Implementing a time-accurate implicit-explicit method with error estimation for compressible flow

By K. Oßwald*, V. Hannemann* AND P. Birken‡

* Institute of Aerodynamics and Flow Technology, DLR Göttingen
‡ Institute of Mathematics, University of Kassel

An implicit-explicit Runge-Kutta method is implemented in the DLR-TAU code to improve the time integration procedure when performing time-accurate simulations like Detached-Eddy Simulation of high Reynolds number turbulent flows. The idea is to avoid the use of the standard dual time stepping approach and instead use an explicit Runge-Kutta scheme in most parts of the grids where the stable time step size is in the order of the physical time scale to resolve the large eddies. In the boundary layers where the underlying RANS model allows for highly stretched grids the local time step reduces significantly. Therefore, fluxes along wall-normal lines are treated implicitly by the time integration scheme. The implicit-explicit scheme together with the error estimation approach is presented in detail.

1. Introduction

The released version of the DLR TAU code provides two possibilities to conduct time accurate flow simulations: an explicit global time stepping or Backward Differentiation Formulas.

The computational effort for each physical time step in the explicit method is orders of magnitude lower than for the iterations needed to solve the system of equations when using an implicit method. But the time step size of the explicit method is limited to the smallest time step for which the scheme is still stable in all cells. Therefore, the explicit scheme with global time stepping is superior as long as the allowed time step is in the order of the physical time step of interest. In cases where boundary layers have to be resolved at high Reynolds numbers or fast chemical reactions are taking place an implicit approach becomes more efficient than an explicit one. To resolve the boundary layers, cells with large aspect ratio are used leading to so-called grid-induced stiffness since the stable time step size decreases dramatically.

When conducting a Detached-Eddy Simulation (DES) the physical time scale to be resolved is in the order of the local explicit time step in those parts of the grid where the large eddies are resolved. In the boundary layers where the underlying RANS model allows for highly stretched grids the local time step reduces significantly and an implicit approach is desired for efficient time accurate calculations. A promising way to combine as much explicit time stepping as possible with only as much implicit method as necessary to keep the global time step in the order of the relevant physical time step are the so called implicit-explicit (IMEX) Runge-Kutta methods [1, 2].

Since the stability of the scheme is not affected anymore by the time step size when

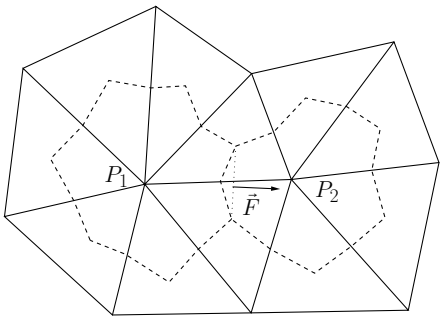

FIGURE 1. Control volumes of dual grid (– – – –) around grid nodes P_1 and P_2, which are vertices of the initial grid (———). The normal flux across the dual mesh face (·····) is referred to as $\vec{F}$.

an implicit method is used, an estimation of the temporal error produced in one time step is necessary to retain a certain level of accuracy. Depending on the estimated error, an adaptive time step can be used which is more efficient than calculating with a constant time step.

2. The DLR-TAU Code

TAU is a second-order, parallel finite volume scheme which solves the compressible, three-dimensional Navier-Stokes equations on the dual grid of a conform mesh consisting of tetrahedra, prisms, pyramids and hexahedra. The governing equations in conservative form can be written as

$$\frac{\partial}{\partial t} \iiint_V \vec{\mathbf{U}} dV = - \iint_{\partial V} \vec{\mathbf{F}} \cdot \vec{n} \, dS \tag{2.1}$$

with the vector

$$\vec{\mathbf{U}} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} \tag{2.2}$$

of the conserved variables (mass, momentum, energy, respectively), and $\vec{F}$ being the inviscid and viscous contributions of the flux vectors in the three coordinate directions. To close the system, the equation of state for an ideal gas is considered. The volume of an arbitrary dual grid cell is V, with an outward facing normal vector $\vec{n}$ on its boundary ∂V.

2.1. *Spatial discretisation*

The type of spatial discretisation is cell-vertex because the flow variables are stored on the vertices of the initial grid. In a preprocessing step the dual metric is computed, illustrated in Fig. 1.

Either an upwind or a central scheme is availabe for the computation of the fluxes $\vec{F}$. For the scheme presented here, an upwind scheme is used.

2.2. *Time integration*

After the computation of the fluxes the solution is advanced in time. The temporal integration of the state variables at a grid point is generally formulated as

$$\frac{d\mathbf{U}}{dt} + \mathbf{R}\left(t, \mathbf{U}\left(t\right)\right) = 0, \quad \mathbf{U}(0) = \mathbf{U}_0 \tag{2.3}$$

where the computation of the spatial discretization $\mathbf{R}$ has already been completed and is only depending on the time t. In TAU, time-accurate simulations can be conducted by using either a global time stepping scheme or Backward Differentiation Formulas (BDF).

When using BDF methods, a steady state solution is computed in a pseudo time for each physical time step. This is called a dual time stepping approach since the outer iteration procedure treats the physical time step and inner iterations are only performed in pseudo time. Acceleration techniques like multigrid are available to improve the convergence behaviour of the inner iteration procedure. Despite this, the large number of inner iterations to advance the solution to steady state is also a drawback of this approach. For example the number of inner iterations for a RANS computation is in the range of 50 to 200, depending on the case.

The second approach for time-accurate simulations in TAU is an explicit Runge-Kutta (ERK) method. ERK methods are very efficient if the governing equations are non-stiff. On the other side, the time step is restricted by the CFL-condition, the stability limit of an explicit method. In case of stiff problems only a very small time step size can be used for a stable calculation and the number of time steps increases significantly to advance the simulation in time. For this reason, ERK methods become inefficient and the application of implicit methods is preferred.

3. Runge-Kutta methods

Runge-Kutta methods are widely used to integrate general ordinary differential equations (ODE) like

$$\frac{d\mathbf{U}}{dt} = \mathbf{F}\left(t, \mathbf{U}\left(t\right)\right), \quad \mathbf{U}(0) = \mathbf{U}_0 \tag{3.1}$$

with respect to the variable t, here defined as the physical time under consideration of the initial condition $\mathbf{U}_0$. In TAU, time integration is done by solving such an ODE, compare Eq. (2.3).

3.1. *General formulation*

To integrate Eq. (3.1) from discrete time levels n to $n+1$, a s-stage Runge-Kutta (RK) scheme can be applied, formulated as

$$\begin{aligned} \mathbf{U}^{(i)} &= \mathbf{U}^{(n)} + \Delta t \sum_{j=1}^{s} a_{ij} \mathbf{F}\left(t^{(n)} + c_j \Delta t, \mathbf{U}^{(j)}\right), \quad 1 \le i \le s, \\ \mathbf{U}^{(n+1)} &= \mathbf{U}^{(n)} + \Delta t \sum_{i=1}^{s} b_i \mathbf{F}\left(t^{(n)} + c_i \Delta t, \mathbf{U}^{(i)}\right). \end{aligned} \tag{3.2}$$

Each Runge-Kutta stage $\mathbf{U}^{(i)}$ is an approximation to the solution at intermediate time levels ($t^{(n)} + c_j \Delta t$), evaluated at the Runge-Kutta nodes c_j and weighted with the coef-

ficients a_{ij} for the ith stage. A common design criterion for Runge-Kutta methods is

$$c_i = \sum_{j=1}^{s} a_{ij}. \tag{3.3}$$

The final stage is then computed with the weights b_i. All these parameters which determine the accuracy and stability of a RK method can also be represented in the so-called Butcher tableau [3].

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

In case of an explicit Runge-Kutta method all coefficients a_{ij} of the upper triangular and on the diagonal of the Butcher tableau are zero ($a_{ij} = 0$ for $j \geq i$). If any stage $\mathbf{U}^{(i)}$ cannot be solely computed from lower stages the method is called implicit.

3.2. *Implicit-explicit Runge-Kutta methods*

Suppose the right hand side of Eq. (3.1) is a sum of N terms and every term is integrated by its own Runge-Kutta method. The generalisation to N-additive Runge-Kutta methods (ARK$_N$) can be done in a straightforward manner from Eq. (3.2) and a general formulation is

$$\begin{aligned} \mathbf{U}^{(i)} &= \mathbf{U}^{(n)} + \Delta t \sum_{k=1}^{N} \sum_{j=1}^{s} a_{ij}^{[k]} \mathbf{F}^{[k]} \left(t^{(n)} + c_j \Delta t, \mathbf{U}^{(j)} \right), \quad 1 \leq i \leq s, \\ \mathbf{U}^{(n+1)} &= \mathbf{U}^{(n)} + \Delta t \sum_{k=1}^{N} \sum_{i=1}^{s} b_i^{[k]} \mathbf{F}^{[k]} \left(t^{(n)} + c_i \Delta t, \mathbf{U}^{(i)} \right). \end{aligned} \tag{3.4}$$

A special case of an ARK$_N$ method is an implicit-explicit Runge-Kutta (IMEX-RK) method where $N = 2$, integrating one term explicitly and the other one implicitly. Such an ARK$_2$ method can be formulated as

$$\begin{aligned} \mathbf{U}_{[E]}^{(i)} &= \mathbf{U}_{[E]}^{(n)} + \Delta t \sum_{j=1}^{s} a_{ij}^{[E]} \mathbf{F}^{[E]} \left(t^{(n)} + c_j \Delta t, \mathbf{U}_{[EX]}^{(j)} \right), \quad 1 \leq i \leq s, \\ \mathbf{U}_{[I]}^{(i)} &= \mathbf{U}_{[I]}^{(n)} + \Delta t \sum_{j=1}^{s} a_{ij}^{[I]} \mathbf{F}^{[I]} \left(t^{(n)} + c_j \Delta t, \mathbf{U}_{[IM]}^{(j)} \right), \quad 1 \leq i \leq s, \\ \mathbf{U}^{(n+1)} &= \mathbf{U}^{(n)} + \Delta t \sum_{i=1}^{s} b_i \mathbf{F} \left(t^{(n)} + c_i \Delta t, \mathbf{U}^{(i)} \right), \end{aligned} \tag{3.5}$$

where the superscripts $[E]$ and $[I]$ denote the explicit and implicit method, respectively. In this case here, the solution at the new time level can be easily computed by simplifying the Runge-Kutta weights $b_i^{[E]} = b_i^{[I]} := b_i$ to be the same for both the explicit and the implicit method. Kennedy and Carpenter [1] derived a set of ARK$_2$ methods and investigated their application to convection-diffusion-reaction equations. Additional applications of these methods are presented e.g. by Kanevsky et al. [2] and Birken [4].

3.3. *Error estimation and time step control*

So-called *embedded methods* are a simple way to estimate the error in a single integration step. Such a method consists of two Runge-Kutta methods sharing the same nodes and coefficients but using different sets of weights. By this means, the whole process of stage evaluation in Eq. (3.5) can be reused to compute the final stage $\hat{\mathbf{U}}^{(n+1)}$ of the embedded scheme on the new time level with almost no additional computational cost.

$$\hat{\mathbf{U}}^{(n+1)} = \mathbf{U}^{(n)} + \Delta t \sum_{i=1}^{s} \hat{b}_i \mathbf{F}\left(t^{(n)} + c_i \Delta t, \mathbf{U}^{(i)}\right) \tag{3.6}$$

The order q of the embedded scheme is always less than the order p of the original scheme. Usually, the suboptimal weights $\hat{b}_i$ of the embedded scheme are designed to satisfy the relation $q + 1 = p$ for the leading order of the error of both schemes. Thus, an estimate of the error produced in a single time step is simply the difference between both solutions. The temporal error is computed by subtracting the solution of the embedded scheme from the solution evaluated by the main scheme

$$\delta = \mathbf{U} - \hat{\mathbf{U}} = \Delta t \sum_{i=1}^{s} (b_i - \hat{b}_i) \mathbf{F}(t^{(n)} + c_i \Delta t, \mathbf{U}^{(i)}). \tag{3.7}$$

A time adaptive scheme can be build with the knowledge of the temporal error. Time adaptivity is useful to speed up computations and/or to provide a certain accuracy of the scheme.

4. Implementation details

This section gives a detailed overview of the chosen IMEX-RK methods and the solution procedure for the line-implicit scheme. Additionally, the determination of a line is briefly described. The implementation of these modules into a branch of the TAU code was the work during the Summer Program.

Before the start of this project, TAU was unable to deal with general Runge-Kutta tableaus. Therefore, the code structure was first extended to handle general Runge-Kutta methods.

4.1. *Implemented IMEX-RK methods*

IMEX-RK schemes belong to the class of Additive Runge-Kutta (ARK) schemes. The ARK3(2), ARK4(3) and ARK5(4) methods by Kennedy and Carpenter [1] are well suited for the approach presented here and are therefore added to the time integration methods in TAU.

Each of these methods is a combination of an explicit Runge-Kutta (ERK) method and a so-called explicit, singly diagonally implicit Runge-Kutta (ESDIRK) method. Amongst other design criteria, the implemented IMEX-RK methods follow the general Butcher tableaus shown in Tab. 1 with $a_{ij}^{[E]}$ and $a_{ij}^{[I]}$ being the coefficients for the ERK and ESDIRK methods, respectively.

The two RK schemes are coupled through the weights $b_i = b_i^{[E]} = b_i^{[I]}$ and also through the nodes $c_i = c_i^{[E]} = c_i^{[I]}$, so in both schemes each Runge-Kutta stage is evaluated at the same intermediate time level. Furthermore, the weights $\hat{b}_i$ of the embedded scheme are the same for both ERK and ESDIRK methods to compute the temporal error after every time step. For the exact values of the IMEX-RK coefficients the reader is referred to [1].

0	0	0	0	0	$\dots$	0
2γ	2γ	0	0	0	$\dots$	0
c_3	$a_{31}^{[E]}$	$a_{32}^{[E]}$	0	0	$\ddots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$	$\ddots$	0
c_{s-1}	$a_{s-1,1}^{[E]}$	$a_{s-1,2}^{[E]}$	$a_{s-1,3}^{[E]}$	$\ddots$	0	0
1	$a_{s,1}^{[E]}$	$a_{s,2}^{[E]}$	$a_{s,3}^{[E]}$	$\dots$	$a_{s-1,s}^{[E]}$	0
	b_1	b_2	b_3	$\dots$	b_{s-1}	γ
	$\hat{b}_1$	$\hat{b}_2$	$\hat{b}_3$	$\dots$	$\hat{b}_{s-1}$	$\hat{b}_s$

0	0	0	0	0	$\dots$	0
2γ	γ	γ	0	0	$\dots$	0
c_3	$a_{31}^{[I]}$	$a_{32}^{[I]}$	γ	0	$\ddots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$	$\ddots$	0
c_{s-1}	$a_{s-1,1}^{[I]}$	$a_{s-1,2}^{[I]}$	$a_{s-1,3}^{[I]}$	$\ddots$	γ	0
1	$a_{s,1}^{[I]}$	$a_{s,2}^{[I]}$	$a_{s,3}^{[I]}$	$\dots$	$a_{s-1,s}^{[I]}$	γ
	b_1	b_2	b_3	$\dots$	b_{s-1}	γ
	$\hat{b}_1$	$\hat{b}_2$	$\hat{b}_3$	$\dots$	$\hat{b}_{s-1}$	$\hat{b}_s$

TABLE 1. General Butcher tableaus for the explicit (left) and the implicit (right) Runge-Kutta method of the implemented IMEX-RK methods.

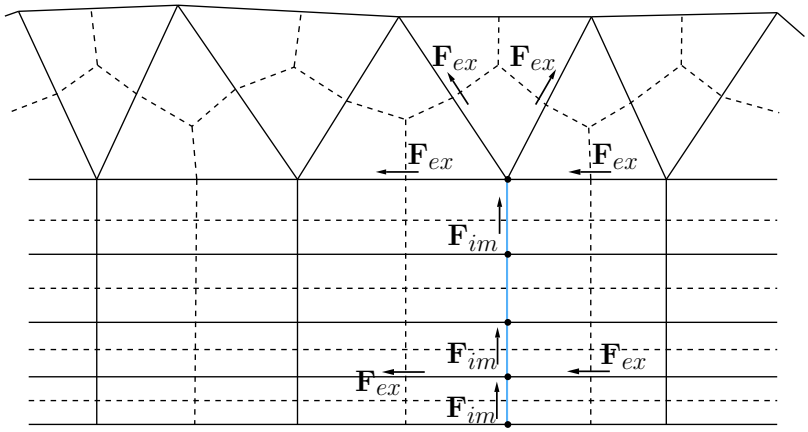

FIGURE 2. Example of an implicit treated line (light blue) in the near-wall region of a hybrid grid. The primary grid (———) and dual grid (– – – –) is shown as well as the line points (dots) on the blue line. Fluxes $\mathbf{F}_{ex}$ are integrated with an explicit scheme whereas fluxes $\mathbf{F}_{im}$ along a line are treated implicitly.

A stage order of two is guaranteed by the explicit first stage of the ESDIRK method. Additionally, the implicit method is stiffly accurate, in case the coefficient matrix A is non-singular and the last stage satisfies $a_{sj}^{[I]} = b_j$ for $j = 1, \dots, s$.

4.2. *Solution algorithm*

Every line can be treated separately because every line point is implicitly coupled to its neighbouring line points but has no implicit dependency to any other grid points. An example of a line treated implicitly in a dual grid flow solver is shown in Fig. 2. From this image it is easy to derive the formulation of the implicit-explicit scheme. For a line point

q the state $\mathbf{U}_q^{(i)}$ of the ith RK stage can be computed from

$$\begin{aligned}\mathbf{U}_q^{(i)} = \mathbf{U}_q^{(n)} &+ \underbrace{\Delta t \left[\sum_{j=1}^{i-1} a_{ij}^{ERK} \left[\sum_{l \in N_E(q)} \mathbf{F}^{[E]} \left(\mathbf{U}_q^{(j)}, \mathbf{U}_l^{(j)} \right) \right] \right]}_{\mathbf{X}_1^{(i)}} \\ &+ \Delta t \left[\sum_{j=1}^{i} a_{ij}^{ESDIRK} \left[\sum_{l \in N_L(q)} \mathbf{F} \left(\mathbf{U}_q^{(j)}, \mathbf{U}_l^{(j)} \right) \right] \right],\end{aligned} \tag{4.1}$$

where $a_{ij}^{ERK} = a_{ij}^{[E]}$ and $a_{ij}^{ESDIRK} = a_{ij}^{[I]}$ are the RK coefficients of Tab. 1. The number of faces of a dual cell which are treated explicitly and implicitly for the current line point q are denoted as N_E and N_L, respectively.

Since an ESDIRK method is applied the latter term of Eq. (4.1) can be rearranged in a known and an unknown part by the time the current Runge-Kutta stage is evaluated. Additionally, the Runge-Kutta coefficients $a_{ii}^{ESDIRK} := \gamma$ on the diagonal in the Butcher tableau are always the same for every stage.

$$\begin{aligned}\mathbf{U}_q^{(i)} = \mathbf{U}_q^{(n)} &+ \mathbf{X}_1^{(i)} \\ &+ \underbrace{\Delta t \left[\sum_{j=1}^{i-1} a_{ij}^{ESDIRK} \left[\sum_{l \in N_L(q)} \mathbf{F}^{[E]} \left(\mathbf{U}_q^{(j)}, \mathbf{U}_l^{(j)} \right) \right] \right]}_{\mathbf{X}_2^{(i)}} \\ &+ \gamma \Delta t \left[\sum_{l \in N_L(q)} \mathbf{F}^{[I]} \left(\mathbf{U}_q^{(i)}, \mathbf{U}_l^{(i)} \right) \right]\end{aligned} \tag{4.2}$$

Both sums $\mathbf{X}_1^{(i)}$ and $\mathbf{X}_2^{(i)}$ can be computed from previous stages, so Eq. (4.2) is a nonlinear equation for the unknown states $\mathbf{U}_q^{(i)}$. This leads to a system of N blocks of nonlinear equations for each line, where N is the number of line points and the size of a block depends on the number of conserved variables considered in the flow problem. One way of solving is to use a Newton method by iterating a solution from an initial guess. Rearranging Eq. (4.2) and setting it to zero leads to

$$\mathbf{H}\left(\mathbf{U}_q^{(i)}\right) := \mathbf{U}_q^{(n)} - \mathbf{U}_q^{(i)} + \mathbf{X}_1^{(i)} + \mathbf{X}_2^{(i)} + \gamma \Delta t \left[\sum_{l \in N_L(q)} \mathbf{F}^{[I]} \left(\mathbf{U}_q^{(i)}, \mathbf{U}_l^{(i)} \right) \right] \stackrel{!}{=} 0\,, \tag{4.3}$$

Newton's method to solve this equation is formulated as

$$\left. \frac{\partial \mathbf{H}}{\partial \mathbf{U}_q^{(i)}} \right|_{\mathbf{U}_{q,k}^{(i)}} \Delta \mathbf{U} = -\mathbf{H}\left(\mathbf{U}_{q,k}^{(i)}\right) \tag{4.4}$$

where the subscript k denotes the value on the kth iteration and $\Delta \mathbf{U}$ is the difference between the actual and the kth iterate value. This yields a linear system of equations with the jacobian $\mathbf{J} := \mathbf{H}'$ being a block-tridiagonal matrix because in this IMEX scheme a line point only depends on its direct neighbours. The derivative at the line point q of Eq. (4.3) with respect to the conservative variables taking into account the dependency

on its two neighbouring points $q-1$ and $q+1$ is

$$\mathbf{J}_{q,q} = -\mathbf{I} + \gamma\Delta t \left[\frac{\partial \mathbf{F}^{[I]}\left(\mathbf{U}_q^{(i)}, \mathbf{U}_{q-1}^{(i)}\right)}{\partial \mathbf{U}_q^{(i)}} + \frac{\partial \mathbf{F}^{[I]}\left(\mathbf{U}_q^{(i)}, \mathbf{U}_{q+1}^{(i)}\right)}{\partial \mathbf{U}_q^{(i)}} \right]. \tag{4.5}$$

A special case for the block $\mathbf{J}_{N,N}$ occurs in case of the line point N, neighbouring the unstructured part of the grid like in Fig. 2. If so, the second term in the bracket of Eq. (4.5) is zero because fluxes between point q and $q+1$ are advanced in time with the explicit scheme. The subdiagonal and superdiagonal entries of the jacobian matrix are

$$\mathbf{J}_{q,q-1} = \gamma\Delta t \frac{\partial \mathbf{F}^{[I]}\left(\mathbf{U}_q^{(i)}, \mathbf{U}_{q-1}^{(i)}\right)}{\partial \mathbf{U}_{q-1}^{(i)}} \tag{4.6}$$

and

$$\mathbf{J}_{q,q+1} = \gamma\Delta t \frac{\partial \mathbf{F}^{[I]}\left(\mathbf{U}_q^{(i)}, \mathbf{U}_{q+1}^{(i)}\right)}{\partial \mathbf{U}_{q+1}^{(i)}}, \tag{4.7}$$

respectively. The jacobian matrix of the whole system of equations is of size $N \times N$. Each block of the jacobian is a square matrix itself, the size being the number of conserved variables depending on the governing equations. Therefore, the size of the matrix varies whether a laminar or a turbulent flow is considered.

Since the solution algorithm used here leaves $\mathbf{X}_1^{(i)}$ and $\mathbf{X}_2^{(i)}$ unchanged throughout the iteration procedure they can be stored and reused in every iteration. Additionally, for every subsequent stage evaluation only the contribution from the current stage adds up to both terms, so there is no need to compute the whole expression from scratch.

The simplest starting guess for the Newton procedure is the state of the previous RK stage. For the second RK stage, this is just the previous time level in case of an ESDIRK method. For every subsequent stage the initial guess can be the iterated state from the previous one.

The iteration procedure is continued until a sufficient converged state is obtained which is equivalent to satisfy Eq. (4.3) as best as possible. Once every RK stage has been iterated, all line points are updated to the new time level by weighting the intermediate states of the RK stages. Additionally, the temporal error can now be computed by using the embedded scheme.

4.3. *Determination of lines*

Several line search algorithms can be found in the literature [5,6], providing the ability to find lines not ending on a boundary. None of the algorithms above is used in the IMEX-RK method presented here since the determination of lines is very easy in the present case.

Every starting point of a line is lying on a boundary. Additionally, a structured grid (containing quadrilaterals in 2D and prisms in 3D) is always assumed where the new time level is integrated implicitly when applying the IMEX-RK method.

Every boundary can be defined by the user to be treated by the IMEX-RK method. Therefore, the algorithm loops over every boundary and checks if the IMEX setting is true. If so, every grid point on this boundary serves as a starting point for a single line. Subsequent line points are easy to detect using the edge-based data structure of the TAU code. The line search algorithm stops automatically if

- the aspect ratio of the dual cell is less than a user-defined threshold,
- the maximum number of line points specified by the user is reached,
- the line would extend into the unstructured part of the grid.

Up to now, several special cases are not handled by the implementation and have to be avoided by the user. One such restriction is that a boundary grid point can only be associated to one line. Another drawback are lines emerging from a concave surface where lines may cross each other. For a more robust behaviour of the line search algorithm on general grids, improvements of the algorithm or an error handling of the mentioned restrictions have to be implemented in the future.

5. Conclusion and outlook

The implementation of an implicit-explicit Runge-Kutta method in the DLR-TAU code to improve the computational efficiency for time-accurate simulations was started during the Summer Program. To resolve high Reynolds number turbulent boundary layers, highly stretched grid cells are necessary. Therefore, wall-normal lines emanating from the wall are constructed in regions where cells with large aspect ratios are present. Along these lines, time integration is done implicitly to circumvent the severe time step restriction.

Since the implementation of the scheme was not finished during the Summer Program, a detailed report on the formulation of the implicit-explicit approach in a flow solver using a dual grid discretisation is given. The implementation of the new time integration scheme is work in progress.

The estimation of the temporal error is done via an embedded method with almost no additional computational cost. The knowledge of the temporal error is useful to extend the flow solver to a time-adaptive scheme to provide a certain accuracy in the implicit part of the time integration.

Acknowledgments

Financial support has been provided by the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) in the framework of the Sonderforschungsbereich Transregio 40.

References

[1] KENNEDY, C. A. AND CARPENTER, M. H. (2003). Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, **44**, 139–181.

[2] KANEVSKY, A., CARPENTER, M. H., GOTTLIEB, D. AND HESTHAVEN, J. S. (2007). Application of implicit-explicit high order Runge-Kutta methods to discontinuous-Galerkin schemes. *J. Comp. Phys.*, **225**, 1753–1781.

[3] BUTCHER, J. C. (2008). *Numerical Methods for Ordinary Differential Equations.* 2nd Ed., Wiley, Chichester, UK.

[4] BIRKEN, P. (2012). *Numerical methods for the unsteady compressible Navier-Stokes equations.* Habilitation Thesis, University of Kassel

[5] ELIASSON, P., WEINERFELT, P. AND NORDSTRÖM, J. (2009). Application of a line-implicit scheme on stretched unstructured grids. *AIAA 2009-0163*

[6] LANGER, S. (2013). Application of a line implicit method to fully coupled system of equations for turbulent flow problems. *Int. J. Computational Fluid Dynamics*, **27**(3), 131–150.