the individual fields (continuous versus discontinuous, modal versus nodal basis, low-order versus high-order). Instead, the information exchange is realized by generic routines using a consistent and high-order interpolation of the respective solution fields. Due to the directional character of the information exchange, there is a destination (dst) side (the field for which the boundary condition has to be prescribed), and a source (src) side (the field on the other side of the interface from which the information for evaluating the boundary condition needs to be queried). The destination side defines a set of points $\boldsymbol{x}$ in physical space at which to evaluate the boundary condition (an illustration is given in Figure 9.2). These points can be the nodes of a discretization, with a typical use case being continuous finite element discretizations with a strong imposition of Dirichlet boundary conditions. However, these points can also be quadrature points, for example in case of Neumann boundary conditions for a continuous finite element

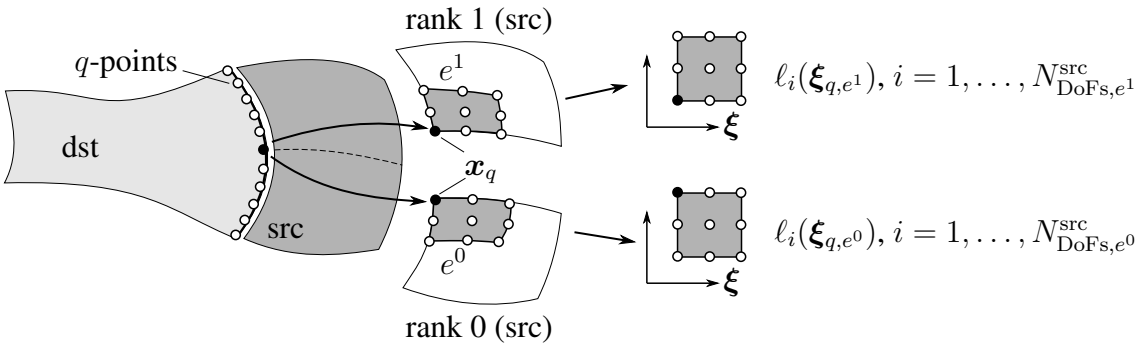


Figure 9.2: Illustration of interface coupling algorithm for the special case that two elements are found on the src-side corresponding to point $\boldsymbol{x}_q$ on the dst-side. In case of domain decomposition used in parallel computations, the two elements might belong to the same rank or to two different ranks on the src-side (the latter case is illustrated in the figure).

discretization, where the Neumann boundary condition appears in the weak form as an integral over the respective part of the boundary which is calculated by numerical quadrature. In case of discontinuous Galerkin discretizations with a weak imposition of boundary conditions, points defined by the destination side are generally quadrature points. The algorithm for interface coupling by consistent interpolation then consists of two steps:

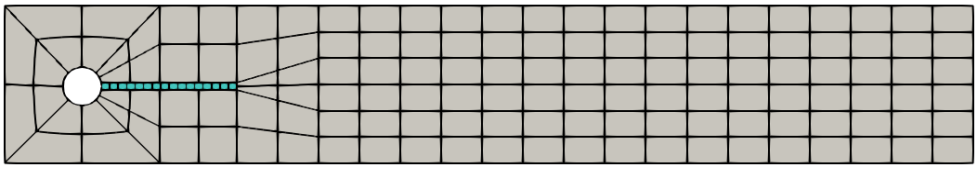
1. Setup phase: For all points $\boldsymbol{x}_q$, $q = 1, \ldots, N_{\text{I}}^{\text{dst}}$, on Γ^{I}, find all elements Ω_e^{src} on the src-side containing point $\boldsymbol{x}_q$ within a given tolerance. Multiple elements might be found, e.g., if the point is located at the interface between elements on the src-side. Then, transform point $\boldsymbol{x}_q$ into reference coordinates $\boldsymbol{\xi}_q$ for each of these elements, evaluate all shape functions $\ell_i(\boldsymbol{\xi}_{q,e})$, $i = 1, \ldots, N^{\text{src}}_{\text{DoFs},e}$, of element e, and determine the dof-indices needed to extract the local degrees of freedom $\boldsymbol{d}_{i,e}$ of element e from the global dof-vector $\mathbf{d}^{\text{src}}$. Fill data structures storing this information (shape function values and dof-indices) in vectors with $N_{\text{I}}^{\text{dst}}$ entries for all the points $\boldsymbol{x}_q$.

2. Interpolation phase: Given the global dof-vector $\mathbf{d}^{\text{src}}$, read the shape function values and dof-indices corresponding to point $\boldsymbol{x}_q$ from the data structures filled in the setup phase, extract the local degrees of freedom $\boldsymbol{d}_{i,e}$ associated to the stored dof-indices, and perform the interpolation for all neighboring elements e associated to point $\boldsymbol{x}_q$

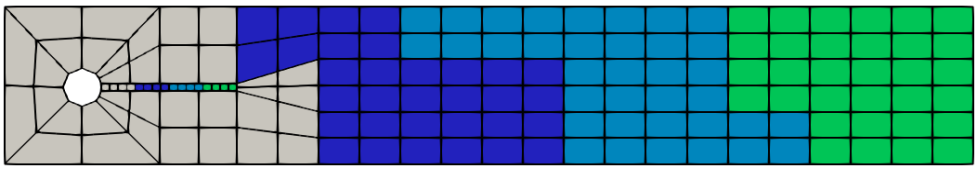
$$\boldsymbol{d}_{q,e} = \sum_{i=1}^{N^{\text{src}}_{\text{DoFs},e}} \ell_i(\boldsymbol{\xi}_{q,e}) \boldsymbol{d}_{i,e} \ . \tag{9.43}$$

   To obtain the final result $\boldsymbol{d}_q$, take the arithmetic mean of $\boldsymbol{d}_{q,e}$ over all neighboring elements e. The interpolation phase is done whenever the boundary condition needs to be evaluated, e.g., in each iteration of the partitioned FSI scheme.

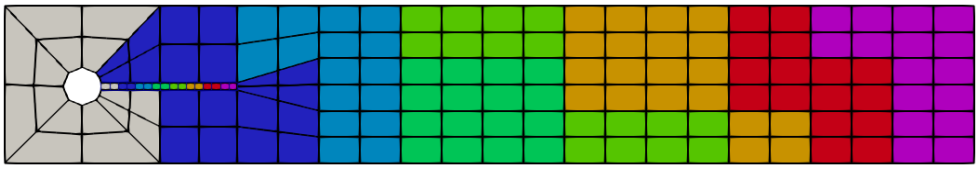
In the context of parallel simulations with a distribution of the mesh and global dof-vectors over the different processors for the individual field solvers, the distribution of points $\boldsymbol{x}_q$ on the



(a) fluid domain (gray) and solid domain (cyan) with quadrilateral meshes



(b) parallel partitioning of fluid and solid meshes for $N_{\text{cores}} = 4$



(c) parallel partitioning of fluid and solid meshes for $N_{\text{cores}} = 8$

Figure 9.3: Illustration of non-conforming meshes at FSI interface and independent partitioning of fluid and solid domains in parallel computations for the cylinder-with-flag FSI benchmark problem (refine level $l = 1$ for fluid domain and $l = 0$ for solid domain). The implementation of the interface coupling is flexible in the sense that it allows continuous or discontinuous finite element spaces with independent polynomial degrees (or quadrature rules) for the fluid and solid domains.

dst-side directly follows the decomposition of the mesh on the dst-side. However, since the partitioning of the domain on the src-side is done independently from the dst-side, the corresponding processor on the src-side will be a different one in general. A communication pattern needs to be constructed in terms of which processors on the src-side hold data relevant for the interpolation in point $\boldsymbol{x}_q$ on the dst-side. In particular, it might happen that different processors on the src-side contribute to the same point $\boldsymbol{x}_q$ on the dst-side, see also Figure 9.2. This part of the algorithm has been implemented by Peter Munch, PhD student at the Institute for Computational Mechanics. Figure 9.3 illustrates the flexibility of the implementation in terms of non-matching discretizations and an independent domain decomposition for fluid and solid domains for a practical example.

Remark 9.3 *A separation of the algorithm into a setup phase (called lookup phase in Lindner et al. (2020)) and an interpolation phase is done for two reasons. On the one hand, the individual field solvers do not have access to the mesh on the other side of the interface when implemented in a black-box fashion, which would be required if setup (geometric search of neighboring elements in the src-mesh) and interpolation were done all at once whenever evaluating the boundary condition within a single-field solver call. On the other hand, the geometric search of elements surrounding a point is computationally the most expensive step, so that it makes sense to perform this step only once in the setup phase instead of repeating this step within every*

iteration of the partitioned FSI scheme in each time step. Note that this is possible since there is no relative motion between the solid and the fluid mesh at the interface. Of course, the data structures constructed in the setup phase need to be updated in case of hp-adaptivity whenever the discretization changes on either side of the interface or when (re)partitioning the computational domain.

Remark 9.4 *To speed up the geometric search of neighboring elements, the search can be restricted to elements located at the fluid–structure interface. Moreover, for high-order elements with many degrees of freedom or quadrature points on each element, it is likely that several points have the same neighboring elements on the src-side, i.e., the result of the search for a previous point can be used as initial guess for the next point. Finally, a space-tree can be used to obtain algorithms of optimal computational complexity in terms of the number of elements that need to be tested to find a neighboring element (logarithmic versus linear complexity). Especially the latter aspect speeds up the search considerably, see also Lindner et al. (2020). These types of algorithms are provided by the* `deal.II` *library and are not discussed in detail here. Since the solution is in general discontinuous between elements, all neighboring elements have to be identified (instead of just one element) as a means to make the result of the overall solver independent of the specific algorithm used for the interface coupling.*

Remark 9.5 *For matching grids at the fluid–structure interface and homogeneous polynomial degree k on both sides of the interface, the interpolation part of the above algorithm has complexity $\mathcal{O}(k^{2d-1})$ for tensor-product elements in d space dimensions, which is a complexity in k that is higher than the memory requirements of $\mathcal{O}(k^d)$ and operation counts of $\mathcal{O}(k^{d+1})$ for the matrix-free evaluation of weak forms for the individual field solvers. Since these operations have to be performed only on a $d-1$ dimensional surface in d space dimensions, the computational costs can be expected to be negligibly small for serial computations, and also for parallel computations provided that a reasonable load balancing can be achieved.*

Remark 9.6 *The explicit coupling algorithm described above is applied to the surface-coupling of different fields at the fluid–structure interface. However, due to the formulation in terms of an arbitrary set of points $\{\boldsymbol{x}_q\}_{q=1}^{N^{\mathrm{dst}}}$ on the dst-side, the algorithm could be applied to volume-coupling as well. It should be noted that such an algorithm would have increased complexity of $\mathcal{O}(k^{2d})$ as compared to $\mathcal{O}(k^{d+1})$ for the sum-factorization algorithms and the matrix-free evaluation of weak forms, see Chapter 4. This is due to the fact that the generic coupling algorithm does not have information about a tensor-product structure of all the points $\boldsymbol{x}_q$ within an element. It can be expected that these operations are still inexpensive if only done once per time step (or per outer iteration of a partitioned solver) and for moderately large polynomial degrees, but might form a bottleneck in terms of memory requirements or compute time especially for very large polynomial degrees due to the complexity of $\mathcal{O}(k^{2d})$ for a volume-coupling in d space dimensions.*

9.5 Dirichlet–Neumann partitioning scheme

Within each time step of the transient problem, partitioned FSI schemes solve the sub-fields sequentially with appropriate boundary conditions at the fluid–structure interface. One sweep

Algorithm 9.1 Dirichlet–Neumann partitioning $\tilde{\mathbf{d}}^{\mathcal{S}} = \mathbf{f}_{\mathrm{FSI}}(\mathbf{d}^{\mathcal{S}})$

function $\mathbf{f}_{\mathrm{FSI}}(\mathbf{d}^{\mathcal{S}})$
  $\mathcal{M}$: Coupling structure $\rightarrow$ mesh: use $\mathbf{d}^{\mathcal{S}}$ to interpolate mesh DBC at Γ^{I}
  $\mathcal{M}$: Solve mesh motion problem on fluid domain for displacement $\mathbf{d}^{\mathcal{M}}$
  $\mathcal{M}$: Compute fluid mesh velocity $\mathbf{u}^{\mathcal{M}}$ and update metric terms on fluid domain
  $\mathcal{F}$: Coupling structure $\rightarrow$ fluid: use $\mathbf{v}^{\mathcal{S}}$ to interpolate velocity DBC at Γ^{I}
  $\mathcal{F}$: Solve fluid problem for velocity $\mathbf{u}^{\mathcal{F}}$ and kinematic pressure $\mathbf{p}^{\mathcal{F}}$
  $\mathcal{F}$: Compute fluid traction vector $\mathbf{t}^{\mathcal{F}}$ at Γ^{I}
  $\mathcal{S}$: Coupling fluid $\rightarrow$ structure: use $\mathbf{t}^{\mathcal{F}}$ to interpolate structure NBC at Γ^{I}
  $\mathcal{S}$: Solve structural problem for new displacements $\tilde{\mathbf{d}}^{\mathcal{S}}$
  $\mathcal{S}$: Update velocity $\mathbf{v}^{\mathcal{S}}$ and acceleration $\mathbf{a}^{\mathcal{S}}$ using new displacements $\tilde{\mathbf{d}}^{\mathcal{S}}$
  return $\tilde{\mathbf{d}}^{\mathcal{S}}$
end function

through all field solvers is called FSI-cycle, and the FSI solver is called strongly-coupled if the FSI-cycle is applied iteratively until convergence is achieved in an appropriate norm. The converged solution then forms the solution of the current time step. Strongly-coupled schemes are in general required for FSI problems involving incompressible flows due to the added-mass effect. The present work considers the widely used Dirichlet–Neumann FSI partitioning scheme with Dirichlet boundary conditions for the fluid and Neumann boundary conditions for the structure on Γ^{I}. Following the methodology in Gerbeau et al. (2005), Küttler and Wall (2008), the Dirichlet–Neumann scheme can be formulated in abstract notation as follows

$$\tilde{\mathbf{d}}^{\mathcal{S}} \leftarrow \mathcal{S} \circ \mathcal{F} \circ \mathcal{M}(\mathbf{d}^{\mathcal{S}}) = \mathbf{f}_{\mathrm{FSI}}(\mathbf{d}^{\mathcal{S}}) \ , \ \mathbf{d}^{\mathcal{S}} \in \mathbb{R}^m \ , \tag{9.44}$$

where $\mathbf{d}^{\mathcal{M}} = \mathcal{M}(\mathbf{d}^{\mathcal{S}})$ denotes the mesh motion problem resulting in the fluid mesh deformation, from which the mesh velocity can be derived. Subsequently, $\mathbf{t}^{\mathcal{F}} = \mathcal{F}(\mathbf{d}^{\mathcal{M}})$ denotes the fluid sub-problem mapping displacements and velocity Dirichlet boundary conditions at the fluid–structure interface to traction forces on the fluid–structure interface. Finally, $\mathbf{d}^{\mathcal{S}} = \mathcal{S}(\mathbf{t}^{\mathcal{F}})$ denotes the structure sub-problem mapping interface forces to displacements of the structural problem. The Dirichlet–Neumann partitioning scheme is described in detail in Algorithm 9.1. The solution of each field consists of three sub-steps, (i) a coupling step interpolating boundary conditions from data provided by other fields (see Section 9.4), (ii) the actual solution step solving for the unknown degrees of freedom of this field, and (iii) a postprocessing step preparing data required by other fields. Since the overall problem is formulated in terms of the structural displacements, the notation is simplified by using $\mathbf{d}^{\mathcal{S}} = \mathbf{d}$ in the following. The notation $\tilde{\mathbf{d}}$ indicates that the obtained displacement field is a preliminary solution that can be used for the next FSI-cycle until convergence is achieved for the fixed-point equation

$$\mathbf{d} = \mathbf{f}_{\mathrm{FSI}}(\mathbf{d}) \ . \tag{9.45}$$

The alternative residual formulation reads

$$\mathbf{r}(\mathbf{d}) = \tilde{\mathbf{d}} - \mathbf{d} = \mathbf{f}_{\mathrm{FSI}}(\mathbf{d}) - \mathbf{d} = \mathbf{0} \ . \tag{9.46}$$

It can be expected that a simple fixed-point iteration using the Dirichlet–Neumann partitioning scheme does not converge without further measures, especially if the density ratio $\rho^{\mathcal{F}}/\rho^{\mathcal{S}}$ is large. Techniques for convergence acceleration are discussed in Section 9.6.

Remark 9.7 *Many works restrict the fixed-point equation or residual equation to the degrees of freedom on the fluid–structure interface, $\mathbf{d}_{\Gamma^{\mathrm{I}}} = \mathbf{d}|_{\Gamma^{\mathrm{I}}}$, see for example Küttler and Wall (2008). In the present work, all degrees of freedom of the structural problem contribute to the residual, see for example Fernández and Moubachir (2005), Sheldon et al. (2014). This design choice is made for ease of implementation, only requiring generic finite element functionality already including the parallel partitioning of the dof-vector. The implementation could be extended by problem-specific routines extracting certain degrees of freedom from a dof-vector and taking care of the parallel partitioning of the interface dof-vector $\mathbf{d}_{\Gamma^{\mathrm{I}}}$. Note, however, that the notion of interface degrees of freedom implies a nodal basis, so that the formulation is restricted to a certain type of discretization approach that is no longer applicable to a modal expansion of shape functions. Hence, the present design can also be considered more generic, again following the goal of black-box FSI solvers that do not introduce knowledge about the individual field solvers. Of course, operating on the full dof-vector $\mathbf{d}$ increases storage requirements and potentially also computational costs. However, the partitioned FSI schemes discussed below are formulated in terms of dof-vectors only and do not require the storage of a matrix related to the dof-vector $\mathbf{d}$ (such as the Jacobian matrix for the residual equation), so that the overhead related to the full dof-vector design is small. In case of a $d-1$ dimensional model for the structure (beams and shells), all structural degrees of freedom are located at the interface and there is no difference between both approaches.*

Remark 9.8 *In the present work, the parallel partitioning is done independently for the different fields, with all parallel resources contributing to the solution of each single-field. This is reasonable, given that the partitioned FSI schemes considered here solve the single-field problems sequentially. Altering the iteration order of the partitioned FSI scheme for the purpose of better parallel efficiency – with one group of processors working on the structural field and the remaining processors simultaneously working on the fluid field – can be expected to deteriorate the convergence of the FSI iteration, rendering an overall performance advantage unclear (Mehl et al. 2016).*

In the first iteration $k = 0$ of the fixed-point scheme, the displacement $\left(\mathbf{d}^{\mathcal{S}}\right)_0^{n+1}$ and velocity $\left(\mathbf{v}^{\mathcal{S}}\right)_0^{n+1}$ at the end of the current time step t_{n+1} are not known, but are required for the solution of the mesh and fluid problems. Hence, the following extrapolation schemes are used to predict these quantities in the first iteration

$$\left(\mathbf{d}^{\mathcal{S}}\right)_0^{n+1} = \left(\mathbf{d}^{\mathcal{S}}\right)^n + \Delta t_n \left(\mathbf{v}^{\mathcal{S}}\right)^n \ , \tag{9.47}$$

$$\left(\mathbf{v}^{\mathcal{S}}\right)_0^{n+1} = \left(\mathbf{v}^{\mathcal{S}}\right)^n \ . \tag{9.48}$$

Note that the type of extrapolation is tied to the type of time integration used for the structural problem (generalized-α in the present work), using only those vectors that are provided by the structural solver as a means to sustain the black-box design. For example, extrapolations involving the solution at previous instants of time would be used in case of BDF time integration for

the structure. The type of extrapolation does not affect the final solution of the time step, but a more accurate extrapolation might allow coarser solver tolerances to achieve the same solution quality at reduced computational costs. The mesh velocity in the fluid domain is calculated as

$$\left(\mathbf{u}^{\mathcal{M}}\right)^{n+1} = \frac{\gamma_0^n \left(\mathbf{d}^{\mathcal{M}}\right)^{n+1} - \sum_{i=0}^{J-1} \alpha_i^n \left(\mathbf{d}^{\mathcal{M}}\right)^{n-i}}{\Delta t_n} \,, \tag{9.49}$$

using a BDF-type time derivative in accordance with the time integration scheme for the fluid domain, see Chapters 2 and 8 for more detailed information. The evaluation of the Cauchy stresses in the fluid domain at the fluid–structure interface Γ^{I} is calculated as follows

$$\boldsymbol{t}^{\mathcal{F}} = \boldsymbol{\sigma}^{\mathcal{F}} \cdot \boldsymbol{n}^{\mathcal{F}} = \rho^{\mathcal{F}} \left(-p^{\mathcal{F}} \boldsymbol{I} + \nu^{\mathcal{F}} \left(\nabla \boldsymbol{u}^{\mathcal{F}} + \left(\nabla \boldsymbol{u}^{\mathcal{F}} \right)^{\mathsf{T}} \right) \right) \cdot \boldsymbol{n}^{\mathcal{F}} \,, \tag{9.50}$$

so that the Neumann boundary condition for the structural problem is $\boldsymbol{t}^{\mathcal{S}} = -\boldsymbol{t}^{\mathcal{F}}$. It is again emphasized that $p^{\mathcal{F}}$ is the kinematic fluid pressure. While the Cauchy stresses are typically formulated using the dynamic pressure and the dynamic viscosity, the above formulation with multiplication by the fluid density is chosen in order to highlight that the incompressible flow solver is implemented in kinematic formulation.

9.6 Acceleration schemes

This section discusses techniques for accelerating the convergence of the partitioned FSI iteration. The methods considered in this work are state-of-the-art methods from the literature that can be realized in a black-box fashion. Section 9.6.1 presents the fixed-point iteration scheme with dynamic Aitken relaxation (Küttler and Wall 2008, Mok and Wall 2001), Section 9.6.2 the IQN-ILS method (Degroote et al. 2009), and Section 9.6.3 the IQN-IMVLS method (Spenke et al. 2020).

9.6.1 Fixed-point iteration with dynamic relaxation (Aitken relaxation)

The first partitioned FSI solver considered here is the fixed-point iteration scheme with dynamic Aitken relaxation developed by Mok and Wall (2001) and published later in Küttler and Wall (2008)

$$\tilde{\mathbf{d}}_k = \mathbf{f}_{\mathrm{FSI}}(\mathbf{d}_k) \,, \tag{9.51}$$

$$\mathbf{r}_k = \tilde{\mathbf{d}}_k - \mathbf{d}_k \,, \tag{9.52}$$

$$\omega_k = -\omega_{k-1} \frac{\mathbf{r}_{k-1}^{\mathsf{T}} \left(\mathbf{r}_k - \mathbf{r}_{k-1} \right)}{\left\| \mathbf{r}_k - \mathbf{r}_{k-1} \right\|^2} \,, \tag{9.53}$$

$$\mathbf{d}_{k+1} = \mathbf{d}_k + \omega_k \mathbf{r}_k \,, \tag{9.54}$$

Due to the recursion in the update formula for the Aitken relaxation parameter, this equation can only be used for $k \geq 1$, making the initial relaxation parameter ω_0 in iteration $k = 0$ a parameter that has to be specified by the user ($0 < \omega_0 < 1$). The same value is used at the beginning of each time step in this work, slightly differing from the version used in Küttler and Wall (2008). The Aitken relaxation scheme is summarized in Algorithm 9.2.

Algorithm 9.2 Fixed-point scheme with Aitken relaxation

function AITKEN($\mathbf{d}_0, \omega_0$)
  $k = 0$
  while not converged **do**
    $\tilde{\mathbf{d}}_k = \mathbf{f}_{\mathrm{FSI}}(\mathbf{d}_k)$
    $\mathbf{r}_k = \tilde{\mathbf{d}}_k - \mathbf{d}_k$
    if not converged **then**
      if $k > 0$ **then**
        $\omega_k = -\omega_{k-1} \frac{\mathbf{r}_{k-1}^{\mathsf{T}}(\mathbf{r}_k - \mathbf{r}_{k-1})}{\|\mathbf{r}_k - \mathbf{r}_{k-1}\|^2}$
      end if
      $\mathbf{d}_{k+1} = \mathbf{d}_k + \omega_k \mathbf{r}_k$
    end if
    $k \leftarrow k + 1$
  end while
end function

Remark 9.9 *Note that updating the displacement vector in the relaxation step overwrites the solution of the structural time integrator, which invokes an update of the velocity and acceleration vectors according to equations* (9.39) *and* (9.40) *using the relaxed displacement vector. For ease of notation, this is not written explicitly in the above equations, and without further mention this also holds for other partitioned FSI schemes presented below.*

9.6.2 IQN-ILS method

The IQN-ILS (interface quasi-Newton with inverse Jacobian by least squares approximation) approach has been proposed by Degroote et al. (2009). The derivation shown below follows the lines of presentation in Lindner et al. (2015). The residual equation is reformulated with the intermediate displacement $\tilde{\mathbf{d}}$ as unknown

$$\mathbf{r}(\tilde{\mathbf{d}}) = \tilde{\mathbf{d}} - \mathbf{d} = \tilde{\mathbf{d}} - \mathbf{f}_{\mathrm{FSI}}^{-1}(\tilde{\mathbf{d}}) = \mathbf{0} \ . \tag{9.55}$$

To obtain a Newton update step, consider the linearization

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \left.\frac{\partial \mathbf{r}}{\partial \tilde{\mathbf{d}}}\right|_{\tilde{\mathbf{d}}_k} \Delta\tilde{\mathbf{d}}_k = \mathbf{r}_k + \mathbf{J}_k \Delta\tilde{\mathbf{d}}_k \overset{!}{=} \mathbf{0} \ . \tag{9.56}$$

For black-box solvers, no information is available for the Jacobian. The idea of the IQN-ILS approach is to approximate the inverse Jacobian from the residuals and solutions obtained from previous iterations. The above linearization is rewritten as a secant equation, i.e., the Jacobian is approximated by the slope of a secant

$$\Delta\tilde{\mathbf{d}} = \mathbf{J}_k^{-1} \Delta\mathbf{r} \ . \tag{9.57}$$

Arranging previous residuals and intermediate solutions into matrices as in Spenke et al. (2020)

$$\mathbf{D}_k = [\Delta\tilde{\mathbf{d}}_0^1, \ldots, \Delta\tilde{\mathbf{d}}_{k-1}^k] \in \mathbb{R}^{m \times k} \ , \ \Delta\tilde{\mathbf{d}}_i^j = \tilde{\mathbf{d}}_j - \tilde{\mathbf{d}}_i \ , \tag{9.58}$$

$$\mathbf{R}_k = [\Delta\mathbf{r}_0^1, \ldots, \Delta\mathbf{r}_{k-1}^k] \in \mathbb{R}^{m \times k} \ , \ \Delta\mathbf{r}_i^j = \mathbf{r}_j - \mathbf{r}_i \ . \tag{9.59}$$

Algorithm 9.3 QR-decomposition $\mathbf{V} = \mathbf{QR}$

function QR($\mathbf{V}, \varepsilon$)
  initialize $\mathbf{Q} = [\mathbf{q}_0, \dots, \mathbf{q}_{k-1}] \leftarrow \mathbf{V}, \mathbf{R} = [r_{ij}]$ with $r_{ij} = 0$
  for $i = 0, \dots, k-1$ **do**
    compute $r_{ii}^0 = \|\mathbf{q}_i\|$
    for $j = 0, \dots, i-1$ **do**
      $r_{ji} = \mathbf{q}_j^\mathsf{T} \mathbf{q}_i$
      $\mathbf{q}_i = \mathbf{q}_i - r_{ji}\mathbf{q}_j$
    end for
    $r_{ii} = \|\mathbf{q}_i\|$
    if $r_{ii} < \varepsilon\, r_{ii}^0$ **then**
      Set $\mathbf{q}_i = \mathbf{0}, r_{ji} = 0\ \forall j < i, r_{ii} = 1$
    else
      $\mathbf{q}_i = \mathbf{q}_i / r_{ii}$
    end if
  end for
  return $\mathbf{Q}, \mathbf{R}$
end function

and requiring the secant equation to be fulfilled for all pairs of vectors, yields the following set of equations for the approximate inverse Jacobian

$$\mathbf{D}_k = \mathbf{J}_k^{-1} \mathbf{R}_k \ . \tag{9.60}$$

Since typically $k \ll m$, this system of equations is under-determined. The IQN-ILS approach approximates the inverse Jacobian by the least squares minimization

$$\mathbf{J}_k^{-1} = \mathbf{D}_k \left(\mathbf{R}_k^\mathsf{T} \mathbf{R}_k \right)^{-1} \mathbf{R}_k^\mathsf{T} \ . \tag{9.61}$$

For reasons of computational efficiency, assembly and storage of the inverse Jacobian should be avoided due to the quadratic complexity m^2. As shown below, this is easy to realize since the only use of the inverse Jacobian is its application to a vector. Further, for reasons of stability and robustness of the algorithm, a QR-decomposition of $\mathbf{R}_k = \mathbf{Q}_k \mathbf{U}_k$ is computed by transforming $\mathbf{R}_k$ into the product of an orthonormal basis $\mathbf{Q}_k \in \mathbb{R}^{m \times k}$ with an upper triangular matrix $\mathbf{U}_k \in \mathbb{R}^{k \times k}$ (small $k \times k$ matrix) via (modified) Gram–Schmidt orthogonalization. For the QR-decomposition the improved algorithm proposed in Haelterman et al. (2016) is used, and is summarized in Algorithm 9.3. The algorithm includes a mechanism to detect linear dependency of the vectors, where a tolerance of $\varepsilon = 10^{-2}$ is used unless specified otherwise. If a vector is close to linear dependency, it is set to zero with $r_{ii} = 1$ on the diagonal to ensure invertibility of $\mathbf{U}_k$, but one could alternatively remove this column and restart the procedure as in Haelterman et al. (2016). With these ingredients, the following update procedure is obtained for the

Algorithm 9.4 IQN-ILS (interface quasi-Newton inverse least-squares) method

function IQN-ILS($\mathbf{d}_0, \omega_0$)
  $k = 0$
  while not converged **do**
    $\tilde{\mathbf{d}}_k = \mathbf{f}_{\mathrm{FSI}}(\mathbf{d}_k)$
    $\mathbf{r}_k = \tilde{\mathbf{d}}_k - \mathbf{d}_k$
    if not converged **then**
      if $k = 0$ **then**
        $\mathbf{d}_1 = \mathbf{d}_0 + \omega_0 \mathbf{r}_0$
      else
        $\mathbf{D}_k = [\tilde{\mathbf{d}}_{k-1} - \tilde{\mathbf{d}}_k, \ldots, \tilde{\mathbf{d}}_0 - \tilde{\mathbf{d}}_k]$
        $\mathbf{R}_k = [\mathbf{r}_{k-1} - \mathbf{r}_k, \ldots, \mathbf{r}_0 - \mathbf{r}_k]$
        $\mathbf{Q}_k, \mathbf{U}_k \leftarrow \mathrm{QR}(\mathbf{R}_k)$
        Solve $\mathbf{U}_k \boldsymbol{\alpha}_k = \mathbf{Q}_k^{\mathsf{T}}(-\mathbf{r}_k)$ via backward substitution
        $\Delta\tilde{\mathbf{d}}_k = \mathbf{D}_k \boldsymbol{\alpha}_k$
        $\mathbf{d}_{k+1} = \tilde{\mathbf{d}}_k + \Delta\tilde{\mathbf{d}}_k$
      end if
    end if
    $k \leftarrow k + 1$
  end while
end function

displacements

$$\begin{aligned} \mathbf{d}_{k+1} &= \tilde{\mathbf{d}}_k + \Delta\tilde{\mathbf{d}}_k = \tilde{\mathbf{d}}_k + \mathbf{J}_k^{-1}\left(\mathbf{0} - \mathbf{r}_k\right) \\ &= \tilde{\mathbf{d}}_k + \mathbf{D}_k \left(\mathbf{R}_k^{\mathsf{T}}\mathbf{R}_k\right)^{-1} \mathbf{R}_k^{\mathsf{T}}(-\mathbf{r}_k) \\ &= \tilde{\mathbf{d}}_k + \mathbf{D}_k \mathbf{U}_k^{-1}\mathbf{Q}_k^{\mathsf{T}}(-\mathbf{r}_k) \\ &= \tilde{\mathbf{d}}_k + \mathbf{D}_k \boldsymbol{\alpha}_k \ . \end{aligned} \tag{9.62}$$

The inverse $\mathbf{U}_k^{-1}$ in the second line of the above equation can be applied by backward substitution. The last line with coefficients $\boldsymbol{\alpha}_k$ highlights that the IQN-ILS method uses a linear combination of previous solutions as relaxation. This method may therefore also be interpreted as a vector extrapolation method according to the vector extrapolation methodology for partitioned FSI problems introduced in Küttler and Wall (2009). The IQN-ILS scheme is summarized in Algorithm 9.4. Since at least two previous vectors are necessary, a relaxation $\mathbf{d}_1 = \mathbf{d}_0 + \omega_0 \mathbf{r}_0$ with parameter ω_0 is used in the first iteration of each time step.

The IQN-ILS method allows to include information from q previous time steps, which can improve the convergence. In that case, the $\mathbf{D}_k, \mathbf{R}_k$ matrices are appended as follows

$$\mathbf{D}_k^{(q)} = \left[\mathbf{D}_k^n, \mathbf{D}_{k_{n-1}}^{n-1}, \ldots, \mathbf{D}_{k_{n-q}}^{n-q}\right] \ , \tag{9.63}$$

$$\mathbf{R}_k^{(q)} = \left[\mathbf{R}_k^n, \mathbf{R}_{k_{n-1}}^{n-1}, \ldots, \mathbf{R}_{k_{n-q}}^{n-q}\right] \ , \tag{9.64}$$

which are then used for the QR-decomposition and the displacement update. In case of reuse with $q > 0$, the relaxation with ω_0 in the first iteration $k = 0$ is only required in the first time

Algorithm 9.5 Algorithm used to aproximate $(\mathbf{J}^n)^{-1}\,\mathbf{r}$ in IQN-IMVLS method.

function INVJXR($\mathbf{D}^{n-q}, \mathbf{R}^{n-q}, \mathbf{Z}^{n-q}, \ldots, \mathbf{D}^{n-1}, \mathbf{R}^{n-1}, \mathbf{Z}^{n-1}, \mathbf{r}$)
  initialize $\mathbf{a} = \mathbf{r}, \mathbf{b} = \mathbf{0}$
  for $i = n-1, \ldots, n-q$ **do**
    $\mathbf{b} \leftarrow \mathbf{b} + \mathbf{D}^i\mathbf{Z}^i\mathbf{a}$
    $\mathbf{a} \leftarrow \mathbf{a} - \mathbf{R}^i\mathbf{Z}^i\mathbf{a}$
  end for
  return b
end function

step $n = 0$. However, it was found that the optimal number of reused time steps minimizing the number of partitioned iterations is problem dependent, see for example Degroote et al. (2009) and Haelterman et al. (2016). In particular, the convergence speed does not monotonously improve for increasing q. A method that aims at overcoming this disadvantage is presented in the next section. In summary, the IQN-ILS method as used here has three user defined parameters, namely ε, ω_0, q.

9.6.3 IQN-IMVLS method

An alternative quasi-Newton method is obtained by replacing the inverse Jacobian in (9.60) by an update formula for the inverse Jacobian from time step n to $n+1$

$$\mathbf{D}_k = \left(\mathbf{J}_k^{n+1}\right)^{-1}\mathbf{R}_k = \left((\mathbf{J}^n)^{-1} + (\Delta\mathbf{J})^{-1}\right)\mathbf{R}_k\,. \tag{9.65}$$

Solving equation (9.65) for the Jacobian increment via a least squares minimization as in equation (9.61) yields the IQN-IMVJ (interface quasi-Newton inverse multiple vector update Jacobian) method introduced by Bogaers et al. (2014)

$$\left(\mathbf{J}_k^{n+1}\right)^{-1} = (\mathbf{J}^n)^{-1} + (\Delta\mathbf{J})^{-1} = (\mathbf{J}^n)^{-1} + \left(\mathbf{D}_k - (\mathbf{J}^n)^{-1}\,\mathbf{R}_k\right)\underbrace{\left(\mathbf{R}_k^\mathsf{T}\mathbf{R}_k\right)^{-1}\mathbf{R}_k^\mathsf{T}}_{=\mathbf{Z}_k\in\mathbb{R}^{k\times m}}\,. \tag{9.66}$$

This method implicitly includes information of all previous time steps, successively updated by newer information as a means to overcome the parameter dependency of the IQN-ILS method regarding the number of reused time steps q. However, this quasi-Newton method updates and stores the inverse Jacobian explicitly. The main disadvantage of the IQN-IMVJ method is, therefore, its quadratic complexity $\mathcal{O}(m^2)$. Even if the method operates only on the interface degrees of freedom on the $d-1$ dimensional fluid–structure interface, the quasi-Newton acceleration scheme of the partitioned FSI problem can be expected to become prohibitively expensive in $d = 3$ space dimensions for increasing problem sizes.

For this reason, the work by Spenke et al. (2020) proposed a method to circumvent this disadvantage by systematically removing all operations of quadratic complexity and formulating the algorithm only in terms of matrix-vector products without explicit storage of an (inverse) Jacobian. The presentation shown here follows Spenke et al. (2020), and the reader is referred to

the original work for more detailed information.[3] Using equation (9.66), the update formula for the displacements can be written as

$$\mathbf{d}_{k+1} = \tilde{\mathbf{d}}_k + \left(\mathbf{J}_k^{n+1}\right)^{-1}(-\mathbf{r}_k) = \tilde{\mathbf{d}}_k - \underbrace{\left(\mathbf{J}^n\right)^{-1}\mathbf{r}_k}_{=\mathbf{b}_k} + \underbrace{\left(\mathbf{D}_k - \left(\mathbf{J}^n\right)^{-1}\mathbf{R}_k\right)}_{=\mathbf{B}_k}\underbrace{\left(\mathbf{R}_k^\mathsf{T}\mathbf{R}_k\right)^{-1}\mathbf{R}_k^\mathsf{T}(-\mathbf{r}_k)}_{=\boldsymbol{\alpha}_k} . \tag{9.67}$$

Recalling that the matrices $\mathbf{D}_k = [\mathbf{D}_{k-1}, \Delta\tilde{\mathbf{d}}_{k-1}^k], \mathbf{R}_k = [\mathbf{R}_{k-1}, \Delta\mathbf{r}_{k-1}^k]$ are appended by one column per iteration, carrying out this induction step yields for the matrix $\mathbf{B}_k$

$$\mathbf{B}_k = \left[\mathbf{B}_{k-1}, \Delta\tilde{\mathbf{d}}_{k-1}^k + \mathbf{b}_{k-1} - \mathbf{b}_k\right] . \tag{9.68}$$

Hence, storage of a Jacobian matrix can be avoided as for the IQN-ILS method described in Section 9.6.2 if it is possible to perform the matrix-vector product $\mathbf{b}_k = \left(\mathbf{J}^n\right)^{-1}\mathbf{r}_k$ using vector-updates only. As shown in Spenke et al. (2020), it holds

$$\left(\mathbf{J}^n\right)^{-1}\mathbf{r}_k = \sum_{i=0}^{n-1}\mathbf{D}_k^i\mathbf{Z}_k^i\prod_{j=i+1}^{n-1}\left(\mathbf{I}_k - \mathbf{R}_k^j\mathbf{Z}_k^j\right)\mathbf{r}_k . \tag{9.69}$$

Since this update would involve information from all previous time steps, it is suggested in Spenke et al. (2020) to limit the update to information from q previous time steps. As a result, the IQN-IMVLS method uses the approximation

$$\mathbf{b}_k = \left(\mathbf{J}^n\right)^{-1}\mathbf{r}_k \approx \sum_{i=n-q}^{n-1}\mathbf{D}_k^i\mathbf{Z}_k^i\prod_{j=i+1}^{n-1}\left(\mathbf{I}_k - \mathbf{R}_k^j\mathbf{Z}_k^j\right)\mathbf{r}_k , \tag{9.70}$$

where the corresponding algorithm is shown in Algorithm 9.5.

[3] Note that the original publication contained two bugs (private communication with the authors) in the summary of the method in (Spenke et al. 2020, Algorithm 1 and Algorithm 2), which is why – apart from changes of nomenclature – the presentation shown here differs from the original publication of the IQN-IMVLS method.

Algorithm 9.6 IQN-IMVLS (interface quasi-Newton implicit multi-vector least-squares) method

1: **function** IQN-IMVLS($\mathbf{d}_0, \omega_0, n, q$)
2: $k = 0$
3: Initialize $\mathbf{D}^n = [], \mathbf{R}^n = [], \mathbf{B} = []$, declare $\mathbf{Q}, \mathbf{U}, \mathbf{Z}^n$
4: **while** not converged **do**
5: $\tilde{\mathbf{d}}_k = \mathbf{f}_{\mathrm{FSI}}(\mathbf{d}_k)$
6: $\mathbf{r}_k = \tilde{\mathbf{d}}_k - \mathbf{d}_k$
7: **if** not converged **then**
8: $\mathbf{b}_k$ = INVJXR($\mathbf{D}^{n-q}, \mathbf{R}^{n-q}, \mathbf{Z}^{n-q}, \ldots, \mathbf{D}^{n-1}, \mathbf{R}^{n-1}, \mathbf{Z}^{n-1}, \mathbf{r}_k$)
9: **if** $k = 0$ **and** ($q = 0$ **or** $n = 0$) **then**
10: $\mathbf{d}_1 = \mathbf{d}_0 + \omega_0 \mathbf{r}_0$
11: **else**
12: $\mathbf{d}_{k+1} = \tilde{\mathbf{d}}_k - \mathbf{b}_k$
13: **if** $k \geq 1$ **then**
14: $\mathbf{D}_k^n = [\mathbf{D}_{k-1}^n, \Delta\tilde{\mathbf{d}}_{k-1}^k]$ with $\Delta\tilde{\mathbf{d}}_i^j = \tilde{\mathbf{d}}_j - \tilde{\mathbf{d}}_i$
15: $\mathbf{R}_k^n = [\mathbf{R}_{k-1}^n, \Delta\mathbf{r}_{k-1}^k]$ with $\Delta\mathbf{r}_i^j = \mathbf{r}_j - \mathbf{r}_i$
16: $\mathbf{B}_k = [\mathbf{B}_{k-1}, \Delta\tilde{\mathbf{d}}_{k-1}^k + \mathbf{b}_{k-1} - \mathbf{b}_k]$
17: $\mathbf{Q}_k, \mathbf{U}_k \leftarrow \mathrm{QR}(\mathbf{R}_k^n)$
18: Solve $\mathbf{U}_k \boldsymbol{\alpha}_k = \mathbf{Q}_k^{\mathsf{T}}(-\mathbf{r}_k)$ via backward substitution
19: $\mathbf{d}_{k+1} \leftarrow \mathbf{d}_{k+1} + \mathbf{B}_k \boldsymbol{\alpha}_k$
20: **end if**
21: **end if**
22: **end if**
23: $k \leftarrow k + 1$
24: **end while**
25: Solve $\mathbf{U}\mathbf{Z}^n = \mathbf{Q}^{\mathsf{T}}$ via backward substitution with m right-hand side vectors
26: Store $\mathbf{D}^n, \mathbf{R}^n, \mathbf{Z}^n$
27: **end function**

Hence, the IQN-IMVLS method also introduces the number of reused time steps q as a parameter. However, it is argued in Spenke et al. (2020) that – due to the implicit reuse of previous data – the IQN-IMVLS method nevertheless overcomes the disadvantage of the IQN-ILS method regarding parameter dependency. For $q = 0$, the IQN-ILS and IQN-IMVLS methods are equivalent. The overall algorithm is summarized in Algorithm 9.6. In the first iteration, a relaxation with relaxation parameter ω_0 is used, which is required only in the first time step if $q > 0$. In summary, the method has three user defined parameters, namely ε, ω_0, q. The multiplication of the inverse Jacobian with a vector in line 8 has complexity $\mathcal{O}(mkq)$, while the QR-decomposition and backward substitution in lines 17, 18, and 25 have complexity $\mathcal{O}(mk^2)$. Assuming that $k \ll m$ and $q \ll m$ and k, q being problem-independent small numbers, the algorithm has therefore $\mathcal{O}(m)$ complexity.

9.6.4 Convergence criterion

As convergence criterion for the partitioned FSI problem, a combination of an absolute criterion with tolerance $\varepsilon_{\mathrm{abs}}$ and a relative criterion with tolerance $\varepsilon_{\mathrm{rel}}$ is used

$$\|\mathbf{r}_k\|_2 < \varepsilon_{\mathrm{abs}} \sqrt{m} \;,$$
$$\|\mathbf{r}_k\|_2 < \varepsilon_{\mathrm{rel}} \, \|\Delta t \, \dot{\mathbf{d}}\|_2 \;,$$

where the results are considered to be converged as soon as one criterion is fulfilled. The absolute criterion is used in Küttler and Wall (2008). However, it might be fulfilled just because the time step size is very small and might therefore not conclusively report the convergence status of the partitioned FSI scheme. For this reason, the absolute criterion is supplemented by the relative one, where $\Delta t \, \dot{\mathbf{d}}$ estimates the change in displacements occurring in one time step due to the motion of the structure with velocity $\dot{\mathbf{d}}$. Hence, the FSI iteration is considered to be converged if the change in displacements after one FSI iteration is small in an absolute sense, or relative as compared to the motion of the structure in the current time step.

Remark 9.10 *It has been remarked in (Küttler and Wall 2008, Remark 7) that strongly-coupled partitioned FSI schemes with multiple iterations per time step require field solvers that can be reset in order to calculate the same time step several times. Here, this is complemented by another important aspect which is motivated from the point of view of computational costs. Due to the nested nature of solvers for partitioned FSI problems, it is important to not "oversolve" the single-field problems. The single-field problems only have to be solved as accurately as necessary, i.e., to not deteriorate the convergence of the partitioned FSI solver. Assuming that the single-field solvers have the highest share of the overall computational costs, a more efficient solution of the single-field problems will directly translate into a more efficient partitioned FSI solver. To enable single-field solvers with coarse relative solver tolerances (reducing residuals by just one or a few orders of magnitude within each partitioned FSI iteration), the single-field solvers must be able to memorize the solution obtained in the previous iteration of the outer partitioned FSI scheme. This is particularly relevant when using splitting methods as single-field solvers, because these then also need to store the solution obtained in intermediate steps of the splitting scheme. If the single-field solvers instead use the same initial guess again and again for each iteration k of the outer partitioned FSI scheme, convergence of the outer FSI iteration can only be guaranteed if the inner single-field problems are solved very accurately. This can be seen in analogy to a Newton–Krylov solver used for a single-field problem, where it is often sufficient to reduce the residual of the linearized problem by one or two orders of magnitude. Another example would be the block-preconditioning of an incompressible flow solver with both velocity and pressure unknowns solved in a monolithic way. In that case, it would also be very expensive to solve the velocity block and pressure block exactly by an iterative method within the preconditioner of the outer Krylov solver. Instead, a single multigrid cycle is sufficient for the purpose of preconditioning and in order to not deteriorate the convergence of the outer solver, being significantly more efficient in terms of computational costs.*

Remark 9.11 *Due to the analogy between block Gauss–Seidel preconditioning of a monolithic system and partitioned Gauss–Seidel iteration schemes (such as the Dirichlet–Neumann partitioning) with iterative solvers for the individual fields, it can be expected that the aspect of not*

oversolving the single-field problems is important to render partitioned FSI solvers competitive to monolithic FSI solvers, e.g. with multigrid block-preconditioning, where the sub-blocks are not solved exactly by an inner solver within the preconditioner but only approximately by one multigrid cycle. However, this aspect has so far barely been discussed in literature addressing comparative studies of monolithic versus partitioned solvers. Note also that the question regarding the computational efficiency of monolithic versus partitioned solvers is difficult to answer for another reason: A main motivation for partitioned solvers is to exploit the most efficient solution techniques available for the individual single-field problems (such as fast projection-type solvers instead of monolithic solvers for the incompressible flow problem), and computationally more efficient preconditioners due to the simpler structure of the algebraic equations.

9.7 Numerical results

This section presents numerical results for classical FSI benchmark problems. As mentioned in the introduction, the aim is to illustrate the flexibility of the proposed FSI framework in term of the interface coupling enabling the use of different discretization techniques for the single-field problems, as well as an investigation of the computational properties of this new FSI solver in terms of the effectiveness of well-known black-box strongly-coupled partitioned iteration schemes that lead to a fast FSI algorithm when combined with efficient single-field solvers.[4] The first example considers the two-dimensional flow around a cylinder with a flexible flag and studies the convergence behavior of different partitioned FSI schemes. The second example simulates a pressure wave in a three-dimensional flexible tube and reports the computational efficiency of the proposed FSI solver compared to a state-of-the-art monolithic FSI solver.

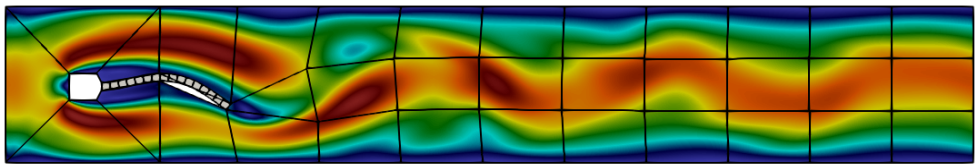
9.7.1 Cylinder-with-flag example

This section considers a well-known FSI benchmark problem proposed in Turek and Hron (2006), flow past a cylindrical obstacle with periodic vortex shedding that amplifies a bending motion of a thin structure attached to the leeward side of the cylinder. The focus of this section is on the convergence behavior of different FSI black-box coupling schemes. The geometry, material parameters, boundary and initial conditions are defined by the benchmark and are not reproduced here, see for example Figure 9.3 for an illustration of the geometry.

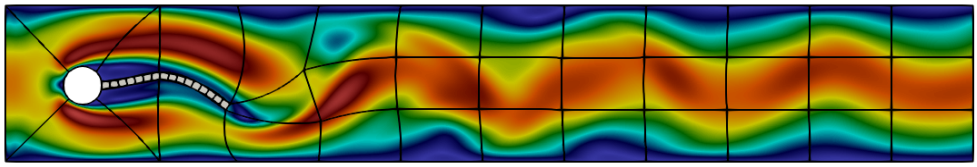
The benchmark configurations FSI-2 and FSI-3 from Turek and Hron (2006) are studied in the following, in particular the benchmark problem FSI-3, since this test case exhibits the highest density ratio ($\rho^{\mathcal{F}}/\rho^{\mathcal{S}} = 1$) between fluid and structure and is therefore expected to be most challenging from an FSI coupling point of view (artificial added-mass effect). The FSI-2 benchmark has a lower density ratio ($\rho^{\mathcal{F}}/\rho^{\mathcal{S}} = 0.1$) but leads to larger deformations of the structure (see also Wick (2011)), so that this benchmark is most interesting to test the robustness of mesh moving algorithms.

Due to the large deformations of the structure in combination with the geometric constraint of a comparably thin channel, it was found that this example is comparably challenging for the mesh deformation solver for the fluid domain, in particular when using coarse meshes with high-order

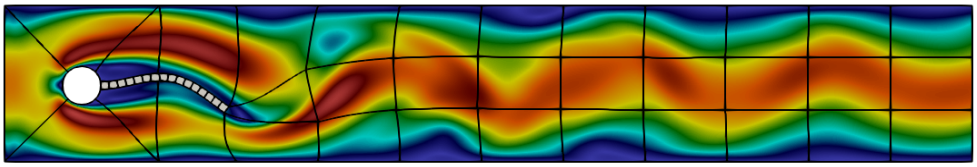
[4] Since every PhD project has to come to an end, an extensive validation of the proposed FSI solver is left for future studies.



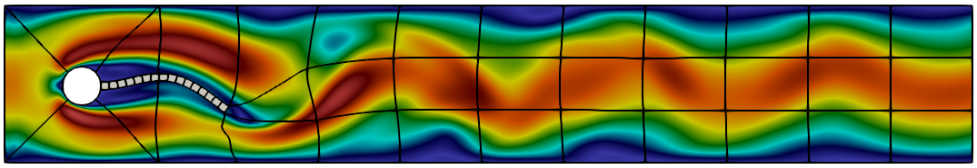
(a) affine mapping $k^{\mathcal{M}} = 1$



(b) quadratic mapping $k^{\mathcal{M}} = 2$



(c) cubic mapping $k^{\mathcal{M}} = 3$



(d) "isoparametric" mapping $k^{\mathcal{M}} = k^{\mathcal{F}} = 6$

Figure 9.4: FSI-2 benchmark: illustration of polynomial degree used for the mapping of the fluid domain for the cylinder-with-flag benchmark problem considering a coarse mesh of refinement level $l = 0$ with polynomial degrees $k^{\mathcal{F}} = 6$ and $k^{\mathcal{S}} = 2$. The magnitude of the fluid velocity is visualized, where blue indicates a low velocity and red a high velocity.

spectral element like discretizations for the fluid. The key ingredient is to use a spatially varying stiffness for the mesh deformation problem, while the question whether a linear or nonlinear elasticity problem is considered was found to be of sub-ordinate importance. For this reason, a linear elasticity problem is solved for the mesh deformation problem with the following spatially varying stiffness

$$E^{\mathcal{M}}(\boldsymbol{\chi}) = \begin{cases} 1 + 100\ \cos^2\left(4\pi\chi_2/H\right) & \text{if } |\chi_2 - H/2| \leq H/8\ , \\ 1 & \text{else}\ . \end{cases} \tag{9.71}$$

and $\nu^{\mathcal{M}} = 0.3$ (and using the plane strain assumption for two-dimensional problems). Here, H is the height of the channel as specified in Turek and Hron (2006). The stiffness is maximal along the centerline of the channel and decreases smoothly to a value of 1 with increasing distance from the centerline.

An illustration of the flow field for benchmark problem FSI-2 is shown in Figure 9.4. A very coarse mesh with refinement level $l = 0$ is chosen in order to visualize the individual elements in the structural domain and the ALE mesh deformation in the fluid domain according to the pseudo-solid elasticity approach. The time snapshots are selected so that the different simulations show a similar state of deformation. The polynomial degrees are $k^{\mathcal{F}} = 6$ for the incompressible flow problem and $k^{\mathcal{S}} = 2$ for the nonlinear elasticity problem. Figure 9.4 compares different polynomial degrees $k^{\mathcal{M}}$ chosen for the ALE mesh deformation problem, considering affine, quadratic, cubic, and isoparametric mappings. Due to the coarse fluid mesh, the cylinder surface and also the deformation of the flexible flag are approximated inaccurately, but the proposed FSI solver and in particular the interface coupling are able to handle such cases, where the rather large gaps between the structural domain and the fluid domain are considered a discretization error. The approximation of the cylinder surface and the bending deformation mode of the flexible structure is significantly improved when using a quadratic mapping. The deformed edges between elements within the fluid domain are also due to the quadratic mapping. Choosing a cubic mapping further improves the approximation quality of the bending mode at the fluid–structure interface, while the results are otherwise similar to the case with quadratic mapping. Finally, the isoparametric mapping illustrates that the high-order mapping may lead to an oscillation-like deformation of edges between elements.

In the following, the convergence behavior of the three partitioned FSI schemes Aitken, IQN-ILS, IQN-IMVLS is studied. The solver tolerances chosen for this example are $\varepsilon_{\text{abs,FSI}} = 10^{-8}, \varepsilon_{\text{rel,FSI}} = 10^{-3}$ for the partitioned FSI schemes and $\varepsilon_{\text{abs,single}} = 10^{-8}, \varepsilon_{\text{rel,single}} = 10^{-6}$ for the single-field solvers (in some cases an absolute tolerance of $\varepsilon_{\text{abs,single}} = 10^{-7}$ was necessary to ensure convergence of the Netwon solver for the nonlinear elasticity problem). The maximum number of partitioned iterations is set to 100 for all acceleration schemes. If the solver tolerances might not be reached within this maximum number of iterations, the time step is considered converged and the solver continues with the next time step. The end time is not defined by the benchmark proposed in Turek and Hron (2006) and an end time of $T = 4L/U_{\text{mean}}$ is chosen here, i.e., 4 flow-through times based on the mean velocity at the inflow and the length of the channel. The dual splitting scheme is used as incompressible Navier–Stokes solver, using adaptive time stepping with $\text{Cr} = 0.5$. Table 9.2 reports the number of partitioned iterations for different acceleration schemes for the FSI-3 benchmark. The Aitken scheme converges in a robust manner with around 5-10 iterations for initial relaxation parameters in the range $0.1 \leq \omega_0 \leq 1.0$. For the quasi-Newton algorithms, the initial relaxation parameter is set to $\omega_0 = 0.3$ and the number of iterations depends significantly on the number of reused time steps q. For $q = 0$, the performance is comparable to the Aitken scheme, while the quasi-Newton algorithms outperform the Aitken scheme significantly for large values of q. The IQN-ILS scheme is able to reduce the number of partitioned iterations by a factor of 2–3 compared to the Aitken scheme. The number of iterations for the IQN-IMVLS scheme is identical to those of the IQN-ILS scheme for $q = 0$ as expected theoretically, and slightly larger compared to the IQN-ILS scheme for $q \geq 1$. Hence, the IQN-ILS scheme appears to be the most effective acceleration scheme in this example.

The above experiment has been repeated for high-order polynomial degrees with $k^{\mathcal{F}} = 6$ for the fluid (velocity), $k^{\mathcal{S}} = 2$ for the structure, and again $k^{\mathcal{M}} = 3$ for the ALE mesh deformation problem. The number of iterations is slightly larger compared to the lower-order case, but the convergence behavior of the different acceleration schemes w.r.t. the parameters studied in Table 9.2 is very similar overall. However, it has been observed that the simulations did not

Table 9.2: FSI-3 benchmark: number of partitioned iterations (averaged over all time steps) for different acceleration schemes and a sequence of mesh refinement levels $l = 0, 1, 2$ using polynomial degrees of $k^{\mathcal{F}} = 3, k^{\mathcal{M}} = 3, k^{\mathcal{S}} = 1$.

(a) Aitken relaxation scheme

l	Initial relaxation parameter ω_0				
	0.1	0.3	0.6	0.9	1.0
0	5.48	4.78	5.62	6.43	6.68
1	7.75	8.36	7.94	9.06	9.34
2	9.31	9.34	10.5	12.0	12.5

(b) IQN-ILS scheme ($\omega_0 = 0.3$)

l	Number of reused time steps q						
	0	1	2	4	8	16	32
0	4.80	3.84	3.43	3.02	2.72	2.54	2.66
1	7.62	4.85	4.23	3.64	3.13	2.86	2.98
2	8.34	5.44	4.70	4.07	3.58	3.27	3.25

(c) IQN-IMVLS scheme ($\omega_0 = 0.3$)

l	Number of reused time steps q						
	0	1	2	4	8	16	32
0	4.80	4.17	3.97	3.72	3.42	3.14	2.93
1	7.62	5.40	5.12	4.78	4.36	4.01	3.66
2	8.34	6.28	6.02	5.66	5.22	4.78	4.42

converge in a robust manner for all parameters, which might be due to the non-robust convergence behavior w.r.t. higher polynomial degrees of the multigrid preconditioner with Chebyshev smoothing used for the elasticity problem, as mentioned in Remark 9.2. To draw precise conclusions and to correctly identify the origin of this lack of robustness, this aspect needs further investigation and should be addressed in the future.

Table 9.3 investigates the impact of the solver tolerance used for the single-field problems within the partitioned FSI scheme by the example of the IQN-ILS acceleration scheme. Relative solver tolerances of $\varepsilon_{\text{rel,single}} = 10^{-6}, 10^{-3}, 10^{-2}, 10^{-1}$ are tested. The number of partitioned FSI iterations degrades only very moderately when using increasingly coarse solver tolerances for the single-field problems. For $\varepsilon_{\text{rel,single}} = 10^{-2}, 10^{-1}$, the simulation did not converge for $q = 0$ with otherwise identical parameters in this example. It requires further investigations to correctly identify where this lack of robustness originates from. Overall, these results suggest that the solver tolerances chosen for the single-field problems are an important parameter w.r.t. the overall efficiency of the partitioned FSI scheme. Note again that this requires an implementation of single-field problems that can memorize the solution obtained in previous iterations of the

Table 9.3: FSI-3 benchmark: number of partitioned iterations for IQN-ILS schemes ($\omega_0 = 0.3$) for different relative solver tolerances used for the single-field problems (considering mesh refinement level $l = 1$ and polynomial degrees of $k^{\mathcal{F}} = 3, k^{\mathcal{M}} = 3, k^{\mathcal{S}} = 1$). The symbol '-' indicates that the simulation did not converge.

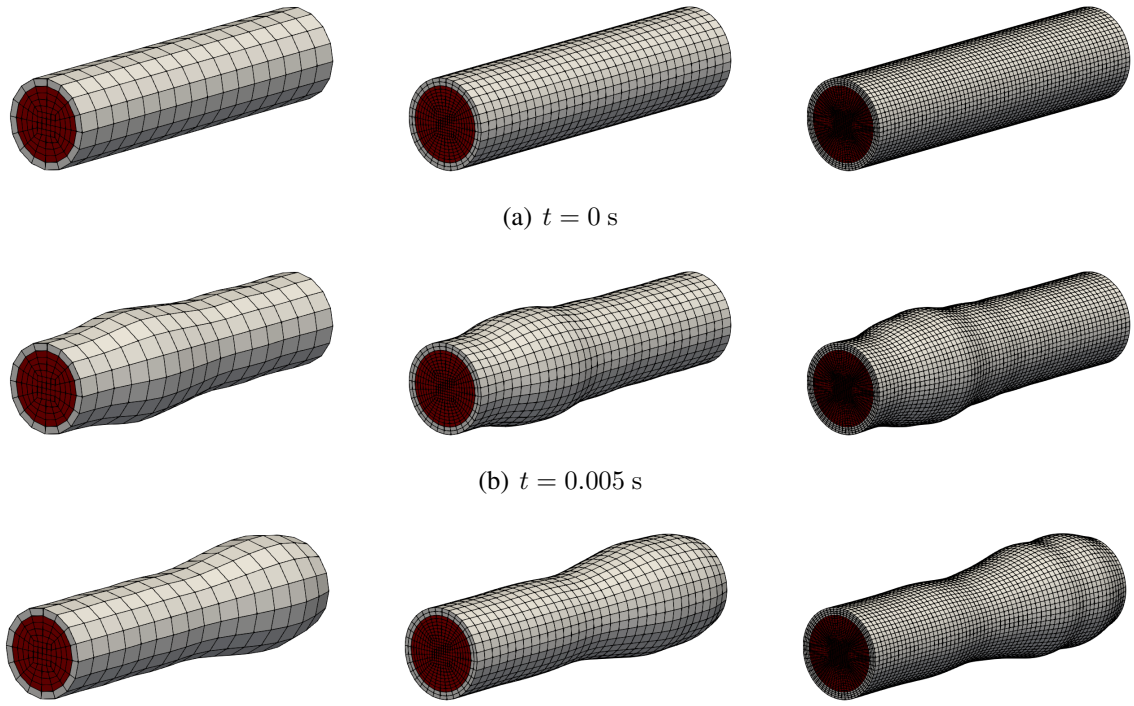
$\varepsilon_{\text{rel,single}}$	Number of reused time steps q						
	0	1	2	4	8	16	32
10^{-6}	7.62	4.85	4.23	3.64	3.13	2.86	2.98
10^{-3}	7.58	4.86	4.24	3.63	3.13	2.88	3.02
10^{-2}	-	5.68	4.90	4.18	3.86	3.41	3.52
10^{-1}	-	5.97	5.24	4.60	3.96	3.71	3.84

coupled FSI scheme in order to guarantee convergence of the outer FSI iteration. While not all black-box solvers for the single-field problems might provide such a functionality, this appears to be a very important design criterion for computationally efficient partitioned FSI schemes, as explained in Remark 9.10.

9.7.2 Pressure wave example

In this section, a widely studied FSI benchmark problem is simulated, the pressure wave example from Fernández et al. (2007), Gerbeau and Vidrascu (2003). In order to mimic hemodynamic conditions, this example exhibits a density ratio of $\rho^{\mathcal{F}}/\rho^{\mathcal{S}} \approx 1$, and, therefore, poses a challenge for many classical FSI coupling schemes. The present work follows the problem setup used in the recent work by Mayr et al. (2020) in order to allow a direct comparison of results on computational efficiency. The domain is a cylindrical tube of length $L = 5$ cm with inner radius $r = 0.5$ cm and outer radius $R = 0.6$ cm where the inner cylinder is filled with fluid. Material parameters are $\mu^{\mathcal{F}} = 0.03\ \text{g}/(\text{cm} \cdot \text{s}), \rho^{\mathcal{F}} = 1\ \text{g/cm}^3$ for the fluid, and $E^{\mathcal{S}} = 3.0 \cdot 10^6\ \text{g}/(\text{cm} \cdot \text{s}^2), \nu^{\mathcal{S}} = 0.3, \rho^{\mathcal{S}} = 1.2\ \text{g/cm}^3$ for the structure (St. Venant–Kirchhoff material). The structure is clamped at both ends of the tube. Regarding the fluid, the pressure is prescribed at one end of the tube and a homogeneous velocity Dirichlet boundary condition is prescribed at the other end. The fluid is at rest initially. A dynamic pressure of $1.3332 \cdot 10^4\ \text{g}/(\text{cm} \cdot \text{s}^2)$ is prescribed at one end of the tube and is imposed over a time interval of 0.003 s, causing a pressure wave traveling through the domain. The end time is $T = 0.02$ s and a fixed time step size of $\Delta t = 10^{-4}$ s is used. Since the deformations are small for this example, a simple Poisson smoothing is sufficient as mesh smoothing algorithm.

The dual splitting scheme is used for the fluid sub-problem. The focus is on computational costs for this three dimensional example. Since linear finite elements are used in Mayr et al. (2020), the lowest possible polynomial degrees are chosen for the present discretization framework, where the pressure shape functions have to be at least linear and the velocity quadratic (inf–sup stability), leading to polynomial degrees of $k^{\mathcal{F}} = 2, k^{\mathcal{M}} = 1, k^{\mathcal{S}} = 1$. Figure 9.5 illustrates the pressure wave traveling through the tube from left to right and the resulting deformations of the structure for three mesh refinement levels $l = 0, 1, 2$. The solver tolerances



(a) $t = 0\,\mathrm{s}$

(b) $t = 0.005\,\mathrm{s}$

(c) $t = 0.01\,\mathrm{s}$

Figure 9.5: 3D pressure wave example: illustration of traveling pressure wave for coarse mesh with refinement level $l = 0$ (left), intermediate mesh with $l = 1$ (middle), and fine mesh with $l = 2$ (right). Displacements are enlarged by a factor of 10.

chosen for this example are $\varepsilon_{\mathrm{abs,FSI}} = 10^{-9}, \varepsilon_{\mathrm{rel,FSI}} = 10^{-3}$ for the partitioned FSI iteration and $\varepsilon_{\mathrm{abs,single}} = 10^{-12}, \varepsilon_{\mathrm{rel,single}} = 10^{-3}$ for the single-field solvers. The IQN-ILS acceleration scheme is used with $\omega_0 = 0.3$ and $q = 10$ due to the convincing performance of this scheme observed for the two-dimensional example investigated above. For the nonlinear elasticity problem, it was found that an AMG preconditioner performs better than the matrix-free geometric multigrid preconditioner with Chebyshev smoothing and with AMG coarse grid solver, and is therefore used in the following. The simulations have been performed on Intel Haswell hardware.

Numerical results are shown in Table 9.4 for three different meshes with refinement levels of $l = 0, 1, 2$. The table lists the total number of unknowns, the number of cores used for each of the simulations, the average number of partitioned FSI iterations, the wall-time per time step, and the throughput per time step. Refinement levels of $l = 0, 1$ yield problem sizes that approximately match those for meshes *pw1*, *pw4* in Mayr et al. (2020). The number of partitioned iterations increases only moderately with increasing mesh refinement level. A throughput of up to 10^5 DoF/s/core is achieved, which is orders of magnitude beyond the numbers collected in Table 9.1 for state-of-the-art monolithic FSI solvers. Compared to results published recently in Mayr et al. (2020) for a monolithic FSI solver, the present solver achieves a throughput that is a factor of 70 higher for the coarse mesh ($l = 0$), and a factor of 110 for the intermediate mesh ($l = 1$). When running the simulation for the intermediate mesh ($l = 1$) on one core, one time step requires a wall-time of 11 seconds, which is lower than the wall-time of 15.8

Table 9.4: 3D pressure wave example: computational efficiency of present FSI solver applied in a low-order regime ($k^{\mathcal{F}} = 2, k^{\mathcal{M}} = 1, k^{\mathcal{S}} = 1$) with performance comparisons to a state-of-the-art monolithic FSI solver that uses continuous linear finite elements. Notation: N_{DoFs} is the total number of unknowns of the three fields (fluid, mesh, structure) and throughput is the number of unknowns solved per time step per core in one second of wall-time.

(a) monolithic FSI solver proposed in Mayr et al. (2020)

mesh	N_{DoFs} [MDoF]	N_{cores}	$t_{\mathrm{wall}}/N_{\Delta \mathrm{t}}$ [s]	Throughput [DoF/s/core]
pw1	0.121	16	5.0	$1.52 \cdot 10^3$
pw4	0.979	128	15.8	$4.84 \cdot 10^2$

(b) present partitioned FSI solver

l	N_{DoFs} [MDoF]	N_{cores}	$N_{\mathrm{iter,FSI}}$	$t_{\mathrm{wall}}/N_{\Delta \mathrm{t}}$ [s]	Throughput [DoF/s/core]
0	0.120	1	6.84	1.13	$1.06 \cdot 10^5$
1	0.954	8	7.70	2.22	$5.37 \cdot 10^4$
2	7.61	48	8.20	5.67	$2.80 \cdot 10^4$

seconds reported in Mayr et al. (2020) for a problem with a similar number of unknowns run on 128 cores (AMD Opteron). Previous works state a superior performance of monolithic over partitioned FSI solvers for this class of problems, see for example Gee et al. (2011), Küttler et al. (2010), Mayr et al. (2015). While these studies put an emphasis on a *fair* comparison in the sense of using the same solution components for both monolithic and partitioned solvers, this aspect might indeed introduce a bias towards monolithic solvers, since this assumption contradicts a main motivation for partitioned schemes, i.e., exploiting the most efficient solution techniques available for the single-field problems. While an in-depth discussion of monolithic versus partitioned FSI schemes is beyond the scope of this work, the present results clearly suggest to re-examine such conclusions from previous works. Given that a performance comparable to the results shown here for a partitioned FSI scheme have not been reported for a monolithic scheme so far to the best of the author's knowledge, the monolithic-vs.-partitioned question appears to be open.

As emphasized in Chapter 6, comparisons in terms of efficiency should consider both accuracy and computational costs. In general, strongly-coupled partitioned FSI solvers can be expected to be comparable in accuracy to monolithic FSI solvers, see for example La Spina et al. (2020b). Assuming that the discretizations used for the single-field problems achieve a similar level of accuracy, the advantage in terms of computational costs reported above can be expected to translate also into a significant advantage in terms of overall efficiency of the FSI solver. The present DG discretization for the fluid appears to be competitive in accuracy (per degree of freedom) to the linear stabilized finite element approach used in Mayr et al. (2020), see for example the results shown in Chapter 2 comparing the accuracy of the present DG fluid discretization to the discretization scheme by Rasthofer and Gravemeier (2013), which is the same type of

fluid discretization as used in Mayr et al. (2020). Finally, the present FSI solver uses a continuous finite element approach for the structure with linear shape functions in this example, which aims at a fair comparison. Note that the speed-up achieved for the present FSI solver compared to Mayr et al. (2020) is consistent with the speed-up achieved in Chapter 6 for a pure fluid problem compared to Rasthofer and Gravemeier (2013). This underlines the above statement that addressing the question regarding the computational efficiency of monolithic vs. partitioned FSI solvers should allow optimizations to be performed independently for monolithic and partitioned solvers.

9.8 Conclusion and outlook

This chapter has proposed a new FSI solver for incompressible flow problems coupled to a nonlinear structure with large deformations. This new framework relies on the design principles of black-box FSI coupling schemes as well as a flexible interface coupling enabling non-matching meshes at the fluid–structure interface and supporting discretization schemes to be selected independently for the fluid and solid problems. Due to the black-box design, this FSI solver allows the use of partitioned schemes in a nested way, i.e., for both the fluid–structure coupling and the velocity–pressure coupling within the fluid sub-problem. Different acceleration schemes have been implemented and a first comparison has been shown, clearly motivating to further investigate and improve partitioned FSI schemes in the future. Given that the examples considered here are academic, it remains for example unclear whether the acceleration schemes studied here are robust FSI coupling schemes for complex engineering examples. All parts of the algorithm have been designed and implemented according to the design principle of fast matrix-free evaluation techniques as a key towards a computationally efficient method. Of course, matrix-based components can and should be used if these prove more efficient for certain parameters (e.g., low polynomial degrees) or a certain type of equations, for which robust efficient matrix-free preconditioners are (currently) not available. A main emphasis of this thesis has been on accurate and efficient computational techniques for the fluid sub-problem, but it is by now unclear to which extent this FSI solver makes use of optimal discretization techniques, implementations, and preconditioners for the solid sub-problem. In particular, it was found that multigrid with Chebyshev smoothing extensively used for the fluid problem does not appear to extend straightforwardly to a robust solver for elasticity problems when using higher polynomial degrees. It appears to be highly relevant to invest dedicated effort into this topic. Performance comparisons to a sophisticated monolithic FSI solver based on linear finite element discretizations have resulted in a significant advantage in computational costs for the proposed matrix-free partitioned FSI scheme. These results are potentially leading the way towards a new generation of FSI solvers and, therefore, motivate to invest further research efforts in the future according to the author's opinion. While convincing single-core or node-level performance has been demonstrated in the present work, the large-scale capabilities of the proposed FSI solver and its parallel scalability should also be investigated in the future. Given that a main focus of this work has been on the simulation of turbulent flows for the fluid sub-problem, an application to turbulent fluid–structure interaction problems could be part of future endeavors.

10 Perspectives

Over the last decade, substantial progress has been made in finite element discretization methods for the incompressible Navier–Stokes equations. In this context, the present work has contributed to the development of stabilized discontinuous Galerkin methods. The L^2-conforming approach is attractive due to its simplicity (compared to tailored finite element spaces), given that it is readily available in most finite element libraries. This approach aims at fulfilling the design principle "Everything should be made as simple as possible, but not simpler." (quote attributed to Albert Einstein) and might prove valuable as a generic turbulent flow solver. Numerical results have demonstrated convincing properties in terms of robustness and accuracy in under-resolved scenarios. The present thesis has put a focus on a holistic view of discretization methods in time and space, iterative solvers and preconditioners, and implementation techniques, with the goal to optimize overall computational efficiency. A fast matrix-free implementation in combination with efficient multigrid methods for preconditioning and efficient Navier–Stokes solution algorithms result in a highly competitive incompressible flow solver. Applications to turbulent flows, natural convection flows, and fluid–structure interaction have been shown. Numerical results have demonstrated that an advantageous combination of these ingredients allows improvements in computational efficiency by one or two orders of magnitude compared to classical state-of-the-art solvers.

The mathematical community currently argues that this type of L^2-conforming discretization approach is not optimal, but that the development of exactly divergence-free and pressure-robust discretizations might be considered the long-standing goal. Not only do these discretization methods provide pressure-independent and visosity-independent error estimates for the velocity, these methods might also pave the way towards generic turbulent flow solvers. While the stabilized L^2-conforming methods investigated in the present thesis have shown a robust behavior, exactly divergence-free approaches have the additional advantage of guaranteed energy stability. These novel discretization methods would render the divergence and normal-continuity penalty terms proposed in the present thesis superfluous. Although the penalty step does not form a bottleneck for the present solver, it might nevertheless cause up to around 50% of the computational costs depending on the problem, indicating some potential for performance improvements. At the same time, the present L^2-conforming approach highly benefits from the fast inversion of mass operators in case of DG formulations, an aspect that is more difficult to realize efficiently for non-DG methods. Against this background and irrespective of the approximation properties of different finite element spaces in the asymptotic regime, it is all but clear which of the two factors will be decisive in terms of computational costs in the end.

The type of matrix-free implementation techniques for tensor-product elements – which has been applied in this thesis to discontinuous Galerkin fluid dynamics solvers – is expected to find widespread use in computational fluid dynamics, and is expected to become a standard ingredient of finite element software over the next decade. While PDE solvers have seen continuous speed-up year-on-year over the last decades due to steady improvements in computer hardware,

it can be expected that Moore's law will enter some form of flattening in the future. Will there be a stagnation at some point in the future, or will new technologies emerge? The CFD community calls for memory bandwidth and reduced network latency. Compute clusters of intermediate size, which improve on these metrics, might be more relevant for the CFD community than striving for ever-increasing Flop rates as known from the race for the world's fastest supercomputer. Energy consumption can be expected to become an increasingly important topic. Due to the trends in computer hardware described above, scientists might feel a stronger pressure in the future in arguing that a particular approach is efficient. Therefore, the importance of benchmarking and a rigorous quantification and documentation of computational costs is expected to grow. According to the author's opinion and as discussed at length in this thesis, the decisive factors that will decide about the success of high-order DG methods as a standard design tool in an industrial context appear to be robustness and accuracy in under-resolved scenarios, node-level performance, and parallel scalability (in terms of minimal wall-time per time step). In the past, the DG community might not have been convincing enough regarding a rigorous quantification of these aspects.

The debate about optimal LES modeling has been lasting over decades. Recent developments and insights into new discretization methods might indeed form some perception that the underlying difficulties are not of physical nature or related to turbulence modeling, but rather of numerical/mathematical origin. This is to be understood in the sense that sophisticated discretization schemes with inbuilt dissipation mechanisms appear to be highly competitive. In this context, a main advantage of robust discretization schemes used as generic turbulent flow solvers appears to be their reduced dependency on parameters, especially in a blinded setup when it comes to predicting rather than reproducing results. The review process of many of the publications related to this thesis has shown that the above perspective is not a fundamental conviction, but rather reflects one part of the scientific spectrum that is currently finding more and more acceptance according to the author's opinion. Common concerns are that LES modeling can not be addressed by moderate-Reynolds-number and academic turbulent flow problems. In order to contribute to this discussion, the present thesis has faced the challenge of trying to obtain grid-converged results for the inviscid Taylor–Green problem with infinite Reynolds number. Yet being academic in nature, it is a stress test for discretization methods in terms of robustness, and a challenge for CFD software in terms of computational costs. The underlying question related to the occurrence of anomalous energy dissipation is also of high physical and mathematical relevance. In the course of this thesis, the author has been most fascinated by this debate, a very interesting topic that currently seems to divide the scientific community.

Future research topics have already been pointed out in previous chapters, of which the most important ones are summarized in the following. Efforts should be made in extending the present matrix-free DG framework for incompressible flows to h-adaptivity in order to improve computational efficiency, and to simplicial elements in order to improve geometric flexibility. Matrix-free preconditioners for general PDE operators such as those arising from fully implicit discretizations in time are another important field of research, in particular when it comes to an application of this matrix-free DG framework in a multi-physics context beyond single-field incompressible flow problems. Ongoing research efforts will be made by the community not only to further optimize matrix-free implementation techniques, but also to keep up with recent hardware developments and increasingly heterogeneous systems. It will also be a future research effort to develop computationally efficient implementations and solvers for novel exactly

divergence-free H^{div}-conforming finite element spaces, algorithms that are by now available for standard H^1-conforming and L^2-conforming methods. Aspects of high-order mesh generation and visualization are currently gaining increasing attention as further ingredients towards an industrialization of high-order discretization methods. A first extension of the present discretization and implementation methodology towards compressible flows has been made in Fehn et al. (2019c). Developing fast solvers for multiphase flows and combustion based on discontinuous Galerkin discretizations with matrix-free implementation techniques could also be an interesting topic. Encouraging performance results have been obtained for fluid–structure interaction solvers that motivate to invest further efforts into this topic.

Bibliography

A. Abbá, L. Bonaventura, M. Nini, and M. Restelli, Dynamic models for Large Eddy Simulation of compressible flows with a high order DG method, *Computers & Fluids* **122**, 209 – 222, 2015.

M. Adams, M. Brezina, J. Hu, and R. Tuminaro, Parallel multigrid smoothing: polynomial versus Gauss–Seidel, *Journal of Computational Physics* **188**, 593 – 610, 2003.

D. S. Agafontsev, E. A. Kuznetsov, and A. A. Mailybaev, Development of high vorticity structures in incompressible 3D Euler equations, *Physics of Fluids* **27**, 085102, 2015.

N. Ahmed, T. C. Rebollo, V. John, and S. Rubino, A review of variational multiscale methods for the simulation of turbulent incompressible flows, *Archives of Computational Methods in Engineering* **24**, 115–164, 2017.

V. Aizinger, D. Kuzmin, and L. Korous, Scale separation in fast hierarchical solvers for discontinuous Galerkin methods, *Applied Mathematics and Computation* **266**, 838 – 849, 2015.

M. Akbas, A. Linke, L. G. Rebholz, and P. W. Schroeder, The analogue of grad–div stabilization in DG methods for incompressible flows: Limiting behavior and extension to tensor-product meshes, *Computer Methods in Applied Mechanics and Engineering* **341**, 917 – 938, 2018.

C. Altmann, A. D. Beck, F. Hindenlang, M. Staudenmaier, G. J. Gassner, and C.-D. Munz, An efficient high performance parallelization of a discontinuous Galerkin spectral element method, In *Facing the Multicore-Challenge III*, pages 37–47, Springer, 2013.

G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmoeller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells, The `deal.II` library, version 9.0, *Journal of Numerical Mathematics* **26**, 173–184, 2018.

R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini, MFEM: A modular finite element methods library, *Computers & Mathematics with Applications*, 2020.

P. Antonietti, M. Sarti, and M. Verani, Multigrid algorithms for hp-discontinuous Galerkin discretizations of elliptic problems, *SIAM Journal on Numerical Analysis* **53**, 598–618, 2015.

P. F. Antonietti, M. Sarti, M. Verani, and L. T. Zikatanov, A uniform additive Schwarz preconditioner for high-order discontinuous Galerkin approximations of elliptic problems, *Journal of Scientific Computing* **70**, 608–630, 2017.

D. Arndt, H. Dallmann, and G. Lube, Local projection FEM stabilization for the time-dependent incompressible Navier–Stokes problem, *Numerical Methods for Partial Differential Equations* **31**, 1224–1250, 2015.

D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells, The deal.II finite element library: Design, features, and insights, *Computers & Mathematics with Applications* **in press**, n/a, 2020a.

D. Arndt, N. Fehn, G. Kanschat, K. Kormann, M. Kronbichler, P. Munch, W. A. Wall, and J. Witte, ExaDG: High-Order Discontinuous Galerkin for the Exa-Scale, In H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, and W. E. Nagel (eds.), *Software for Exascale Computing - SPPEXA 2016-2019*, pages 189–224, Cham, 2020b, Springer International Publishing.

D. N. Arnold, An interior penalty finite element method with discontinuous elements, *SIAM Journal on Numerical Analysis* **19**, 742–760, 1982.

D. N. Arnold, F. Brezzi, B. Cockburn, and D. Marini, Discontinuous Galerkin methods for elliptic problems, In B. Cockburn, G. Karniadakis, and C.-W. Shu (eds.), *Discontinuous Galerkin Methods*, Volume 11 of *Lecture Notes in Computational Science and Engineering*, pages 89–101, Springer Berlin Heidelberg, 2000.

D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, *SIAM Journal on Numerical Analysis* **39**, 1749–1779, 2002.

E. Aulisa, S. Bná, and G. Bornia, A monolithic ALE Newton–Krylov solver with Multigrid-Richardson–Schwarz preconditioning for incompressible fluid-structure interaction, *Computers & Fluids* **174**, 213 – 228, 2018.

S. Badia, F. Nobile, and C. Vergara, Fluid–structure partitioned procedures based on Robin transmission conditions, *Journal of Computational Physics* **227**, 7027 – 7051, 2008a.

S. Badia, A. Quaini, and A. Quarteroni, Splitting methods based on algebraic factorization for fluid-structure interaction, *SIAM Journal on Scientific Computing* **30**, 1778–1805, 2008b.

S. Badia, A. Quaini, and A. Quarteroni, Modular vs. non-modular preconditioners for fluid–structure systems with large added-mass effect, *Computer Methods in Applied Mechanics and Engineering* **197**, 4216 – 4232, 2008c.

H. Baek and G. E. Karniadakis, A convergence study of a new partitioned fluid–structure interaction algorithm based on fictitious mass and damping, *Journal of Computational Physics* **231**, 629 – 652, 2012.

D. H. Bailey, Misleading performance claims in parallel computations, DAC '09, page 528533, New York, NY, USA, 2009, Association for Computing Machinery.

W. Bangerth, R. Hartmann, and G. Kanschat, deal.II — a general purpose object oriented finite element library, *ACM Transactions on Mathematical Software* **33**, article no. 24, 2007.

W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler, Algorithms and data structures for massively parallel generic adaptive finite element codes, *ACM Transactions on Mathematical Software* **38**, 2012.

A. T. Barker and X.-C. Cai, Scalable parallel methods for monolithic coupling in fluid–structure interaction with application to blood flow modeling, *Journal of Computational Physics* **229**, 642 – 659, 2010.

F. Bassi and S. Rebay, A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations, *Journal of Computational Physics* **131**, 267 – 279, 1997.

F. Bassi and S. Rebay, GMRES Discontinuous Galerkin Solution of the Compressible Navier–Stokes Equations, In B. Cockburn, G. E. Karniadakis, and C.-W. Shu (eds.), *Discontinuous Galerkin Methods*, pages 197–208, Berlin, Heidelberg, 2000, Springer Berlin Heidelberg.

F. Bassi, A. Crivellini, D. D. Pietro, and S. Rebay, An artificial compressibility flux for the discontinuous Galerkin solution of the incompressible Navier–Stokes equations, *Journal of Computational Physics* **218**, 794 – 815, 2006.

F. Bassi, A. Ghidoni, S. Rebay, and P. Tesini, High-order accurate p-multigrid discontinuous Galerkin solution of the Euler equations, *International Journal for Numerical Methods in Fluids* **60**, 847–865, 2009.

F. Bassi, N. Franchina, A. Ghidoni, and S. Rebay, Spectral p-multigrid discontinuous Galerkin solution of the Navier–Stokes equations, *International Journal for Numerical Methods in Fluids* **67**, 1540–1558, 2011.

F. Bassi, L. Botti, A. Colombo, A. Crivellini, A. Ghidoni, and F. Massa, On the development of an implicit high-order discontinuous Galerkin method for DNS and implicit LES of turbulent flows, *European Journal of Mechanics - B/Fluids* **55, Part 2**, 367 – 379, 2016.

F. Bassi and S. Rebay, Numerical Solution of the Euler Equations with a Multiorder Discontinuous Finite Element Method, In S. W. Armfield, P. Morgan, and K. Srinivas (eds.), *Computational Fluid Dynamics 2002*, pages 199–204, Berlin, Heidelberg, 2003, Springer Berlin Heidelberg.

F. Bassi, A. Crivellini, D. A. D. Pietro, and S. Rebay, An implicit high-order discontinuous Galerkin method for steady and unsteady incompressible flows, *Computers & Fluids* **36**, 1529 – 1546, 2007.

P. Bastian, M. Blatt, and R. Scheichl, Algebraic multigrid for discontinuous Galerkin discretizations of heterogeneous elliptic problems, *Numerical Linear Algebra with Applications* **19**, 367–388, 2012.

P. Bastian, E. H. Müller, S. Müthing, and M. Piatkowski, Matrix-free multigrid block-preconditioners for higher order discontinuous Galerkin discretisations, *Journal of Computational Physics* **394**, 417 – 439, 2019.

P. Bastian, M. Altenbernd, N.-A. Dreier, C. Engwer, J. Fahlke, R. Fritze, M. Geveler, D. Göddeke, O. Iliev, O. Ippisch, J. Mohring, S. Müthing, M. Ohlberger, D. Ribbrock, N. Shegunov, and S. Turek, Exa-dune—flexible PDE solvers, numerical methods and applications, In H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, and W. E. Nagel (eds.), *Software for Exascale Computing - SPPEXA 2016-2019*, pages 225–269, Cham, 2020a, Springer International Publishing.

P. Bastian, M. Blatt, A. Dedner, N.-A. Dreier, C. Engwer, R. Fritze, C. Gräser, C. Grüninger, D. Kempf, R. Klöfkorn, M. Ohlberger, and O. Sander, The Dune framework: Basic concepts and recent developments, *Computers & Mathematics with Applications*, 2020b.

J. T. Batina, Unsteady Euler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysis, *AIAA Journal* **29**, 327–333, 1991.

S. Bauer, H.-P. Bunge, D. Drzisga, S. Ghelichkhan, M. Huber, N. Kohl, M. Mohr, U. Rüde, D. Thönnes, and B. Wohlmuth, TerraNeo—Mantle Convection Beyond a Trillion Degrees of Freedom, In H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, and W. E. Nagel (eds.), *Software for Exascale Computing - SPPEXA 2016-2019*, pages 569–610, Cham, 2020, Springer International Publishing.

C. E. Baumann and J. T. Oden, A discontinuous hp finite element method for the Euler and Navier–Stokes equations, *International Journal for Numerical Methods in Fluids* **31**, 79–95, 1999.

J. T. Beale, T. Kato, and A. Majda, Remarks on the breakdown of smooth solutions for the 3-D Euler equations, *Communications in Mathematical Physics* **94**, 61–66, 1984.

A. D. Beck, T. Bolemann, D. Flad, H. Frank, G. J. Gassner, F. Hindenlang, and C.-D. Munz, High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations, *International Journal for Numerical Methods in Fluids* **76**, 522–548, 2014.

G. Bell, D. H. Bailey, J. Dongarra, A. H. Karp, and K. Walsh, A look back on 30 years of the Gordon Bell prize, *The International Journal of High Performance Computing Applications* **31**, 469–484, 2017.

M. Benzi, G. H. Golub, and J. Liesen, Numerical solution of saddle point problems, *Acta numerica* **14**, 1–137, 2005.

A. Beskok and T. C. Warburton, An unstructured hp finite-element scheme for fluid flow and heat transfer in moving domains, *Journal of Computational Physics* **174**, 492 – 509, 2001.

A. Bogaers, S. Kok, B. Reddy, and T. Franz, Quasi-Newton methods for implicit black-box FSI coupling, *Computer Methods in Applied Mechanics and Engineering* **279**, 113 – 132, 2014.

O. N. Boratav and R. B. Pelz, Direct numerical simulation of transition to turbulence from a highsymmetry initial condition, *Physics of Fluids* **6**, 2757–2784, 1994.

L. Botti and D. A. D. Pietro, A pressure-correction scheme for convection-dominated incompressible flows with discontinuous velocity and continuous pressure, *Journal of Computational Physics* **230**, 572 – 585, 2011.

M. E. Brachet, M. Meneguzzi, A. Vincent, H. Politano, and P. L. Sulem, Numerical evidence of smooth self-similar dynamics and possibility of subsequent collapse for three-dimensional ideal flows, *Physics of Fluids A: Fluid Dynamics* **4**, 2845–2854, 1992.

M. E. Brachet, D. I. Meiron, S. A. Orszag, B. Nickel, R. H. Morf, and U. Frisch, Small-scale structure of the Taylor–Green vortex, *Journal of Fluid Mechanics* **130**, 411–452, 1983.

M. Brachet, Direct simulation of three-dimensional turbulence in the Taylor–Green vortex, *Fluid dynamics research* **8**, 1–8, 1991.

Y. Brenier, C. De Lellis, and L. Székelyhidi, Weak-strong uniqueness for measure-valued solutions, *Communications in mathematical physics* **305**, 351–361, 2011.

M. P. Brenner, S. Hormoz, and A. Pumir, Potential singularity mechanism for the Euler equations, *Physical Review Fluids* **1**, 084503, 2016.

S. C. Brenner, J. Cui, and L.-Y. Sung, Multigrid methods for the symmetric interior penalty method on graded meshes, *Numerical Linear Algebra with Applications* **16**, 481–501, 2009.

S. C. Brenner and J. Zhao, Convergence of multigrid algorithms for interior penalty methods, *Applied Numerical Analysis & Computational Mathematics* **2**, 3–18, 2005.

M. Breuer, G. De Nayer, M. Münsch, T. Gallinger, and R. Wüchner, Fluid–structure interaction using a partitioned semi-implicit predictor–corrector coupling scheme for the application of large-eddy simulation, *Journal of Fluids and Structures* **29**, 107 – 130, 2012.

A. N. Brooks and T. J. Hughes, Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations, *Computer Methods in Applied Mechanics and Engineering* **32**, 199 – 259, 1982.

D. L. Brown, Performance of under-resolved two-dimensional incompressible flow simulations, *Journal of Computational Physics* **122**, 165 – 183, 1995.

J. Brown, Efficient Nonlinear Solvers for Nodal High-Order Finite Elements in 3D, *Journal of Scientific Computing* **45**, 48–63, 2010.

T. Buckmaster and V. Vicol, Nonuniqueness of weak solutions to the Navier–Stokes equation, *Annals of mathematics* **189**, 101–144, 2019.

T. Buckmaster and V. Vicol, Convex integration and phenomenologies in turbulence, *EMS Surveys in Mathematical Sciences* **6**, 173–263, 2020.

T. Buckmaster, C. De Lellis, L. Székelyhidi, and V. Vicol, Onsager's conjecture for admissible weak solutions, *Communications on Pure and Applied Mathematics* **72**, 229–274, 2018.

T. Buckmaster, N. Masmoudi, M. Novack, and V. Vicol, Non-conservative $H^{1/2-}$ weak solutions of the incompressible 3D Euler equations, *arXiv preprint arXiv:2101.09278*, 2021.

T. Buckmaster, C. De Lellis, and L. Székelyhidi Jr., Dissipative Euler flows with Onsager-critical spatial regularity, *Communications on Pure and Applied Mathematics* **69**, 1613–1670, 2016.

J. Burgers, A mathematical model illustrating the theory of turbulence, Volume 1 of *Advances in Applied Mechanics*, pages 171 – 199, Elsevier, 1948.

C. Burstedde, L. C. Wilcox, and O. Ghattas, p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, *SIAM Journal on Scientific Computing* **33**, 1103–1133, 2011.

M. D. Bustamante and M. Brachet, Interplay between the Beale–Kato–Majda theorem and the analyticity-strip method to investigate numerically the incompressible Euler singularity problem, *Physical Review E* **86**, 066302, 2012.

S. Busto, M. Tavelli, W. Boscheri, and M. Dumbser, Efficient high order accurate staggered semi-implicit discontinuous Galerkin methods for natural convection problems, *Computers & Fluids* **198**, 104399, 2020.

J. Cahouet and J.-P. Chabard, Some fast 3d finite element solvers for the generalized Stokes problem, *International Journal for Numerical Methods in Fluids* **8**, 869–895, 1988.

C. S. Campolina and A. A. Mailybaev, Chaotic blowup in the 3D incompressible Euler equations on a logarithmic lattice, *Physical Review Letters* **121**, 064501, 2018.

C. Cantwell, S. Sherwin, R. Kirby, and P. Kelly, From h to p efficiently: Strategy selection for operator evaluation on hexahedral and tetrahedral elements, *Computers & Fluids* **43**, 23 – 28, 2011.

C. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. D. Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. Kirby, and S. Sherwin, Nektar++: An open-source spectral/hp element framework, *Computer Physics Communications* **192**, 205 – 219, 2015.

C. Carton de Wiart, K. Hillewaert, M. Duponcheel, and G. Winckelmans, Assessment of a discontinuous Galerkin method for the simulation of vortical flows at high Reynolds number, *International Journal for Numerical Methods in Fluids* **74**, 469–493, 2014.

P. Causin, J. Gerbeau, and F. Nobile, Added-mass effect in the design of partitioned algorithms for fluid–structure problems, *Computer Methods in Applied Mechanics and Engineering* **194**, 4506 – 4527, 2005.

N. Chalmers, G. Agbaglah, M. Chrust, and C. Mavriplis, A parallel hp-adaptive high order discontinuous Galerkin method for the incompressible Navier–Stokes equations, *Journal of Computational Physics: X* **2**, 100023, 2019.

J. Chang, M. S. Fabien, M. G. Knepley, and R. T. Mills, Comparative study of finite element methods using the Time-Accuracy-Size (TAS) spectrum analysis, *SIAM Journal on Scientific Computing* **40**, C779–C802, 2018.

J.-B. Chapelier, M. de la Llave Plata, F. Renac, and E. Lamballais, Evaluation of a high-order discontinuous Galerkin method for the DNS of turbulent flows, *Computers & Fluids* **95**, 210 – 226, 2014.

J.-B. Chapelier, M. D. L. L. Plata, and E. Lamballais, Development of a multiscale LES model in the context of a modal discontinuous Galerkin method, *Computer Methods in Applied Mechanics and Engineering* **307**, 275 – 299, 2016.

J.-B. Chapelier, M. De La Llave Plata, and F. Renac, Inviscid and viscous simulations of the Taylor–Green vortex flow using a modal discontinuous Galerkin approach, In *42nd AIAA Fluid Dynamics Conference and Exhibit*, page 3073, 2012.

A. Cheskidov, P. Constantin, S. Friedlander, and R. Shvydkoy, Energy conservation and Onsager's conjecture for the Euler equations, *Nonlinearity* **21**, 1233–1252, 2008.

S. Chippada, C. Dawson, M. Martnez, and M. Wheeler, A projection method for constructing a mass conservative velocity field, *Computer Methods in Applied Mechanics and Engineering* **157**, 1 – 10, 1998.

A. J. Chorin, Numerical solution of the Navier–Stokes equations, *Mathematics of Computation* **22**, 745–762, 1968.

J. Chung and G. M. Hulbert, A Time Integration Algorithm for Structural Dynamics With Improved Numerical Dissipation: The Generalized-α Method, *Journal of Applied Mechanics* **60**, 371–375, 1993.

C. Cichowlas and M.-E. Brachet, Evolution of complex singularities in Kida–Pelz and Taylor–Green inviscid flows, *Fluid Dynamics Research* **36**, 239–248, 2005.

C. Cichowlas, P. Bonaïti, F. Debbasch, and M. Brachet, Effective dissipation and turbulence in spectrally truncated Euler flows, *Physical Review Letters* **95**, 264502, 2005.

T. C. Clevenger, T. Heister, G. Kanschat, and M. Kronbichler, A flexible, parallel, adaptive geometric multigrid method for FEM, *ACM Transactions on Mathematical Software* **47**, 2020.

B. Cockburn, G. Kanschat, D. Schötzau, and C. Schwab, Local discontinuous Galerkin methods for the Stokes system, *SIAM Journal on Numerical Analysis* **40**, 319–343, 2002.

B. Cockburn, G. Kanschat, and D. Schötzau, The local discontinuous Galerkin method for the Oseen equations, *Mathematics of Computation* **73**, 569–593, 2004.

B. Cockburn, G. Kanschat, and D. Schötzau, A locally conservative LDG method for the incompressible Navier–Stokes equations, *Mathematics of Computation* **74**, 1067–1095, 2005.

B. Cockburn, G. Kanschat, and D. Schötzau, A note on discontinuous Galerkin divergence-free solutions of the Navier–Stokes equations, *Journal of Scientific Computing* **31**, 61–73, 2007.

B. Cockburn, G. Kanschat, and D. Schötzau, An equal-order DG method for the incompressible Navier–Stokes equations, *Journal of Scientific Computing* **40**, 188–210, 2009.

S. S. Collis, Discontinuous Galerkin methods for turbulence simulation, In *Proceedings of the 2002 Center for Turbulence Research Summer Program*, page 115167, 2002.

P. Constantin, W. E, and E. S. Titi, Onsager's conjecture on the energy conservation for solutions of Euler's equation, *Communications in Mathematical Physics* **165**, 207, 1994.

G. Coppola, F. Capuano, and L. de Luca, Discrete Energy-Conservation Properties in the Numerical Simulation of the Navier–Stokes Equations, *Applied Mechanics Reviews* **71**, 2019.

W. Couzy and M. O. Deville, Spectral-element preconditioners for the Uzawa pressure operator applied to incompressible flows, *Journal of Scientific Computing* **9**, 107–122, 1994.

W. Couzy and M. Deville, A Fast Schur Complement Method for the Spectral Element Discretization of the Incompressible Navier-Stokes Equations, *Journal of Computational Physics* **116**, 135 – 142, 1995.

S. Daneri, E. Runa, and L. Székelyhidi, Non-uniqueness for the Euler equations up to Onsager's critical exponent, *Annals of PDE* **7**, 1–44, 2021.

D. Davydov, J.-P. Pelteret, D. Arndt, M. Kronbichler, and P. Steinmann, A matrix-free approach for finite-strain hyperelastic problems using geometric multigrid, *International Journal for Numerical Methods in Engineering* **121**, 2874–2895, 2020.

C. Dawson, S. Sun, and M. F. Wheeler, Compatible algorithms for coupled flow and transport, *Computer Methods in Applied Mechanics and Engineering* **193**, 2565 – 2580, 2004.

A. de Boer, M. van der Schoot, and H. Bijl, Mesh deformation based on radial basis function interpolation, *Computers & Structures* **85**, 784 – 795, 2007.

M. de la Llave Plata, V. Couaillier, and M.-C. le Pape, On the use of a high-order discontinuous Galerkin method for DNS and LES of wall-bounded turbulence, *Computers & Fluids*, 2017.

C. De Lellis and L. Székelyhidi Jr., Dissipative continuous Euler flows, *Inventiones mathematicae* **193**, 377–407, 2013.

C. De Lellis and L. Székelyhidi Jr., Dissipative Euler flows and Onsager's conjecture, *Journal of the European Mathematical Society* **16**, 1467–1505, 2014.

J. DeBonis, Solutions of the Taylor–Green vortex problem using high-resolution explicit finite difference methods, In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 382, 2013.

J. Degroote, K.-J. Bathe, and J. Vierendeels, Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction, *Computers & Structures* **87**, 793 – 801, 2009.

J. Degroote, R. Haelterman, S. Annerel, P. Bruggeman, and J. Vierendeels, Performance of partitioned procedures in fluid–structure interaction, *Computers & Structures* **88**, 446 – 457, 2010.

J. C. Del Alamo, J. Jiménez, P. Zandonade, and R. D. Moser, Scaling of the energy spectra of turbulent channels, *Journal of Fluid Mechanics* **500**, 135–144, 2004.

M. Deville and E. Mund, Chebyshev pseudospectral solution of second-order elliptic equations with finite element preconditioning, *Journal of Computational Physics* **60**, 517 – 533, 1985.

M. Deville and E. Mund, Finite-Element Preconditioning for Pseudospectral Solutions of Elliptic Problems, *SIAM Journal on Scientific and Statistical Computing* **11**, 311–342, 1990.

M. O. Deville, P. F. Fischer, and E. H. Mund, *High-order methods for incompressible fluid flow*, Volume 9, Cambridge University Press, 2002.

L. T. Diosady and D. L. Darmofal, Preconditioning methods for discontinuous Galerkin solutions of the Navier–Stokes equations, *Journal of Computational Physics* **228**, 3917 – 3935, 2009.

R. J. DiPerna and A. J. Majda, Oscillations and concentrations in weak solutions of the incompressible fluid equations, *Communications in mathematical physics* **108**, 667–689, 1987.

V. A. Dobrev, R. D. Lazarov, P. S. Vassilevski, and L. T. Zikatanov, Two-level preconditioning of discontinuous Galerkin approximations of second-order elliptic equations, *Numerical Linear Algebra with Applications* **13**, 753–770, 2006.

T. Dockhorn. Turbulence modeling for large eddy simulation of incompressible flows using high-order discontinuous Galerkin methods. Bachelor's thesis, Technical University of Munich, 2017.

J. Donea, P. Fasoli-Stella, and S. Giuliani, Lagrangian and Eulerian finite element techniques for transient fluid-structure interaction problems, *Therm and Fluid/Struct Dyn Anal*, 1977.

J. Donea, S. Giuliani, and J. Halleux, An arbitrary Lagrangian–Eulerian finite element method for transient dynamic fluid-structure interactions, *Computer Methods in Applied Mechanics and Engineering* **33**, 689 – 723, 1982.

J. Donea and A. Huerta, *Finite element methods for flow problems*, John Wiley & Sons, 2003.

J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodrguez-Ferran, pages 1–23, *Arbitrary Lagrangian–Eulerian Methods*, , American Cancer Society, 2017.

O. Dorok, W. Grambow, and L. Tobiska, Aspects of finite element discretizations for solving the Boussinesq approximation of the Navier–Stokes equations, In *Numerical methods for the Navier–Stokes equations*, pages 50–61, Springer, 1994.

T. D. Drivas and G. L. Eyink, An Onsager singularity theorem for Leray solutions of incompressible Navier–Stokes, *Nonlinearity* **32**, 4465, 2019.

T. D. Drivas and H. Q. Nguyen, Remarks on the emergence of weak Euler solutions in the vanishing viscosity limit, *Journal of Nonlinear Science* **29**, 709–721, 2019.

B. Dubrulle, Beyond Kolmogorov cascades, *Journal of Fluid Mechanics* **867**, P1, 2019.

J. Duchon and R. Robert, Inertial energy dissipation for weak solutions of incompressible Euler and Navier–Stokes equations, *Nonlinearity* **13**, 249–255, 2000.

M. Durufle, P. Grob, and P. Joly, Influence of Gauss and Gauss–Lobatto quadrature rules on the accuracy of a quadrilateral finite element method in the time domain, *Numerical Methods for Partial Differential Equations* **25**, 526–551, 2009.

J. Eggers, Role of singularities in hydrodynamics, *Physical Review Fluids* **3**, 110503, 2018.

H. Elman and D. Silvester, Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations, *SIAM Journal on Scientific Computing* **17**, 33–46, 1996.

H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, Block preconditioners based on approximate commutators, *SIAM Journal on Scientific Computing* **27**, 1651–1668, 2006.

H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations, *Journal of Computational Physics* **227**, 1790–1808, 2008.

H. C. Elman, Preconditioning for the Steady-State Navier–Stokes Equations with Low Viscosity, *SIAM Journal on Scientific Computing* **20**, 1299–1316, 1999.

H. C. Elman, D. J. Silvester, and A. J. Wathen, Block preconditioners for the discrete incompressible Navier–Stokes equations, *International Journal for Numerical Methods in Fluids* **40**, 333–344, 2002a.

H. C. Elman, D. J. Silvester, and A. J. Wathen, Performance and analysis of saddle point preconditioners for the discrete steady-state Navier–Stokes equations, *Numerische Mathematik* **90**, 665–688, 2002b.

H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*, Oxford University Press, USA, 2014.

N. Emamy, *Numerical simulation of deformation of a droplet in a stationary electric field using DG*, PhD thesis, Technische Universität Darmstadt, 2014.

N. Emamy, F. Kummer, M. Mrosek, M. Karcher, and M. Oberlack, Implicit-explicit and explicit projection schemes for the unsteady incompressible Navier–Stokes equations using a high-order dG method, *Computers & Fluids* **154**, 285 – 295, 2017.

C. R. Ethier and D. Steinman, Exact fully 3D Navier–Stokes solution for benchmarking, *International Journal for Numerical Methods in Fluids* **19**, 369–376, 1994.

S. Étienne, A. Garon, and D. Pelletier, Perspective on the geometric conservation law and finite element methods for ALE simulations of incompressible flow, *Journal of Computational Physics* **228**, 2313 – 2333, 2009.

G. L. Eyink, Energy dissipation without viscosity in ideal hydrodynamics i. Fourier analysis and local energy transfer, *Physica D: Nonlinear Phenomena* **78**, 222 – 240, 1994.

G. L. Eyink, Dissipative anomalies in singular Euler flows, *Physica D: Nonlinear Phenomena* **237**, 1956 – 1968, 2008.

G. L. Eyink and K. R. Sreenivasan, Onsager and the theory of hydrodynamic turbulence, *Reviews of modern physics* **78**, 87–135, 2006.

M. Fabien, M. Knepley, R. Mills, and B. Rivière, Manycore parallel computing for a hybridizable discontinuous Galerkin nested multigrid method, *SIAM Journal on Scientific Computing* **41**, C73–C96, 2019.

F. Fambri and M. Dumbser, Spectral semi-implicit and space–time discontinuous Galerkin methods for the incompressible Navier–Stokes equations on staggered Cartesian grids, *Applied Numerical Mathematics* **110**, 41 – 74, 2016.

C. Farhat and P. Geuzaine, Design and analysis of robust ALE time-integrators for the solution of unsteady flow problems on moving grids, *Computer Methods in Applied Mechanics and Engineering* **193**, 4073 – 4095, 2004.

C. Farhat, C. Degand, B. Koobus, and M. Lesoinne, Torsional springs for two-dimensional dynamic unstructured fluid meshes, *Computer Methods in Applied Mechanics and Engineering* **163**, 231–245, 1998.

N. Fehn. A discontinuous Galerkin approach for the unsteady incompressible Navier–Stokes equations. Master's thesis, Technical University of Munich, 2015.

N. Fehn, W. A. Wall, and M. Kronbichler, On the stability of projection methods for the incompressible Navier–Stokes equations based on high-order discontinuous Galerkin discretizations, *Journal of Computational Physics* **351**, 392–421, 2017.

N. Fehn, W. A. Wall, and M. Kronbichler, Efficiency of high-performance discontinuous Galerkin spectral element methods for under-resolved turbulent incompressible flows, *International Journal for Numerical Methods in Fluids* **88**, 32–54, 2018a.

N. Fehn, W. A. Wall, and M. Kronbichler, Robust and efficient discontinuous Galerkin methods for under-resolved turbulent incompressible flows, *Journal of Computational Physics* **372**, 667–693, 2018b.

N. Fehn, M. Kronbichler, C. Lehrenfeld, G. Lube, and P. W. Schroeder, High-order DG solvers for under-resolved turbulent incompressible flows: A comparison of L^2 and $H(\text{div})$ methods, *International Journal for Numerical Methods in Fluids* **91**, 533–556, 2019a.

N. Fehn, W. A. Wall, and M. Kronbichler, Modern discontinuous Galerkin methods for the simulation of transitional and turbulent flows in biomedical engineering: A comprehensive LES study of the FDA benchmark nozzle model, *International Journal for Numerical Methods in Biomedical Engineering* **35**, e3228, 2019b.

N. Fehn, W. A. Wall, and M. Kronbichler, A matrix-free high-order discontinuous Galerkin compressible Navier–Stokes solver: A performance comparison of compressible and incompressible formulations for turbulent incompressible flows, *International Journal for Numerical Methods in Fluids* **89**, 71–102, 2019c.

N. Fehn, P. Munch, W. A. Wall, and M. Kronbichler, Hybrid multigrid methods for high-order discontinuous Galerkin discretizations, *Journal of Computational Physics* **415**, 109538, 2020.

N. Fehn, J. Heinz, W. A. Wall, and M. Kronbichler, High-order arbitrary Lagrangian–Eulerian discontinuous Galerkin methods for the incompressible Navier–Stokes equations, *Journal of Computational Physics* **430**, 110040, 2021a.

N. Fehn, M. Kronbichler, P. Munch, and W. A. Wall, Numerical evidence of anomalous energy dissipation in incompressible Euler flows: Towards grid-converged results for the inviscid Taylor–Green problem, *Journal of Fluid Mechanics* **accepted**, arXiv preprint arXiv:2007.01656, 2021b.

M. A. Fernández and M. Moubachir, A Newton method using exact jacobians for solving fluid–structure coupling, *Computers & Structures* **83**, 127 – 142, 2005.

M. A. Fernández, J.-F. Gerbeau, and C. Grandmont, A projection semi-implicit scheme for the coupling of an elastic structure with an incompressible fluid, *International Journal for Numerical Methods in Engineering* **69**, 794–821, 2007.

P. Fernandez, N. C. Nguyen, and J. Peraire, On the ability of discontinuous Galerkin methods to simulate under-resolved turbulent flows, *arXiv preprint arXiv:1810.09435*, 2018.

E. Ferrer and R. H. J. Willden, A high order discontinuous Galerkin finite element solver for the incompressible Navier–Stokes equations, *Computers & Fluids* **46**, 224 – 230, 2011.

E. Ferrer, D. Moxey, R. H. J. Willden, and S. J. Sherwin, Stability of projection methods for incompressible flows using high order pressure-velocity pairs of same degree: Continuous and discontinuous Galerkin formulations, *Commun. Comput. Phys.* **16**, 817–840, 2014.

E. Ferrer, N. Saito, H. Blackburn, and D. Pullin, High-Reynolds-number wall-modelled large eddy simulations of turbulent pipe flows using explicit and implicit subgrid stress treatments within a spectral element solver, *Computers & Fluids* **191**, 104239, 2019.

E. Ferrer, *A high order Discontinuous Galerkin–Fourier incompressible 3D Navier–Stokes solver with rotating sliding meshes for simulating cross-flow turbines*, PhD thesis, University of Oxford, 2012.

E. Ferrer, An interior penalty stabilised incompressible discontinuous Galerkin–Fourier solver for implicit large eddy simulations, *Journal of Computational Physics* **348**, 754 – 775, 2017.

E. Ferrer and R. H. J. Willden, A high order discontinuous Galerkin–Fourier incompressible 3D Navier–Stokes solver with rotating sliding meshes, *Journal of Computational Physics* **231**, 7037 – 7056, 2012.

J. H. Ferziger and M. Peric, *Numerische Strömungsmechanik*, Springer-Verlag, 2008.

K. Fidkowski and D. Darmofal, Development of a higher-order solver for aerodynamic applications, In *42nd AIAA Aerospace Sciences Meeting and Exhibit*, page 436, 2004.

K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal, p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, *Journal of Computational Physics* **207**, 92 – 113, 2005.

P. F. Fischer, H. M. Tufo, and N. I. Miller, An Overlapping Schwarz Method for Spectral Element Simulation of Three-Dimensional Incompressible Flows, In P. Bjørstad and M. Luskin (eds.), *Parallel Solution of Partial Differential Equations*, pages 159–180, New York, NY, 2000, Springer New York.

P. Fischer and J. Mullen, Filter-based stabilization of spectral element methods, *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics* **332**, 265 – 270, 2001.

P. Fischer, S. Kerkemeier, A. Peplinski, D. Shaver, A. Tomboulides, M. Min, A. Obabko, and E. Merzari. NEK5000 Web page, 2020a. https://nek5000.mcs.anl.gov.

P. Fischer, M. Min, T. Rathnayake, S. Dutta, T. Kolev, V. Dobrev, J.-S. Camier, M. Kronbichler, T. Warburton, K. Świrydowicz, and J. Brown, Scalability of high-performance PDE solvers, *The International Journal of High Performance Computing Applications* **34**, 562–586, 2020b.

P. F. Fischer, Analysis and application of a parallel spectral element method for the solution of the Navier–Stokes equations, *Computer Methods in Applied Mechanics and Engineering* **80**, 483 – 491, 1990.

P. F. Fischer, An Overlapping Schwarz Method for Spectral Element Solution of the Incompressible Navier-Stokes Equations, *Journal of Computational Physics* **133**, 84 – 101, 1997.

P. F. Fischer, Scaling limits for PDE-based simulation, In *22nd AIAA Computational Fluid Dynamics Conference*, page 3049, 2015.

P. F. Fischer and J. W. Lottes, Hybrid Schwarz-Multigrid Methods for the Spectral Element Method: Extensions to Navier-Stokes, In T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, T. Schlick, R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering*, pages 35–49, Berlin, Heidelberg, 2005, Springer Berlin Heidelberg.

P. F. Fischer and A. T. Patera, Parallel spectral element solution of the Stokes problem, *Journal of Computational Physics* **92**, 380 – 421, 1991.

P. F. Fischer, L.-W. Ho, G. E. Karniadakis, E. M. Rønquist, and A. T. Patera, Recent advances in parallel spectral element simulation of unsteady incompressible flows, In A. K. NOOR and D. L. DWOYER (eds.), *Computational Structural Mechanics & Fluid Dynamics*, pages 217 – 231, Pergamon, Oxford, 1988.

P. F. Fischer, G. W. Kruse, and F. Loth, Spectral element methods for transitional flows in complex geometries, *Journal of Scientific Computing* **17**, 81–98, 2002.

D. Flad, A. Beck, and C.-D. Munz, Simulation of underresolved turbulent flows by adaptive filtering using the high order discontinuous Galerkin spectral element method, *Journal of Computational Physics* **313**, 1 – 12, 2016.

D. Flad and G. Gassner, On the use of kinetic energy preserving DG-schemes for large eddy simulation, *Journal of Computational Physics* **350**, 782 – 795, 2017.

C. Förster, W. A. Wall, and E. Ramm, On the geometric conservation law in transient flow calculations on deforming domains, *International Journal for Numerical Methods in Fluids* **50**, 1369–1379, 2006.

C. Förster, W. A. Wall, and E. Ramm, Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows, *Computer Methods in Applied Mechanics and Engineering* **196**, 1278 – 1293, 2007.

D. Forti, A. Quarteroni, and S. Deparis, A parallel algorithm for the solution of large-scale nonconforming fluid–structure interaction problems in hemodynamics, *Journal of Computational Mathematics* **35**, 363–380, 2017.

L. P. Franca and T. J. Hughes, Two classes of mixed finite element methods, *Computer Methods in Applied Mechanics and Engineering* **69**, 89 – 129, 1988.

M. Franciolini, A. Crivellini, and A. Nigro, On the efficiency of a matrix-free linearly implicit time integration strategy for high-order Discontinuous Galerkin solutions of incompressible turbulent flows, *Computers & Fluids* **159**, 276 – 294, 2017.

M. Franciolini, L. Botti, A. Colombo, and A. Crivellini, p-Multigrid matrix-free discontinuous Galerkin solution strategies for the under-resolved simulation of incompressible turbulent flows, *Computers & Fluids* **206**, 104558, 2020.

M. Franco, J.-S. Camier, J. Andrej, and W. Pazner, High-order matrix-free incompressible flow solvers with GPU acceleration and low-order refined preconditioners, *Computers & Fluids* **203**, 104541, 2020.

M. Frigo and S. G. Johnson, The Design and Implementation of FFTW3, *Proceedings of the IEEE* **93**, 216–231, 2005.

B. Froehle and P.-O. Persson, A high-order discontinuous Galerkin method for fluid–structure interaction with efficient implicit–explicit time stepping, *Journal of Computational Physics* **272**, 455 – 470, 2014.

B. Froehle and P.-O. Persson, Nonlinear elasticity for mesh deformation with high-order discontinuous Galerkin methods for the Navier–Stokes equations on deforming domains, In R. M. Kirby, M. Berzins, and J. S. Hesthaven (eds.), *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014*, pages 73–85, Cham, 2015, Springer International Publishing.

G. Fu, An explicit divergence-free DG method for incompressible flow, *Computer Methods in Applied Mechanics and Engineering* **345**, 502 – 517, 2019.

G. Fu, Arbitrary Lagrangian–Eulerian hybridizable discontinuous Galerkin methods for incompressible flow with moving boundaries and interfaces, *Computer Methods in Applied Mechanics and Engineering* **367**, 113158, 2020.

K. J. Galvin, A. Linke, L. G. Rebholz, and N. E. Wilson, Stabilizing poor mass conservation in incompressible flow problems with large irrotational forcing and application to thermal convection, *Computer Methods in Applied Mechanics and Engineering* **237-240**, 166 – 176, 2012.

P. Gao, J. Ouyang, P. Dai, and W. Zhou, A coupled continuous and discontinuous finite element method for the incompressible flows, *International Journal for Numerical Methods in Fluids* **84**, 477–493, 2017.

G. Gassner and D. A. Kopriva, A Comparison of the Dispersion and Dissipation Errors of Gauss and Gauss-Lobatto Discontinuous Galerkin Spectral Element Methods, *SIAM Journal on Scientific Computing* **33**, 2560–2579, 2011.

G. J. Gassner, A skew-symmetric discontinuous Galerkin spectral element discretization and its relation to SBP-SAT finite difference methods, *SIAM Journal on Scientific Computing* **35**, A1233–A1253, 2013.

G. J. Gassner and A. D. Beck, On the accuracy of high-order discretizations for underresolved turbulence simulations, *Theoretical and Computational Fluid Dynamics* **27**, 221–237, 2013.

G. J. Gassner, A. R. Winters, and D. A. Kopriva, Split form nodal discontinuous Galerkin schemes with summation-by-parts property for the compressible Euler equations, *Journal of Computational Physics* **327**, 39 – 66, 2016.

N. R. Gauger, A. Linke, and P. W. Schroeder, On high-order pressure-robust space discretisations, their advantages for incompressible high Reynolds number generalised Beltrami flows and beyond, *The SMAI Journal of Computational Mathematics* **5**, 89–129, 2019.

M. W. Gee, C. M. Siefert, J. J. Hu, R. S. Tuminaro, and M. G. Sala, ML 5.0 smoothed aggregation users guide, Technical report, Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

M. W. Gee, U. Küttler, and W. A. Wall, Truly monolithic algebraic multigrid for fluid–structure interaction, *International Journal for Numerical Methods in Engineering* **85**, 987–1016, 2011.

J.-F. Gerbeau and M. Vidrascu, A quasi-Newton algorithm based on a reduced model for fluid-structure interaction problems in blood flows, *ESAIM: Mathematical Modelling and Numerical Analysis* **37**, 631647, 2003.

J.-F. Gerbeau, C. Le Bris, and M. Bercovier, Spurious velocities in the steady flow of an incompressible fluid subjected to external forces, *International Journal for Numerical Methods in Fluids* **25**, 679–695, 1997.

J.-F. Gerbeau, M. Vidrascu, and P. Frey, Fluid–structure interaction in blood flows on geometries based on medical imaging, *Computers & Structures* **83**, 155 – 165, 2005.

P. Geuzaine, C. Grandmont, and C. Farhat, Design and analysis of ALE schemes with provable second-order time-accuracy for inviscid and viscous flow simulations, *Journal of Computational Physics* **191**, 206 – 227, 2003.

A. Ghidoni, A. Colombo, F. Bassi, and S. Rebay, Efficient p-multigrid discontinuous Galerkin solver for complex viscous flows on stretched grids, *International Journal for Numerical Methods in Fluids* **75**, 134–154, 2014.

A. Gholami, D. Malhotra, H. Sundar, and G. Biros, FFT, FMM, or Multigrid? A comparative Study of State-Of-the-Art Poisson Solvers for Uniform and Nonuniform Grids in the Unit Cube, *SIAM Journal on Scientific Computing* **38**, C280–C306, 2016.

J. Gibbon, The three-dimensional Euler equations: Where do we stand?, *Physica D: Nonlinear Phenomena* **237**, 1894 – 1904, 2008.

V. Girault, B. Rivière, and M. Wheeler, A discontinuous Galerkin method with nonoverlapping domain decomposition for the Stokes and Navier–Stokes problems, *Mathematics of Computation* **74**, 53–84, 2005a.

V. Girault, B. Rivière, and M. F. Wheeler, A splitting method using discontinuous Galerkin for the transient incompressible Navier–Stokes equations, *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique* **39**, 1115–1147, 2005b.

B. Gmeiner, U. Rüde, H. Stengel, C. Waluga, and B. Wohlmuth, Performance and scalability of hierarchical hybrid multigrid solvers for Stokes systems, *SIAM Journal on Scientific Computing* **37**, C143–C168, 2015a.

B. Gmeiner, U. Rüde, H. Stengel, C. Waluga, and B. Wohlmuth, Towards textbook efficiency for parallel multigrid, *Numerical Mathematics: Theory, Methods and Applications* **8**, 2246, 2015b.

K. Goda, A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows, *Journal of Computational Physics* **30**, 76–95, 1979.

J. Gopalakrishnan and G. Kanschat, A multilevel discontinuous Galerkin method, *Numerische Mathematik* **95**, 527–550, 2003.

T. Grafke, H. Homann, J. Dreher, and R. Grauer, Numerical simulations of possible finite time singularities in the incompressible Euler equations: Comparison of numerical methods, *Physica D: Nonlinear Phenomena* **237**, 1932 – 1936, 2008.

R. Grauer, C. Marliani, and K. Germaschewski, Adaptive mesh refinement for singular solutions of the incompressible Euler equations, *Physical Review Letters* **80**, 4177–4180, 1998.

V. Gravemeier, The variational multiscale method for laminar and turbulent flow, *Archives of Computational Methods in Engineering* **13**, 249, 2006.

V. Gravemeier, M. W. Gee, M. Kronbichler, and W. A. Wall, An algebraic variational multiscale–multigrid method for large eddy simulation of turbulent flow, *Computer Methods in Applied Mechanics and Engineering* **199**, 853 – 864, 2010.

V. Gravemeier, A. Comerford, L. Yoshihara, M. Ismail, and W. A. Wall, A novel formulation for Neumann inflow boundary conditions in biomechanics, *International Journal for Numerical Methods in Biomedical Engineering* **28**, 560–573, 2012.

F. F. Grinstein, L. G. Margolin, and W. J. Rider, *Implicit large eddy simulation: computing turbulent fluid dynamics*, Cambridge University Press, 2007.

W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, Performance modeling and tuning of an unstructured mesh CFD application, In *SC '00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, pages 34–34, 2000.

J. L. Guermond, P. Minev, and J. Shen, An overview of projection methods for incompressible flows, *Computer Methods in Applied Mechanics and Engineering* **195**, 6011 – 6045, 2006.

J.-L. Guermond and J. Shen, Velocity-correction projection methods for incompressible flows, *SIAM Journal on Numerical Analysis* **41**, 112–134, 2003.

J. Guermond and J. Shen, On the error estimates for the rotational pressure-correction projection methods, *Mathematics of Computation* **73**, 1719–1737, 2004.

J. Guzmán, C.-W. Shu, and F. A. Sequeira, H(div) conforming and DG methods for incompressible Euler's equations, *IMA Journal of Numerical Analysis* **37**, 1733–1771, 2016.

R. Haelterman, A. Bogaers, K. Scheufele, B. Uekermann, and M. Mehl, Improving the performance of the partitioned QN-ILS procedure for fluid–structure interaction problems: Filtering, *Computers & Structures* **171**, 9 – 17, 2016.

G. Hager and G. Wellein, *Introduction to high performance computing for scientists and engineers*, CRC Press, 2010.

R. Hartmann and P. Houston, Symmetric Interior Penalty DG Methods for the Compressible Navier–Stokes Equations I: Method Formulation, Technical Report 164, University of Nottingham, 2005.

M. Heil, An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems, *Computer Methods in Applied Mechanics and Engineering* **193**, 1 – 23, 2004.

M. Heil, A. L. Hazel, and J. Boyle, Solvers for large-displacement fluid–structure interaction problems: segregated versus monolithic approaches, *Computational Mechanics* **43**, 91–101, 2008.

A. Heinecke, A. Breuer, S. Rettenberger, M. Bader, A. Gabriel, C. Pelties, A. Bode, W. Barth, X. Liao, K. Vaidyanathan, M. Smelyanskiy, and P. Dubey, Petascale high order dynamic rupture earthquake simulations on heterogeneous supercomputers, In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 3–14, 2014.

J. Heinz. High-order discontinuous Galerkin methods on moving meshes for the incompressible Navier–Stokes equations. Master's thesis, Technical University of Munich, 2019.

T. Heister and G. Rapin, Efficient augmented Lagrangian-type preconditioning for the Oseen problem using Grad-Div stabilization, *International Journal for Numerical Methods in Fluids* **71**, 118–134, 2013.

B. Helenbrook, D. Mavriplis, and H. Atkins, Analysis of "p"-Multigrid for Continuous and Discontinuous Finite Element Discretizations, In *16th AIAA Computational Fluid Dynamics Conference*, page 3989, 2003.

B. T. Helenbrook, Mesh deformation using the biharmonic operator, *International Journal for Numerical Methods in Engineering* **56**, 1007–1021, 2003.

B. T. Helenbrook and H. Atkins, Solving discontinuous Galerkin formulations of Poisson's equation using geometric and p multigrid, *AIAA journal* **46**, 894–902, 2008.

B. T. Helenbrook and B. S. Mascarenhas, Analysis of Implicit Time-Advancing p-Multigrid schemes for Discontinuous Galerkin Discretizations of the Euler Equations, In *46th AIAA Fluid Dynamics Conference*, page 3494, 2016.

B. Helenbrook, A two-fluid spectral-element method, *Computer Methods in Applied Mechanics and Engineering* **191**, 273 – 294, 2001.

P. Hemker, W. Hoffmann, and M. van Raalte, Two-level Fourier analysis of a multigrid approach for discontinuous Galerkin discretization, *SIAM Journal on Scientific Computing* **25**, 1018–1041, 2003.

M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Volume 49, NBS Washington, DC, 1952.

J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, Springer, 2007.

J. Heys, T. Manteuffel, S. McCormick, and L. Olson, Algebraic multigrid for higher-order finite elements, *Journal of Computational Physics* **204**, 520 – 532, 2005.

S. Hickel and N. A. Adams, On implicit subgrid-scale modeling in wall-bounded flows, *Physics of Fluids* **19**, 105106, 2007.

S. Hickel, N. A. Adams, and J. A. Domaradzki, An adaptive local deconvolution method for implicit LES, *Journal of Computational Physics* **213**, 413 – 436, 2006.

K. Hillewaert, *Development of the discontinuous Galerkin method for high-resolution, large scale CFD and acoustics in industrial geometries*, PhD thesis, Univ. de Louvain, 2013.

K. Hillewaert, N. Chevaugeon, P. Geuzaine, and J.-F. Remacle, Hierarchic multigrid iteration strategy for the discontinuous Galerkin solution of the steady Euler equations, *International Journal for Numerical Methods in Fluids* **51**, 1157–1176, 2006.

F. Hindenlang, G. Gassner, C. Altmann, A. Beck, M. Staudenmaier, and C.-D. Munz, Explicit discontinuous Galerkin methods for unsteady problems, *Computers & Fluids* **61**, 86–93, 2012.

C. Hirt and J. Cook, Calculating three-dimensional flows around structures and over rough terrain, *Journal of Computational Physics* **10**, 324–340, 1972.

C. Hirt, A. Amsden, and J. Cook, An arbitrary Lagrangian-Eulerian computing method for all flow speeds, *Journal of Computational Physics* **14**, 227 – 253, 1974.

L.-W. Ho and A. T. Patera, A Legendre spectral element method for simulation of unsteady incompressible viscous free-surface flows, *Computer Methods in Applied Mechanics and Engineering* **80**, 355 – 366, 1990.

A. G. Holzapfel, *Nonlinear solid mechanics*, John Wiley & Sons, Inc., 2000.

T. L. Horváth and S. Rhebergen, A locally conservative and energy-stable finite-element method for the Navier–Stokes problem on time-dependent domains, *International Journal for Numerical Methods in Fluids* **89**, 519–532, 2019.

T. Y. Hou and R. Li, Blowup or no blowup? The interplay between theory and numerics, *Physica D: Nonlinear Phenomena* **237**, 1937 – 1944, 2008.

T. Houba, A. Dasgupta, S. Gopalakrishnan, R. Gosse, and S. Roy, Supersonic turbulent flow simulation using a scalable parallel modal discontinuous Galerkin numerical method, *Scientific reports* **9**, 1–19, 2019.

S. Hoyas and J. Jiménez, Reynolds number effects on the Reynolds-stress budgets in turbulent channels, *Physics of Fluids* **20**, 101511, 2008.

J. Hron and S. Turek, A monolithic FEM/multigrid solver for an ALE formulation of fluid–structure interaction with applications in biomechanics, In H.-J. Bungartz and M. Schäfer (eds.), *Fluid-Structure Interaction*, pages 146–170, Berlin, Heidelberg, 2006, Springer Berlin Heidelberg.

A. Huerta, A. Angeloski, X. Roca, and J. Peraire, Efficiency of high-order elements for continuous and discontinuous Galerkin methods, *International Journal for Numerical Methods in Engineering* **96**, 529–560, 2013.

T. J. Hughes, W. K. Liu, and T. K. Zimmermann, Lagrangian-Eulerian finite element formulation for incompressible viscous flows, *Computer Methods in Applied Mechanics and Engineering* **29**, 329 – 349, 1981.

I. Huismann, J. Stiller, and J. Fröhlich, Factorizing the factorization – a spectral-element solver for elliptic equations with linear operation count, *Journal of Computational Physics* **346**, 437 – 448, 2017.

I. Huismann, J. Stiller, and J. Fröhlich, Scaling to the stars – a linearly scaling elliptic solver for p-multigrid, *Journal of Computational Physics* **398**, 108868, 2019.

H. Ibeid, L. Olson, and W. Gropp, FFT, FMM, and multigrid on the road to exascale: Performance challenges and opportunities, *Journal of Parallel and Distributed Computing* **136**, 63 – 74, 2020.

T. Ichimura, K. Fujita, P. E. B. Quinay, L. Maddegedara, M. Hori, S. Tanaka, Y. Shizawa, H. Kobayashi, and K. Minami, Implicit nonlinear wave simulation with 1.08T DOF and 0.270T unstructured finite elements to enhance comprehensive earthquake simulation, In *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2015.

P. Isett, Nonuniqueness and existence of continuous, globally dissipative Euler flows, *arXiv preprint arXiv:1710.11186*, 2017.

P. Isett, A proof of Onsager's conjecture, *Annals of Mathematics* **188**, 871–963, 2018.

B. Janssen and G. Kanschat, Adaptive multilevel methods with local smoothing for H^1- and H^{curl}-conforming high order finite element methods, *SIAM Journal on Scientific Computing* **33**, 2095–2114, 2011.

J. Jiménez, Computing high-Reynolds-number turbulence: will simulations ever replace experiments?, *Journal of Turbulence* **4**, N22, 2003.

D. Jodlbauer, U. Langer, and T. Wick, Parallel block-preconditioned monolithic solvers for fluid–structure interaction problems, *International Journal for Numerical Methods in Engineering* **117**, 623–643, 2019.

V. John, Reference values for drag and lift of a two-dimensional time-dependent flow around a cylinder, *International Journal for Numerical Methods in Fluids* **44**, 777–788, 2004.

V. John, A. Linke, C. Merdon, M. Neilan, and L. G. Rebholz, On the divergence constraint in mixed finite element methods for incompressible flows, *SIAM Review* **59**, 492–544, 2017.

A. Johnson and T. Tezduyar, Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces, *Computer Methods in Applied Mechanics and Engineering* **119**, 73 – 94, 1994.

S. M. Joshi, P. J. Diamessis, D. T. Steinmoeller, M. Stastna, and G. N. Thomsen, A post-processing technique for stabilizing the discontinuous pressure projection operator in marginally-resolved incompressible inviscid flow, *Computers & Fluids* **139**, 120 – 129, 2016.

C. Josserand, Y. Pomeau, and S. Rica, Finite-time localized singularities as a mechanism for turbulent dissipation, *Physical Review Fluids* **5**, 054607, 2020.

H. Kanchi and A. Masud, A 3D adaptive mesh moving scheme, *International Journal for Numerical Methods in Fluids* **54**, 923–944, 2007.

Y. Kaneda, T. Ishihara, M. Yokokawa, K. Itakura, and A. Uno, Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box, *Physics of Fluids* **15**, L21–L24, 2003.

G. Kanschat, Multi-level methods for discontinuous Galerkin FEM on locally refined meshes, *Computers & Structures* **82**, 2437–2445, 2004.

G. Kanschat, Robust smoothers for high order discontinuous Galerkin discretizations of advection-diffusion problems, *Journal of Computational and Applied Mathematics* **218**, 53–60, 2008.

A. Karakus, N. Chalmers, K. Świrydowicz, and T. Warburton, A GPU accelerated discontinuous Galerkin incompressible flow solver, *Journal of Computational Physics* **390**, 380–404, 2019.

G. S. Karamanos and G. E. Karniadakis, A spectral vanishing viscosity method for large-eddy simulations, *Journal of Computational Physics* **163**, 22–50, 2000.

G. E. Karniadakis and S. J. Sherwin, *Spectral/hp element methods for computational fluid dynamics*, Oxford University Press, 2013.

G. E. Karniadakis, M. Israeli, and S. A. Orszag, High-order splitting methods for the incompressible Navier–Stokes equations, *Journal of Computational Physics* **97**, 414 – 443, 1991.

N. Kasagi and A. Matsunaga, Three-dimensional particle-tracking velocimetry measurement of turbulence statistics and energy budget in a backward-facing step flow, *International Journal of Heat and Fluid Flow* **16**, 477 – 485, 1995.

D. Kay, D. Loghin, and A. Wathen, A preconditioner for the steady-state Navier–Stokes equations, *SIAM Journal on Scientific Computing* **24**, 237–256, 2002.

D. Kempf, R. Heß, S. Müthing, and P. Bastian, Automatic code generation for high-performance discontinuous Galerkin methods on modern architectures, *ACM Transactions on Mathematical Software* **47**, 2020.

R. M. Kerr, Evidence for a singularity of the three-dimensional, incompressible Euler equations, *Physics of Fluids A: Fluid Dynamics* **5**, 1725–1746, 1993.

R. M. Kerr, Bounds for Euler from vorticity moments and line divergence, *Journal of Fluid Mechanics* **729**, R2, 2013.

R. M. Kerr and F. Hussain, Simulation of vortex reconnection, *Physica D: Nonlinear Phenomena* **37**, 474 – 484, 1989.

S. Kida, Three-dimensional periodic flows with high-symmetry, *Journal of the Physical Society of Japan* **54**, 2132–2136, 1985.

R. M. Kirby and G. E. Karniadakis, De-aliasing on non-uniform grids: algorithms and applications, *Journal of Computational Physics* **191**, 249 – 264, 2003.

R. M. Kirby and G. E. Karniadakis, Coarse resolution turbulence simulations with spectral vanishing viscosity–large-eddy simulations (SVV-LES), *Journal of Fluids Engineering* **124**, 886–891, 2002.

R. M. Kirby and S. J. Sherwin, Stabilisation of spectral/hp element methods through spectral vanishing viscosity: Application to fluid mechanics modelling, *Computer Methods in Applied Mechanics and Engineering* **195**, 3128 – 3144, 2006.

R. M. Kirby, S. J. Sherwin, and B. Cockburn, To CG or to HDG: A comparative study, *Journal of Scientific Computing* **51**, 183–212, 2012.

B. Klein, F. Kummer, M. Keil, and M. Oberlack, An extension of the SIMPLE based discontinuous Galerkin solver to unsteady incompressible flows, *International Journal for Numerical Methods in Fluids* **77**, 571–589, 2015.

B. Klein, F. Kummer, and M. Oberlack, A SIMPLE based discontinuous Galerkin solver for steady incompressible flows, *Journal of Computational Physics* **237**, 235 – 250, 2013.

A. Klöckner, T. Warburton, J. Bridge, and J. Hesthaven, Nodal discontinuous Galerkin methods on graphics processors, *Journal of Computational Physics* **228**, 7863 – 7882, 2009.

A. N. Kolmogorov, The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers, *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* **434**, 9–13, 1991.

F. Kong and X.-C. Cai, Scalability study of an implicit solver for coupled fluid–structure interaction problems on unstructured meshes in 3d, *The International Journal of High Performance Computing Applications* **32**, 207–219, 2018.

D. A. Kopriva, *Implementing spectral methods for partial differential equations: algorithms for scientists and engineers*, Springer, 2009.

N. Krais, A. Beck, T. Bolemann, H. Frank, D. Flad, G. Gassner, F. Hindenlang, M. Hoffmann, T. Kuhn, M. Sonntag, and C.-D. Munz, FLEXI: A high order discontinuous Galerkin framework for hyperbolic–parabolic conservation laws, *Computers & Mathematics with Applications*, 2020a.

N. Krais, G. Schnücke, T. Bolemann, and G. J. Gassner, Split form ALE discontinuous Galerkin methods with applications to under-resolved turbulent low-Mach number flows, *Journal of Computational Physics* **421**, 109726, 2020b.

B. Krank, *Wall modeling via function enrichment for computational fluid dynamics*, PhD thesis, Technische Universität München, 2019.

B. Krank, N. Fehn, W. A. Wall, and M. Kronbichler, A high-order semi-explicit discontinuous Galerkin solver for 3D incompressible flow with application to DNS and LES of turbulent channel flow, *Journal of Computational Physics* **348**, 634–659, 2017.

B. Krank, M. Kronbichler, and W. A. Wall, Direct numerical simulation of flow over periodic hills up to $\mathrm{Re}_H = 10,595$, *Flow, turbulence and combustion* **101**, 521–551, 2018a.

B. Krank, M. Kronbichler, and W. A. Wall, Wall modeling via function enrichment within a high-order DG method for RANS simulations of incompressible flow, *International Journal for Numerical Methods in Fluids* **86**, 107–129, 2018b.

B. Krank, M. Kronbichler, and W. A. Wall, Wall modeling via function enrichment: Extension to detached-eddy simulation, *Computers & Fluids* **179**, 718 – 725, 2019a.

B. Krank, M. Kronbichler, and W. A. Wall, A multiscale approach to hybrid RANS/LES wall modeling within a high-order discontinuous Galerkin scheme using function enrichment, *International Journal for Numerical Methods in Fluids* **90**, 81–113, 2019b.

M. Kronbichler and K. Kormann, A generic interface for parallel cell-based finite element operator application, *Computers & Fluids* **63**, 135–147, 2012.

M. Kronbichler and K. Kormann, Fast matrix-free evaluation of discontinuous Galerkin finite element operators, *ACM Transactions on Mathematical Software* **45**, 29:1–29:40, 2019.

M. Kronbichler and W. A. Wall, A Performance Comparison of Continuous and Discontinuous Galerkin Methods with Fast Multigrid Solvers, *SIAM Journal on Scientific Computing* **40**, A3423–A3448, 2018.

M. Kronbichler, S. Schoeder, C. Müller, and W. A. Wall, Comparison of implicit and explicit hybridizable discontinuous Galerkin methods for the acoustic wave equation, *International Journal for Numerical Methods in Engineering* **106**, 712–739, 2016.

M. Kronbichler, A. Diagne, and H. Holmgren, A fast massively parallel two-phase flow solver for microfluidic chip simulation, *The International Journal of High Performance Computing Applications* **32**, 266–287, 2018a.

M. Kronbichler, K. Kormann, N. Fehn, P. Munch, and J. Witte, A Hermite-like basis for faster matrix-free evaluation of interior penalty discontinuous Galerkin operators, *arXiv preprint arXiv:1907.08492*, 2019.

M. Kronbichler, The Discontinuous Galerkin Method: Derivation and Properties, In M. Kronbichler and P.-O. Persson (eds.), *Efficient High-Order Discretizations for Computational Fluid Dynamics*, pages 1–55, Cham, 2021a, Springer International Publishing.

M. Kronbichler, High-Performance Implementation of Discontinuous Galerkin Methods with Application in Fluid Flow, In M. Kronbichler and P.-O. Persson (eds.), *Efficient High-Order Discretizations for Computational Fluid Dynamics*, pages 57–115, Cham, 2021b, Springer International Publishing.

M. Kronbichler and M. Allalen, Efficient high-order discontinuous Galerkin finite elements with matrix-free implementations, In H.-J. Bungartz, D. Kranzlmüller, V. Weinberg, J. Weismüller, and V. Wohlgemuth (eds.), *Advances and New Trends in Environmental Informatics*, pages 89–110, Cham, 2018, Springer International Publishing.

M. Kronbichler and K. Ljungkvist, Multigrid for matrix-free high-order finite element computations on graphics processors, *ACM Trans. Parallel Comput.* **6**, 2019.

M. Kronbichler and P.-O. Persson (eds.), *Efficient High-Order Discretizations for Computational Fluid Dynamics*, Volume 602 of *CISM International Centre for Mechanical Sciences book series*, Springer International Publishing, 2021.

M. Kronbichler, T. Heister, and W. Bangerth, High accuracy mantle convection simulation through modern numerical methods, *Geophysical Journal International* **191**, 12–29, 2012.

M. Kronbichler, K. Kormann, I. Pasichnyk, and M. Allalen, Fast matrix-free discontinuous Galerkin kernels on modern computer architectures, In J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes (eds.), *High Performance Computing*, pages 237–255, Cham, 2017, Springer International Publishing.

M. Kronbichler, B. Krank, N. Fehn, S. Legat, and W. A. Wall, A new high-order discontinuous Galerkin solver for DNS and LES of turbulent incompressible flow, In *New Results in Numerical and Experimental Fluid Mechanics XI*, pages 467–477, Springer, 2018b.

U. Küttler and W. A. Wall, Fixed-point fluid–structure interaction solvers with dynamic relaxation, *Computational Mechanics* **43**, 61–72, 2008.

U. Küttler and W. A. Wall, Vector extrapolation for strong coupling fluid–structure interaction solvers, *Journal of Applied Mechanics* **76**, 021205–1–021205–7, 2009.

U. Küttler, M. Gee, C. Förster, A. Comerford, and W. A. Wall, Coupling strategies for biomedical fluid–structure interaction problems, *International Journal for Numerical Methods in Biomedical Engineering* **26**, 305–321, 2010.

D. Kuzzay, E.-W. Saw, F. J. Martins, D. Faranda, J.-M. Foucaut, F. Daviaud, and B. Dubrulle, New method for detecting singularities in experimental incompressible flows, *Nonlinearity* **30**, 2381, 2017.

A. La Spina, C. Förster, M. Kronbichler, and W. A. Wall, On the role of (weak) compressibility for fluid–structure interaction solvers, *International Journal for Numerical Methods in Fluids* **92**, 129–147, 2020a.

A. La Spina, M. Kronbichler, M. Giacomini, W. A. Wall, and A. Huerta, A weakly compressible hybridizable discontinuous Galerkin formulation for fluid–structure interaction problems, *Computer Methods in Applied Mechanics and Engineering* **372**, 113392, 2020b.

E. Lamballais, T. Dairay, S. Laizet, and J. C. Vassilicos, Implicit/explicit spectral viscosity and large-scale SGS effects, In M. V. Salvetti, V. Armenio, J. Fröhlich, B. J. Geurts, and H. Kuerten (eds.), *Direct and Large-Eddy Simulation XI*, pages 107–113, Cham, 2019, Springer International Publishing.

U. Langer and H. Yang, Robust and efficient monolithic fluid-structure-interaction solvers, *International Journal for Numerical Methods in Engineering* **108**, 303–325, 2016.

A. Larios, M. R. Petersen, E. S. Titi, and B. Wingate, A computational investigation of the finite-time blow-up of the 3D incompressible Euler equations based on the Voigt regularization, *Theoretical and Computational Fluid Dynamics* **32**, 23–34, 2018.

C. Lasser and A. Toselli, Overlapping preconditioners for discontinuous Galerkin approximations of second order problems, In *Thirteenth International Conference on Domain Decomposition Methods, N. Debit, M. Garbey, R. Hoppe, J. Périaux, D. Keyes, and Y. Kuznetsov, eds*, 2001.

P. Le Tallec and J. Mouro, Fluid structure interaction with large structural displacements, *Computer Methods in Applied Mechanics and Engineering* **190**, 3039 – 3067, 2001.

P. L. Lederer, A. Linke, C. Merdon, and J. Schöberl, Divergence-free reconstruction operators for pressure-robust Stokes discretizations with continuous pressure finite elements, *SIAM Journal on Numerical Analysis* **55**, 1291–1314, 2017.

C. Lehrenfeld and J. Schöberl, High order exactly divergence-free hybrid discontinuous Galerkin methods for unsteady incompressible flows, *Computer Methods in Applied Mechanics and Engineering* **307**, 339 – 361, 2016.

C. Lehrenfeld, G. Lube, and P. W. Schroeder, A natural decomposition of viscous dissipation in DG methods for turbulent incompressible flows, *arXiv preprint arXiv:1811.12769*, 2018.

E. Leriche and G. Labrosse, High-order direct Stokes solvers with or without temporal splitting: numerical investigations of their comparative properties, *SIAM Journal on Scientific Computing* **22**, 1386–1410, 2000.

E. Leriche, E. Perchat, G. Labrosse, and M. O. Deville, Numerical evaluation of the accuracy and stability properties of high-order direct Stokes solvers with or without temporal splitting, *Journal of Scientific Computing* **26**, 25–43, 2006.

M. Lesoinne and C. Farhat, Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations, *Computer Methods in Applied Mechanics and Engineering* **134**, 71 – 90, 1996.

F. Lindner, M. Mehl, K. Scheufele, and B. Uekermann, A comparison of various quasi-Newton schemes for partitioned fluid–structure interaction, In *Proceedings of 6th International Conference on Computational Methods for Coupled Problems in Science and Engineering, Venice*, pages 1–12, 2015.

F. Lindner, A. Totounferoush, M. Mehl, B. Uekermann, N. E. Pour, V. Krupp, S. Roller, T. Reimann, D. C. Sternel, R. Egawa, H. Takizawa, and F. Simonis, ExaFSA: Parallel fluid-structure-acoustic simulation, In H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, and W. E. Nagel (eds.), *Software for Exascale Computing - SPPEXA 2016-2019*, pages 271–300, Cham, 2020, Springer International Publishing.

A. Linke and C. Merdon, Pressure-robustness and discrete Helmholtz projectors in mixed finite element methods for the incompressible Navier–Stokes equations, *Computer Methods in Applied Mechanics and Engineering* **311**, 304 – 326, 2016a.

A. Linke and C. Merdon, On velocity errors due to irrotational forces in the Navier–Stokes momentum balance, *Journal of Computational Physics* **313**, 654 – 661, 2016b.

A. Linke, On the role of the Helmholtz decomposition in mixed methods for incompressible flows and a new variational crime, *Computer Methods in Applied Mechanics and Engineering* **268**, 782 – 800, 2014.

A. Linke, G. Matthies, and L. Tobiska, Robust arbitrary order mixed finite element methods for the incompressible Stokes equations with pressure independent velocity errors, *ESAIM: Mathematical Modelling and Numerical Analysis* **50**, 289–309, 2016.

P.-L. Lions, *Mathematical Topics in Fluid Mechanics. Volume 1: Incompressible Models*, Volume 3 of Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, 1996.

K. Ljungkvist, Matrix-free finite-element operator application on graphics processing units, In L. Lopes, J. Žilinskas, A. Costan, R. G. Cascella, G. Kecskemeti, E. Jeannot, M. Cannataro, L. Ricci, S. Benkner, S. Petit, V. Scarano, J. Gracia, S. Hunold, S. L. Scott, S. Lankes, C. Lengauer, J. Carretero, J. Breitbart, and M. Alexander (eds.), *Euro-Par 2014: Parallel Processing Workshops*, pages 450–461, Cham, 2014, Springer International Publishing.

K. Ljungkvist, Matrix-free finite-element computations on graphics processors with adaptively refined unstructured meshes, In *SpringSim (HPC)*, pages 1–1, 2017.

R. Löhner, Multistage explicit advective prediction for projection-type incompressible flow solvers, *Journal of Computational Physics* **195**, 143 – 152, 2004.

R. Löhner, *Applied computational fluid dynamics techniques: an introduction based on finite element methods*, John Wiley & Sons, 2008.

R. Löhner, Error and work estimates for high-order elements, *International Journal for Numerical Methods in Fluids* **67**, 2184–2188, 2011.

R. Löhner, Improved error and work estimates for high-order elements, *International Journal for Numerical Methods in Fluids* **72**, 1207–1218, 2013.

R. Löhner, Towards overcoming the LES crisis, *International Journal of Computational Fluid Dynamics* **33**, 87–97, 2019.

R. Löhner and C. Yang, Improved ALE mesh velocities for moving bodies, *Communications in Numerical Methods in Engineering* **12**, 599–608, 1996.

R. Löhner, C. Othmer, M. Mrosek, A. Figueroa, and A. Degro, Overnight industrial LES for external aerodynamics, In *AIAA Scitech 2020 Forum*, 2020.

I. Lomtev, R. Kirby, and G. Karniadakis, A discontinuous Galerkin ALE method for compressible viscous flows in moving domains, *Journal of Computational Physics* **155**, 128 – 159, 1999.

I. Lomtev and G. E. Karniadakis, A discontinuous Galerkin method for the Navier–Stokes equations, *International Journal for Numerical Methods in Fluids* **29**, 587–603, 1999.

N. Loppi, F. Witherden, A. Jameson, and P. Vincent, A high-order cross-platform incompressible Navier–Stokes solver via artificial compressibility with application to a turbulent jet, *Computer Physics Communications* **233**, 193 – 205, 2018.

J. Lorentzon and J. Revstedt, A numerical study of partitioned fluid-structure interaction applied to a cantilever in incompressible turbulent flow, *International Journal for Numerical Methods in Engineering* **121**, 806–827, 2020.

J. W. Lottes and P. F. Fischer, Hybrid Multigrid/Schwarz Algorithms for the Spectral Element Method, *Journal of Scientific Computing* **24**, 45–78, 2005.

C. Lu, X. Jiao, and N. Missirlis, A hybrid geometric+algebraic multigrid method with semi-iterative smoothers, *Numerical Linear Algebra with Applications* **21**, 221–238, 2014.

G. Luo and T. Y. Hou, Potentially singular solutions of the 3D axisymmetric Euler equations, *Proceedings of the National Academy of Sciences* **111**, 12968–12973, 2014.

H. Luo, J. D. Baum, and R. Löhner, A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids, *Journal of Computational Physics* **211**, 767 – 783, 2006.

H. Luo, H. Segawa, and M. R. Visbal, An implicit discontinuous Galerkin method for the unsteady compressible Navier–Stokes equations, *Computers & Fluids* **53**, 133 – 144, 2012.

R. E. Lynch, J. R. Rice, and D. H. Thomas, Direct solution of partial difference equations by tensor product methods, *Numerische Mathematik* **6**, 185–199, 1964.

Y. Maday, A. T. Patera, and E. M. Rønquist, An operator-integration-factor splitting method for time-dependent problems: Application to incompressible fluid flow, *Journal of Scientific Computing* **5**, 263–292, 1990.

Y. Maday and R. Munoz, Spectral element multigrid. II. Theoretical justification, *Journal of Scientific Computing* **3**, 323–353, 1988.

R. A. Malinauskas, P. Hariharan, S. W. Day, L. H. Herbertson, M. Buesen, U. Steinseifer, K. I. Aycock, B. C. Good, S. Deutsch, K. B. Manning, et al., FDA benchmark medical device flow models for CFD validation, *ASAIO Journal* **63**, 150–160, 2017.

J. Manzanero, E. Ferrer, G. Rubio, and E. Valero, Design of a Smagorinsky spectral vanishing viscosity turbulence model for discontinuous Galerkin methods, *Computers & Fluids* **200**, 104440, 2020.

M. Marek, A. Tyliszczak, and A. Bogusławski, Large eddy simulation of incompressible free round jet with discontinuous Galerkin method, *International Journal for Numerical Methods in Fluids* **79**, 164–182, 2015.

L. G. Margolin, W. J. Rider, and F. F. Grinstein, Modeling turbulent flow with implicit LES, *Journal of Turbulence* **7**, N15, 2006.

L. G. Margolin and W. J. Rider, A rationale for implicit turbulence modelling, *International Journal for Numerical Methods in Fluids* **39**, 821–841, 2002.

B. S. Mascarenhas, B. T. Helenbrook, and H. L. Atkins, Application of p-multigrid to discontinuous Galerkin formulations of the Euler equations, *AIAA journal* **47**, 1200–1208, 2009.

B. S. Mascarenhas, B. T. Helenbrook, and H. L. Atkins, Coupling p-multigrid to geometric multigrid for discontinuous Galerkin formulations of the convection–diffusion equation, *Journal of Computational Physics* **229**, 3664 – 3674, 2010.

H. G. Matthies and J. Steindorf, Partitioned but strongly coupled iteration schemes for nonlinear fluid–structure interaction, *Computers & Structures* **80**, 1991 – 1999, 2002.

D. J. Mavriplis and C. R. Nastase, On the geometric conservation law for high-order discontinuous Galerkin discretizations on dynamically deforming meshes, *Journal of Computational Physics* **230**, 4285 – 4300, 2011.

D. A. May, J. Brown, and L. Le Pourhiet, pTatin3D: High-performance methods for long-term lithospheric dynamics, In J. M. Kunkel, T. Ludwig, and H. W. Meuer (eds.), *Supercomputing (SC14)*, pages 1–11, New Orleans, 2014.

M. Mayr, T. Klöppel, W. A. Wall, and M. W. Gee, A temporal consistent monolithic approach to fluid-structure interaction enabling single field predictors, *SIAM Journal on Scientific Computing* **37**, B30–B59, 2015.

M. Mayr, M. H. Noll, and M. W. Gee, A hybrid interface preconditioner for monolithic fluid–structure interaction solvers, *Advanced Modeling and Simulation in Engineering Sciences* **7**, 1–33, 2020.

R. McKeown, R. Ostilla-Mónico, A. Pumir, M. P. Brenner, and S. M. Rubinstein, Cascade leading to the emergence of small structures in vortex ring collisions, *Physical Review Fluids* **3**, 124702, 2018.

M. Mehl, B. Uekermann, H. Bijl, D. Blom, B. Gatzhammer, and A. van Zuijlen, Parallel coupling numerics for partitioned fluid–structure interaction simulations, *Computers & Mathematics with Applications* **71**, 869 – 891, 2016.

G. Mengaldo, D. D. Grazia, D. Moxey, P. E. Vincent, and S. J. Sherwin, Dealiasing techniques for high-order spectral element methods on regular and irregular grids, *Journal of Computational Physics* **299**, 56 – 81, 2015.

P. Minev and P. Gresho, A remark on pressure correction schemes for transient viscous incompressible flow, *Communications in Numerical Methods in Engineering* **14**, 335–346, 1998.

H. K. Moffatt, Singularities in fluid mechanics, *Physical Review Fluids* **4**, 110502, 2019.

D. Mok and W. A. Wall, Partitioned analysis schemes for the transient interaction of incompressible flows and nonlinear flexible structures, In W. A. Wall, K.-U. Bletzinger, and K. Schweitzerhof (eds.), *Trends in Computational Structural Mechanics*, CIMNE, 2001.

A. Montlaur, S. Fernandez-Mendez, and A. Huerta, Discontinuous Galerkin methods for the Stokes equations using divergence-free approximations, *International Journal for Numerical Methods in Fluids* **57**, 1071–1092, 2008.

G. E. Moore, Cramming more components onto integrated circuits, *Electronics* **38**, 114–117, 1965.

R. H. Morf, S. A. Orszag, and U. Frisch, Spontaneous singularity in three-dimensional inviscid, incompressible flow, *Physical Review Letters* **44**, 572–575, 1980.

R. D. Moser, J. Kim, and N. N. Mansour, Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$, *Physics of Fluids* **11**, 943–945, 1999.

R. C. Moura, J. Peiró, and S. J. Sherwin, Implicit LES approaches via discontinuous Galerkin methods at very large Reynolds, In M. V. Salvetti, V. Armenio, J. Fröhlich, B. J. Geurts, and H. Kuerten (eds.), *Direct and Large-Eddy Simulation XI*, pages 53–59, Cham, 2019, Springer International Publishing.

R. Moura, S. Sherwin, and J. Peiró, Linear dispersion–diffusion analysis and its application to under-resolved turbulence simulations using discontinuous Galerkin spectral/hp methods, *Journal of Computational Physics* **298**, 695 – 710, 2015.

R. Moura, G. Mengaldo, J. Peiró, and S. Sherwin, On the eddy-resolving capability of high-order discontinuous Galerkin approaches to implicit LES / under-resolved DNS of Euler turbulence, *Journal of Computational Physics* **330**, 615 – 623, 2017a.

R. Moura, G. Mengaldo, J. Peiró, and S. Sherwin, An LES setting for DG-based implicit LES with insights on dissipation and robustness, In *Spectral and High Order Methods for Partial Differential Equations*. Springer, 2017b.

D. Moxey, R. Amici, and M. Kirby, Efficient matrix-free high-order finite element evaluation for simplicial elements, *SIAM Journal on Scientific Computing* **42**, C97–C123, 2020a.

D. Moxey, C. D. Cantwell, Y. Bao, A. Cassinelli, G. Castiglioni, S. Chun, E. Juda, E. Kazemi, K. Lackhove, J. Marcon, G. Mengaldo, D. Serson, M. Turner, H. Xu, J. Peiró, R. M. Kirby, and S. J. Sherwin, Nektar++: Enhancing the capability and application of high-fidelity spectral/hp element methods, *Computer Physics Communications* **249**, 107110, 2020b.

R. L. Muddle, M. Mihajlović, and M. Heil, An efficient preconditioner for monolithically-coupled large-displacement fluid–structure interaction problems with pseudo-solid mesh updates, *Journal of Computational Physics* **231**, 7315 – 7334, 2012.

A. Müller, M. A. Kopera, S. Marras, L. C. Wilcox, T. Isaac, and F. X. Giraldo, Strong scaling for numerical weather prediction at petascale with the atmospheric model NUMA, *The International Journal of High Performance Computing Applications* **33**, 411–426, 2019.

P. Münch. An efficient hybrid multigrid solver for high-order discontinuous Galerkin methods. Master's thesis, Technical University of Munich, 2018.

S. D. Murugan, U. Frisch, S. Nazarenko, N. Besse, and S. S. Ray, Suppressing thermalization and constructing weak solutions in truncated inviscid equations of hydrodynamics: Lessons from the Burgers equation, *Physical Review Research* **2**, 033202, 2020.

S. Müthing, M. Piatkowski, and P. Bastian, High-performance Implementation of Matrix-free High-order Discontinuous Galerkin Methods, *arXiv preprint arXiv:1711.10885*, 2017.

C. Nastase and D. Mavriplis, Discontinuous Galerkin methods using an hp-multigrid solver for inviscid compressible flows on three-dimensional unstructured meshes, In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 107, 2006a.

C. R. Nastase and D. J. Mavriplis, High-order discontinuous Galerkin methods using an hp-multigrid approach, *Journal of Computational Physics* **213**, 330 – 357, 2006b.

O. Neumann. Analyse des spektralen Verhaltens von diskontinuierlichen Galerkin-Verfahren für turbulente Strömungen. Bachelor's thesis, Technical University of Munich, 2018.

M. Neunteufel and J. Schöberl, Fluid–structure interaction with $H(\mathrm{div})$-conforming finite elements, *Computers & Structures* **243**, 106402, 2021.

N. C. Nguyen, J. Peraire, and B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier–Stokes equations, *Journal of Computational Physics* **230**, 1147–1170, 2011.

V.-T. Nguyen, An arbitrary Lagrangian–Eulerian discontinuous Galerkin method for simulations of flows over variable geometries, *Journal of Fluids and Structures* **26**, 312 – 329, 2010.

J. Nitzler, J. Biehler, N. Fehn, P.-S. Koutsourelakis, and W. A. Wall, A generalized probabilistic learning approach for multi-fidelity uncertainty propagation in complex physical simulations, *arXiv preprint arXiv:2001.02892*, 2020.

F. Nobile and C. Vergara, An effective fluid–structure interaction formulation for vascular dynamics by generalized Robin conditions, *SIAM Journal on Scientific Computing* **30**, 731–763, 2008.

N. Offermans, O. Marin, M. Schanen, J. Gong, P. Fischer, P. Schlatter, A. Obabko, A. Peplinski, M. Hutchinson, and E. Merzari, On the strong scaling of the spectral element solver Nek5000 on petascale systems, In *Proceedings of the Exascale Applications and Software Conference 2016*, EASC '16, pages 5:1–5:10, New York, NY, USA, 2016, ACM.

N. Offermans, A. Peplinski, O. Marin, P. F. Fischer, and P. Schlatter, Towards adaptive mesh refinement for the spectral element solver Nek5000, In M. V. Salvetti, V. Armenio, J. Fröhlich, B. J. Geurts, and H. Kuerten (eds.), *Direct and Large-Eddy Simulation XI*, pages 9–15, Cham, 2019, Springer International Publishing.

M. Olshanskii, G. Lube, T. Heister, and J. Löwe, Grad–div stabilization and subgrid pressure models for the incompressible Navier–Stokes equations, *Computer Methods in Applied Mechanics and Engineering* **198**, 3975 – 3988, 2009.

L. N. Olson and J. B. Schroder, Smoothed aggregation multigrid solvers for high-order discontinuous Galerkin methods for elliptic problems, *Journal of Computational Physics* **230**, 6959 – 6976, 2011.

B. O'Malley, J. Kópházi, R. Smedley-Stevenson, and M. Eaton, P-multigrid expansion of hybrid multilevel solvers for discontinuous Galerkin finite element discrete ordinate (DG-FEM-SN) diffusion synthetic acceleration (DSA) of radiation transport algorithms, *Progress in Nuclear Energy* **98**, 177 – 186, 2017.

L. Onsager, Statistical hydrodynamics, *Il Nuovo Cimento (1943-1954)* **6**, 279–287, 1949.

P. Orlandi, Energy spectra power laws and structures, *Journal of Fluid Mechanics* **623**, 353374, 2009.

P. Orlandi, S. Pirozzoli, and G. F. Carnevale, Vortex events in Euler and Navier–Stokes simulations with smooth initial conditions, *Journal of Fluid Mechanics* **690**, 288320, 2012.

S. A. Orszag, Spectral methods for problems in complex geometries, *Journal of Computational Physics* **37**, 70–92, 1980.

S. A. Orszag, M. Israeli, and M. O. Deville, Boundary conditions for incompressible flows, *Journal of Scientific Computing* **1**, 75–111, 1986.

E. Otero, J. Gong, M. Min, P. Fischer, P. Schlatter, and E. Laure, OpenACC acceleration for the $P_N - P_{N-2}$ algorithm in Nek5000, *Journal of Parallel and Distributed Computing* **132**, 69 – 78, 2019.

W. Pazner, Efficient low-order refined preconditioners for high-order matrix-free continuous and discontinuous Galerkin methods, *SIAM Journal on Scientific Computing* **42**, A3055–A3083, 2020.

W. Pazner and P.-O. Persson, Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods, *Journal of Computational Physics* **354**, 344 – 369, 2018.

B. R. Pearson, P.-Å. Krogstad, and W. van de Water, Measurements of the turbulent energy dissipation rate, *Physics of Fluids* **14**, 1288–1290, 2002.

D. Pelletier, A. Fortin, and R. Camarero, Are FEM solutions of incompressible flows really incompressible? (or how simple flows can cause headaches!), *International Journal for Numerical Methods in Fluids* **9**, 99–112, 1989.

R. B. Pelz and Y. Gulak, Evidence for a real-time singularity in hydrodynamics from time series analysis, *Physical Review Letters* **79**, 4998–5001, 1997.

R. B. Pelz, Symmetry and the hydrodynamic blow-up problem, *Journal of Fluid Mechanics* **444**, 299320, 2001.

P. Persson and J. Peraire, Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier–Stokes Equations, *SIAM Journal on Scientific Computing* **30**, 2709–2733, 2008.

P.-O. Persson, J. Peraire, and J. Bonet, A high order discontinuous Galerkin method for fluid-structure interaction, In *18th AIAA Computational Fluid Dynamics Conference*, page 4327, 2007.

P.-O. Persson, J. Bonet, and J. Peraire, Discontinuous Galerkin solution of the Navier-Stokes equations on deformable domains, *Computer Methods in Applied Mechanics and Engineering* **198**, 1585–1595, 2009.

M. Piatkowski and P. Bastian, A high-order discontinuous Galerkin pressure robust splitting scheme for incompressible flows, *arXiv preprint arXiv:1912.10242*, 2019.

M. Piatkowski, S. Müthing, and P. Bastian, A stable and high-order accurate discontinuous Galerkin based splitting method for the incompressible Navier–Stokes equations, *Journal of Computational Physics* **356**, 220 – 239, 2018.

S.-M. Piatkowski, *A Spectral Discontinuous Galerkin method for incompressible flow with Applications to turbulence*, PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2019.

S. B. Pope, *Turbulent flows*, IOP Publishing, 2001.

S. B. Pope, Ten questions concerning the large-eddy simulation of turbulent flows, *New Journal of Physics* **6**, 35, 2004.

F. Prill, M. Lukov-Medviov, and R. Hartmann, Smoothed aggregation multigrid for the discontinuous Galerkin method, *SIAM Journal on Scientific Computing* **31**, 3503–3528, 2009.

S. Ramakrishnan and S. S. Collis, Multiscale modeling for turbulence simulation in complex geometries, *AIAA paper* **241**, 2004, 2004.

P. Rasetarinera and M. Hussaini, An Efficient Implicit Discontinuous Spectral Galerkin Method, *Journal of Computational Physics* **172**, 718 – 738, 2001.

U. Rasthofer and V. Gravemeier, Multifractal subgrid-scale modeling within a variational multiscale method for large-eddy simulation of turbulent flow, *Journal of Computational Physics* **234**, 79 – 107, 2013.

U. Rasthofer and V. Gravemeier, Recent developments in variational multiscale methods for large-eddy simulation of turbulent flow, *Archives of Computational Methods in Engineering* **25**, 647–690, 2018.

P.-A. Raviart and J.-M. Thomas, A mixed finite element method for 2-nd order elliptic problems, In *Mathematical aspects of finite element methods*, pages 292–315, Springer, 1977.

S. S. Ray, U. Frisch, S. Nazarenko, and T. Matsumoto, Resonance phenomenon for the Galerkin-truncated Burgers and Euler equations, *Physical Review E* **84**, 016301, 2011.

J.-F. Remacle, R. Gandham, and T. Warburton, GPU accelerated spectral finite elements on all-hex meshes, *Journal of Computational Physics* **324**, 246 – 257, 2016.

S. Rhebergen and B. Cockburn, A space–time hybridizable discontinuous Galerkin method for incompressible flows on deforming domains, *Journal of Computational Physics* **231**, 4185 – 4204, 2012.

S. Rhebergen and G. N. Wells, A hybridizable discontinuous Galerkin method for the Navier–Stokes equations with pointwise divergence-free velocity field, *Journal of Scientific Computing* **76**, 1484–1501, 2018.

L. Riccius. Advanced timestepping techniques for an incompressible Navier–Stokes solver with discontinuous Galerkin discretization. Term paper, Technical University of Munich, 2019.

T. Richter, A monolithic geometric multigrid solver for fluid–structure interactions in ALE formulation, *International Journal for Numerical Methods in Engineering* **104**, 372–390, 2015.

G. Robalo Rei. Evaluation of different turbulence models for turbulent channel flow and bluff body flow. Bachelor's thesis, Technical University of Munich, 2017.

D. Rojas, R. Boukharfane, L. Dalcin, D. C. Del Rey Fernndez, H. Ranocha, D. E. Keyes, and M. Parsani, On the robustness and performance of entropy stable collocated discontinuous Galerkin methods, *Journal of Computational Physics* **426**, 109891, 2021.

E. M. Rønquist and A. T. Patera, Spectral element multigrid. I. Formulation and numerical results, *Journal of Scientific Computing* **2**, 389–406, 1987.

C. J. Roth, K. M. Förster, A. Hilgendorff, B. Ertl-Wagner, W. A. Wall, and A. W. Flemmer, Gas exchange mechanisms in preterm infants on HFOV–a computational approach, *Scientific reports* **8**, 13008, 2018.

J. Rudi, A. C. I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P. W. J. Staar, Y. Ineichen, C. Bekas, A. Curioni, and O. Ghattas, An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth's mantle, In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, page 5. ACM, 2015.

A. M. Rueda-Ramrez, J. Manzanero, E. Ferrer, G. Rubio, and E. Valero, A p-multigrid strategy with anisotropic p-adaptation based on truncation errors for high-order discontinuous Galerkin methods, *Journal of Computational Physics* **378**, 209 – 233, 2019.

Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, second Edition, 2003.

Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, *SIAM Journal on Scientific Computing* **14**, 461–469, 1993.

Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM Journal on Scientific and Statistical Computing* **7**, 856–869, 1986.

P. Sackinger, P. Schunk, and R. Rao, A Newton–Raphson pseudo-solid domain mapping technique for free and moving boundary problems: A finite element implementation, *Journal of Computational Physics* **125**, 83 – 103, 1996.

P. Sagaut, *Large eddy simulation for incompressible flows: an introduction*, Springer Science & Business Media, 2006.

E.-W. Saw, D. Kuzzay, D. Faranda, A. Guittonneau, F. Daviaud, C. Wiertel-Gasquet, V. Padilla, and B. Dubrulle, Experimental characterization of extreme events of inertial dissipation in a turbulent swirling flow, *Nature communications* **7**, 12466, 2016.

M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher, *Benchmark computations of laminar flow around a cylinder*, Springer, 1996.

G. Schnücke, N. Krais, T. Bolemann, and G. J. Gassner, Entropy stable discontinuous Galerkin schemes on moving meshes with summation-by-parts property for hyperbolic conservation laws, *arXiv preprint arXiv:1812.09093*, 2018.

B. Schott, C. Ager, and W. Wall, A monolithic approach to fluid-structure interaction based on a hybrid Eulerian-ALE fluid domain decomposition involving cut elements, *International Journal for Numerical Methods in Engineering* **119**, 208–237, 2019.

F. S. Schranner, V. Rozov, and N. A. Adams, Optimization of an implicit large-eddy simulation method for underresolved incompressible flow simulations, *AIAA Journal* **54**, 1567–1577, 2016.

P. W. Schroeder, *Robustness of High-Order Divergence-Free Finite Element Methods for Incompressible Computational Fluid Dynamics*, PhD thesis, Georg-August-Universität Göttingen, 2019.

P. W. Schroeder and G. Lube, Stabilised dG-FEM for incompressible natural convection flows with boundary and moving interior layers on non-adapted meshes, *Journal of Computational Physics* **335**, 760 – 779, 2017.

P. W. Schroeder and G. Lube, Divergence-free $H(\mathrm{div})$-FEM for time-dependent incompressible flows with applications to high Reynolds number vortex dynamics, *Journal of Scientific Computing* **75**, 830–858, 2018.

K. Sengupta, F. Mashayek, and G. Jacobs, Large-eddy simulation using a discontinuous Galerkin spectral element method, In *45th AIAA Aerospace Sciences Meeting and Exhibit*, pages 8–11. AIAA-2007-402. AIAA Reno, NV, 2007.

D. Serino, J. Banks, W. Henshaw, and D. Schwendeman, A stable added-mass partitioned (AMP) algorithm for elastic solids and incompressible flow, *Journal of Computational Physics* **399**, 108923, 2019.

K. Shahbazi, An explicit expression for the penalty parameter of the interior penalty method, *Journal of Computational Physics* **205**, 401 – 407, 2005.

K. Shahbazi, P. F. Fischer, and C. R. Ethier, A high-order discontinuous Galerkin method for the unsteady incompressible Navier–Stokes equations, *Journal of Computational Physics* **222**, 391 – 407, 2007.

K. Shahbazi, D. J. Mavriplis, and N. K. Burgess, Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, *Journal of Computational Physics* **228**, 7917 – 7940, 2009.

J. P. Sheldon, S. T. Miller, J. S. Pitt, et al., Methodology for comparing coupling algorithms for fluid-structure interaction problems, *World Journal of Mechanics* **4**, 54, 2014.

J. P. Sheldon, S. T. Miller, and J. S. Pitt, A hybridizable discontinuous Galerkin method for modeling fluid–structure interaction, *Journal of Computational Physics* **326**, 91 – 114, 2016.

C.-W. Shu, W.-S. Don, D. Gottlieb, O. Schilling, and L. Jameson, Numerical convergence study of nearly incompressible, inviscid Taylor–Green vortex flow, *Journal of Scientific Computing* **24**, 1–27, 2005.

C. Siefert, R. Tuminaro, A. Gerstenberger, G. Scovazzi, and S. S. Collis, Algebraic multigrid techniques for discontinuous Galerkin methods with varying polynomial order, *Computational Geosciences* **18**, 597–612, 2014.

D. Silvester, H. Elman, D. Kay, and A. Wathen, Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow, *Journal of Computational and Applied Mathematics* **128**, 261–279, 2001.

T. Spenke, N. Hosters, and M. Behr, A multi-vector interface quasi-Newton method with linear complexity for partitioned fluid–structure interaction, *Computer Methods in Applied Mechanics and Engineering* **361**, 112810, 2020.

K. R. Sreenivasan, An update on the energy dissipation rate in isotropic turbulence, *Physics of Fluids* **10**, 528–529, 1998.

K. Stein, T. Tezduyar, and R. Benney, Mesh Moving Techniques for Fluid-Structure Interactions With Large Displacements, *Journal of Applied Mechanics* **70**, 58–63, 2003.

D. T. Steinmoeller, M. Stastna, and K. G. Lamb, A short note on the discontinuous Galerkin discretization of the pressure projection operator in incompressible flow, *Journal of Computational Physics* **251**, 480 – 486, 2013.

J. Stiller, Robust multigrid for high-order discontinuous Galerkin methods: A fast Poisson solver suitable for high-aspect ratio Cartesian grids, *Journal of Computational Physics* **327**, 317 – 336, 2016.

J. Stiller, Nonuniformly Weighted Schwarz Smoothers for Spectral Element Multigrid, *Journal of Scientific Computing* **72**, 81–96, 2017a.

J. Stiller, Robust Multigrid for Cartesian Interior Penalty DG Formulations of the Poisson Equation in 3D, In M. L. Bittencourt, N. A. Dumont, and J. S. Hesthaven (eds.), *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2016*, pages 189–201, Cham, 2017b, Springer International Publishing.

C. Sulem, P. L. Sulem, and H. Frisch, Tracing complex singularities with spectral methods, *Journal of Computational Physics* **50**, 138 – 161, 1983.

T. Sun, L. Mitchell, K. Kulkarni, A. Klöckner, D. A. Ham, and P. H. Kelly, A study of vectorization for matrix-free finite element methods, *The International Journal of High Performance Computing Applications* **34**, 629–644, 2020.

H. Sundar, G. Biros, C. Burstedde, J. Rudi, O. Ghattas, and G. Stadler, Parallel geometric-algebraic multigrid on unstructured forests of octrees, In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 43. IEEE Computer Society Press, 2012.

H. Sundar, G. Stadler, and G. Biros, Comparison of multigrid algorithms for high-order continuous finite element discretizations, *Numerical Linear Algebra with Applications* **22**, 664–680, 2015.

K. Świrydowicz, N. Chalmers, A. Karakus, and T. Warburton, Acceleration of tensor-product operations for high-order finite element methods, *The International Journal of High Performance Computing Applications* **33**, 735–757, 2019.

M. Tavelli and M. Dumbser, A staggered semi-implicit discontinuous Galerkin method for the two dimensional incompressible Navier–Stokes equations, *Applied Mathematics and Computation* **248**, 70 – 92, 2014.

M. Tavelli and M. Dumbser, A staggered space–time discontinuous Galerkin method for the three-dimensional incompressible Navier–Stokes equations on unstructured tetrahedral meshes, *Journal of Computational Physics* **319**, 294–323, 2016.

G. I. Taylor, Statistical theory of turbulence, *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* **151**, 421–444, 1935.

G. Taylor and A. Green, Mechanism of the production of small eddies from large ones, *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* **158**, 499–521, 1937.

P. D. Thomas and C. K. Lombard, Geometric conservation law and its application to flow computations on moving grids, *AIAA Journal* **17**, 1030–1037, 1979.

L. Timmermans, P. Minev, and F. Van De Vosse, An approximate projection scheme for incompressible flow using spectral elements, *International Journal for Numerical Methods in Fluids* **22**, 673–688, 1996.

J. Treibig, G. Hager, and G. Wellein, Likwid: A lightweight performance-oriented tool suite for x86 multicore environments, In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 207–216. IEEE, 2010.

U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Elsevier Academic Press, London, 2001.

H. M. Tufo and P. F. Fischer, Terascale spectral element algorithms and implementations, In *Proceedings of the 1999 ACM/IEEE conference on Supercomputing*, pages 68–es, 1999.

S. Turek and J. Hron, Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow, In H.-J. Bungartz and M. Schäfer (eds.), *Fluid-Structure Interaction*, pages 371–385, Berlin, Heidelberg, 2006, Springer Berlin Heidelberg.

A. Uranga, P.-O. Persson, M. Drela, and J. Peraire, Implicit Large Eddy Simulation of transition to turbulence at low Reynolds numbers using a Discontinuous Galerkin method, *International Journal for Numerical Methods in Engineering* **87**, 232–261, 2011.

E. H. van Brummelen, Added Mass Effects of Compressible and Incompressible Flows in Fluid-Structure Interaction, *Journal of Applied Mechanics* **76**, 2009.

F. Van Der Bos and B. J. Geurts, Computational error-analysis of a discontinuous Galerkin discretization applied to large-eddy simulation of homogeneous turbulence, *Computer Methods in Applied Mechanics and Engineering* **199**, 903 – 915, 2010.

J. Van Kan, A second-order accurate pressure-correction scheme for viscous incompressible flow, *SIAM Journal on Scientific and Statistical Computing* **7**, 870–891, 1986.

R. S. Varga, *Matrix iterative analysis*, Springer, Berlin, 2nd Edition, 2009.

B. Vermeire, F. Witherden, and P. Vincent, On the utility of GPU accelerated high-order methods for unsteady flow simulations: A comparison with industry-standard tools, *Journal of Computational Physics* **334**, 497 – 521, 2017.

P. E. Vos, S. J. Sherwin, and R. M. Kirby, From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations, *Journal of Computational Physics* **229**, 5161 – 5181, 2010.

W. A. Wall, A. Gerstenberger, P. Gamnitzer, C. Förster, and E. Ramm, Large deformation fluid-structure interaction – advances in ALE methods and new fixed grid approaches, In H.-J. Bungartz and M. Schäfer (eds.), *Fluid-Structure Interaction*, pages 195–232, Berlin, Heidelberg, 2006, Springer Berlin Heidelberg.

C. Waluga, B. Wohlmuth, and U. Rüde, Mass-corrections for the conservative coupling of flow and transport on collocated meshes, *Journal of Computational Physics* **305**, 319 – 332, 2016.

D. Wang and S. J. Ruuth, Variable step-size implicit-explicit linear multistep methods for time-dependent partial differential equations, *Journal of Computational Mathematics*, 838–855, 2008.

Y. Wang, A. Quaini, and S. Čanić, A higher-order discontinuous Galerkin/arbitrary Lagrangian Eulerian partitioned approach to solving fluid–structure interaction problems with incompressible, viscous fluids and elastic structures, *Journal of Scientific Computing* **76**, 481–520, 2018.

Y. Wang. Numerical investigations on the efficiency of an in-house high-order discontinuous Galerkin flow solver compared to OpenFOAM. Master's thesis, Technical University of Munich, 2019.

Z. Wang and A. Przekwas, Unsteady flow computation using moving grid with mesh enrichment, In *32nd Aerospace Sciences Meeting and Exhibit*, page 285, 1994.

Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal, High-order CFD Methods: Current status and perspective, *International Journal for Numerical Methods in Fluids* **72**, 811–845, 2013.

L. Wei and A. Pollard, Direct numerical simulation of compressible turbulent channel flows using the discontinuous Galerkin method, *Computers & Fluids* **47**, 85 – 100, 2011.

C. C. Wiart, K. Hillewaert, L. Bricteux, and G. Winckelmans, Implicit LES of free and wall-bounded turbulent flows based on the discontinuous Galerkin/symmetric interior penalty method, *International Journal for Numerical Methods in Fluids* **78**, 335–354, 2015.

T. Wick, Fluid-structure interactions using different mesh motion techniques, *Computers & Structures* **89**, 1456 – 1467, 2011.

E. Wiedemann, Weak-strong uniqueness in fluid dynamics, *arXiv preprint arXiv:1705.04220*, 2017.

M. V. Wilkes, The memory gap and the future of high performance memories, *SIGARCH Computer Architecture News* **29**, 27, 2001.

S. Williams, A. Waterman, and D. Patterson, Roofline: an insightful visual performance model for multicore architectures, *Communications of the ACM* **52**, 65–76, 2009.

A. R. Winters, R. C. Moura, G. Mengaldo, G. J. Gassner, S. Walch, J. Peiró, and S. J. Sherwin, A comparative study on polynomial dealiasing and split form discontinuous Galerkin schemes for under-resolved turbulence computations, *Journal of Computational Physics* **372**, 1 – 21, 2018.

F. Witherden, A. Farrington, and P. Vincent, PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach, *Computer Physics Communications* **185**, 3028 – 3040, 2014.

J. Witte, D. Arndt, and G. Kanschat, Fast tensor product Schwarz smoothers for high-order discontinuous Galerkin methods, *Computational Methods in Applied Mathematics*, 000010151520200078, 2020.

L. Xu, X. Xu, X. Ren, Y. Guo, Y. Feng, and X. Yang, Stability evaluation of high-order splitting method for incompressible flow based on discontinuous velocity and continuous pressure, *Advances in Mechanical Engineering* **11**, 1687814019855586, 2019.

S. Yakovlev, D. Moxey, R. M. Kirby, and S. J. Sherwin, To CG or to HDG: A comparative study in 3D, *Journal of Scientific Computing* **67**, 192–220, 2016.

Z.-G. Yan, Y. Pan, G. Castiglioni, K. Hillewaert, J. Peir, D. Moxey, and S. J. Sherwin, Nektar++: Design and implementation of an implicit, spectral/hp element, compressible flow solver using a Jacobian-free Newton Krylov approach, *Computers & Mathematics with Applications* **81**, 351 – 372, 2021.

Z. Yang and D. Mavriplis, Unstructured dynamic meshes with higher-order time integration schemes for the unsteady Navier-Stokes equations, In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, page 1222, 2005.

S. Yigit, M. Schäfer, and M. Heck, Grid movement techniques and their influence on laminar fluid–structure interaction computations, *Journal of Fluids and Structures* **24**, 819 – 832, 2008.

X. Zhang, D. Stanescu, and J. W. Naughton, Development of a spectral element DNS/LES method for turbulent flow simulations, *FEDSM2007-37443*, 2007.

Verzeichnis der betreuten Studienarbeiten

Im Rahmen dieser Dissertation entstanden am Lehrstuhl für Numerische Mechanik (LNM) in den Jahren von 2016 bis 2020 unter wesentlicher wissenschaftlicher, fachlicher und inhaltlicher Anleitung des Autors die im Folgenden aufgeführten studentischen Arbeiten. Der Autor dankt allen Studierenden für ihr Engagement bei der Unterstützung dieser wissenschaftlichen Arbeit.

Studierende(r)	**Studienarbeit**
Shahbaz Haider	*Numerical analysis and comparison of high-performance, incompressible Navier–Stokes solvers based on the discontinuous Galerkin method*, Masterarbeit, 2017
Tim Dockhorn	*Turbulence modeling for large eddy simulation of incompressible flows using high-order discontinuous Galerkin methods*, Bachelorarbeit, 2017, eingeflossen in Abschnitt 2.6.7.4
Christoph Haslinger	*A discontinuous Galerkin approach for the compressible Navier–Stokes equations*, Masterarbeit, 2017
Anian Fuchs	*Hermite-artige Basisfunktionen für diskontinuierliche Galerkin-Verfahren hoher Ordnung*, Bachelorarbeit, 2017
Oliver Neumann	*Analyse des spektralen Verhaltens von diskontinuierlichen Galerkin-Verfahren für turbulente Strömungen*, Bachelorarbeit, 2018, eingeflossen in Abschnitt 7.3.3
Peter Münch	*An efficient hybrid multigrid solver for high-order discontinuous Galerkin methods*, Masterarbeit, 2018, eingeflossen in Abschnitt 5.1 und 5.2
Leon Riccius	*Advanced timestepping techniques for an incompressible Navier–Stokes solver with discontinuous Galerkin discretization*, Semesterarbeit, 2019
Yingxian Wang	*Numerical Investigations on the Efficiency of an In-House High-Order Discontinuous Galerkin Flow Solver Compared to OpenFOAM*, Masterarbeit, 2019, eingeflossen in Abschnitt 6.3.4.2
Johannes Heinz	*High-Order Discontinuous Galerkin Methods on Moving Meshes for the Incompressible Navier–Stokes Equations*, Masterarbeit, 2019, eingeflossen in Kapitel 8
Xuhui Zhang	*Computational Fluid Dynamics Simulation of an Endotracheal Tube Using a High-Order Discontinuous Galerkin Turbulent Flow Solver and Patient-Specific Settings*, Bachelorarbeit, 2019
Anian Fuchs	*Simulation of aeroacoustic problems with discontinuous Galerkin FEM*, Semesterarbeit, 2020