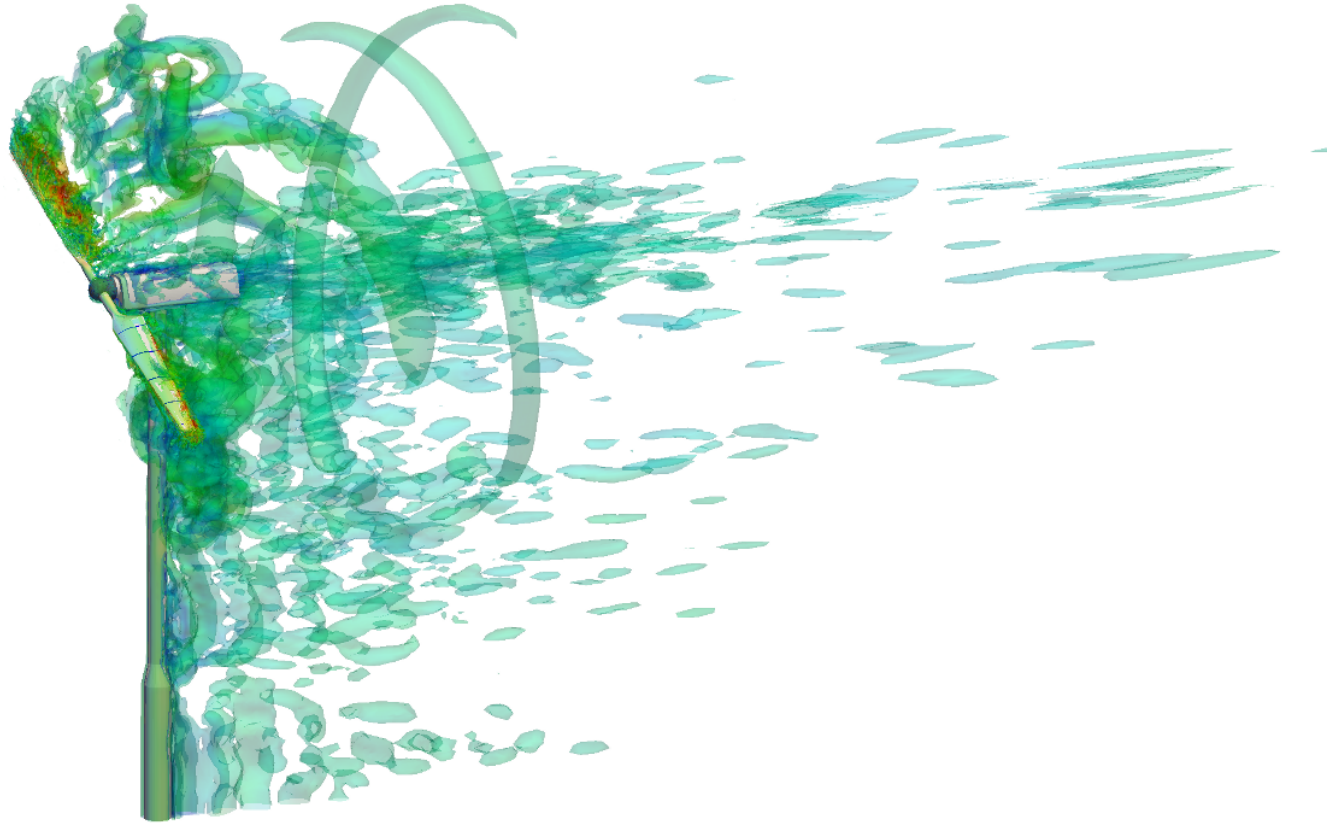
Lehrstuhl für Statik
Ingenieurfakultät Bau Geo Umwelt
Technische Universität München

Bachelor's Thesis
in Mechanical Engineering (Bachelor of Engineering (B.Eng.))
Hochschule für angewandte Wissenschaften Deggendorf

Towards a Turbulent Fluid–Structure–Signal Co–Simulation: Validate with the NREL 10–m Wind Turbine Testing in NASA Ames Wind Tunnel



Author Christopher Lerch, B.Eng. M.Sc. (hons) (TUM)

Examiner Prof. Dr. rer. nat. Stefan Schulte
Hochschule für angewandte Wissenschaften Deggendorf

Supervisor Dr.-Ing. Stefan Sicklinger, M.Sc. (hons) (TUM)
Technische Universität München

Deggendorf · München
March 20, 2013

Revision November 3, 2018 12:20pm

Bachelorarbeit

Hochschule für angewandte Wissenschaften Deggendorf

Fakultät Maschinenbau und Mechatronik

Studiengang Maschinenbau

Towards a Turbulent Fluid–Structure–Signal Co–Simulation: Validate with the NREL 10–m Wind Turbine Testing in NASA Ames Wind Tunnel

Bachelorarbeit zur Erlangung des akademischen Grades:

Bachelor of Engineering (B.Eng.)

vorgelegt von: Christopher Lerch, Deggendorf

Prüfer: Prof. Dr. rer. nat. Stefan Schulte

Deggendorf, 20. März 2013

Author

Christopher Lerch

Matrikelnr. 292752

Email christopher.lerch@stud.hdu-deggendorf.de

Hochschule für angewandte Wissenschaften Deggendorf

University of Applied Sciences (HDU)

Fakultät Maschinenbau und Mechatronik

Edlmairstraße 6 und 8

94469 Deggendorf, Germany

Phone +49 (0)991 3615 - 0

Fax +49 (0)991 3615 - 297

Email info@hdu-deggendorf.de

Web `http://www.hdu-deggendorf.de`

Supervisor

Dipl.-Ing. (FH) Stefan Sicklinger, M.Sc. (hons) (TUM)

Phone +49 (0)89 289 - 22426

Email stefan.sicklinger@tum.de

Technische Universität München (TUM)

Fakultät für Bauingenieur- und Vermessungswesen

Lehrstuhl für Statik

Prof. Dr.-Ing. Kai-Uwe Bletzinger

Arcisstraße 21

80333 München, Germany

Phone +49 (0)89 289 - 22422

Fax +49 (0)89 289 - 22421

Email statik@bv.tum.de

Web `http://www.st.bv.tum.de`

LaTeX

Disclaimer (*Erklärung*)

Name des Studierenden: Christopher LERCH

Name des Betreuenden: Prof. Dr. rer. nat. Stefan SCHULTE

Thema der Bachelorarbeit: TOWARDS A TURBULENT FLUID–STRUCTURE–SIGNAL CO–SIMULATION: VALIDATE WITH THE NREL 10–M WIND TURBINE TESTING IN NASA AMES WIND TUNNEL

1. Ich erkläre hiermit, dass ich die Bachelorarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

   Deggendorf, 20. März 2013 _____

   Christopher LERCH

2. Ich bin damit einverstanden, dass die von mir angefertigte Bachelorarbeit über die Bibliothek der Hochschule einer breiteren Öffentlichkeit zugänglich gemacht wird.

   - [X] Ja
   - [ ] Nein

   Ich erkläre und stehe dafür ein, dass ich alleiniger Inhaber aller Rechte an der Bachelorarbeit, einschließlich des Verfügungsrechts über Vorlagen an beigefügten Abbildungen, Plänen o.ä., bin und durch deren öffentliche Zugänglichmachung weder Rechte und Ansprüche Dritter noch gesetzliche Bestimmungen verletzt werden.

   Deggendorf, 20. März 2013 _____

   Christopher LERCH

Bei Einverständnis des Verfassers mit einer Zugänglichmachung der Bachelorarbeit vom Betreuer auszufüllen:

3. Eine Aufnahme eines Exemplars der Bachelorarbeit in den Bestand der Bibliothek und die Ausleihe des Exemplars wird

   ☐ befürwortet
   ☐ nicht befürwortet.

   Deggendorf, _____

   (Datum) Prof. Dr. rer. nat. Stefan SCHULTE

Acknowledgements

It is a personal matter and pleasure to take the opportunity to thank all those people who made this thesis possible.

At first I would like to thank Prof. Dr.-Ing. Kai-Uwe Bletzinger for affording me the opportunity to do my bachelor's thesis at the Chair of Structural Analysis in continuation of my internship here.

Thanks goes also to Dr.-Ing. Roland Wüchner for his confidence and support from the beginning of my work here and for many interesting professional discussions during the time.

I wish to express sincere gratitude and appreciation to my supervisor, Stefan Sicklinger for his guidance, support and encouragement throughout this thesis. It would not have been possible without his efforts in providing me all those great given opportunities.

Special thanks goes to my examiner Prof. Dr. rer. nat. Stefan Schulte for his great support during my studies at the University of Applied Sciences Deggendorf, which essentially influenced my further decisions. I always appreciated his advice.

I am grateful to all other colleagues at the Chair of Structural Analysis for helpful hints, interesting professional discussions and support.

Finally I want to thank all my friends and family for backing me up and supporting me all the time.

Abstract

The basic idea for starting with this thesis work was to validate the tool EMPIRE. The Enhanced MultiPhysics Interface Research Engine, short EMPIRE, is a *N*-code coupling tool for doing multiphysic co-simulation, developed by Stefan Sicklinger and Tianyang Wang at the Chair for Structural Analysis at Technische Universität München. This idea should be implemented by doing a fluid-structure-signal co-simulation of a full-scale, turbulent flow scenario containing any flexible structure of interest.

Chosen for this purpose was NREL's Unsteady Aerodynamic Experiment in its VIth Phase, which performed wind tunnel tests on a 10 m diameter wind turbine. This was done for different reasons. No other experimental datums are available in such enormous and detailed amount for any other technical building dealing with the generation of renewable energy. In addition big interest showed up by companies working also in this field. At least the minimum case of three coupled codes, fluid, structure and gearbox/generator, is simple upgradable to four or more codes, e.g. with adding control components. Simulating exactly the NREL UAE Phase VI wind turbine enables validation of the simulation results via comparison to the available experimental data.

This part of the work on developing a turbulent, full-scale fluid-structure-signal co-simulation of NREL's UAE Phase VI wind turbine focused on the fluid side, which should be realized with the open source tool OpenFOAM. The challenge arising with this decision was to fill the lack of documentation, that comes with OpenFOAM. The best combination and all necessary settings had to be worked out. This was done stepwise, so debugging and validation was more efficient.

The extend of this tasks however was underestimated by all involved parties. So the results presented in this thesis are limited to the achievements until one-way coupling of a quasi structural solver and work will be continued.

Contents

1

Basics of Computational Fluid Dynamics

1.1 Basic Equations of Fluid Flow

1.1.1 Model Assumptions

To get the physical and mathematical model for derivation of the basic equations OpenFOAM's solvers and other computational fluid dynamics (CFD) codes are based on, different assumptions have to be made. The first one is the fluid to be regarded as continuum. That implies for analysis a macroscopic length scale is used. So molecular structures of matter, molecular motions and other molecular effects are ignored and the behavior is described by macroscopic properties like the velocity vector $\mathbf{u} = (u\ v\ w)^T$, the pressure p, the density ρ and the temperature T, which are all averages over a suitable large number of molecules. After this assumption a so called fluid particle or a point in fluid is the smallest element, which is not influenced by single molecules.

For the following analysis a infinite small control volume Ω with sides of the length δx, δy and δz, six faces and a center with the coordinates $\mathbf{x} = (x\ y\ z)^{\mathrm{T}}$ is used. It is called a fluid element and all its fluid properties are functions of space and time: $\rho := \rho(x, y, z, t)$, $p := p(x, y, z, t)$, $\mathbf{u} := \mathbf{u}(x, y, z, t)$ and

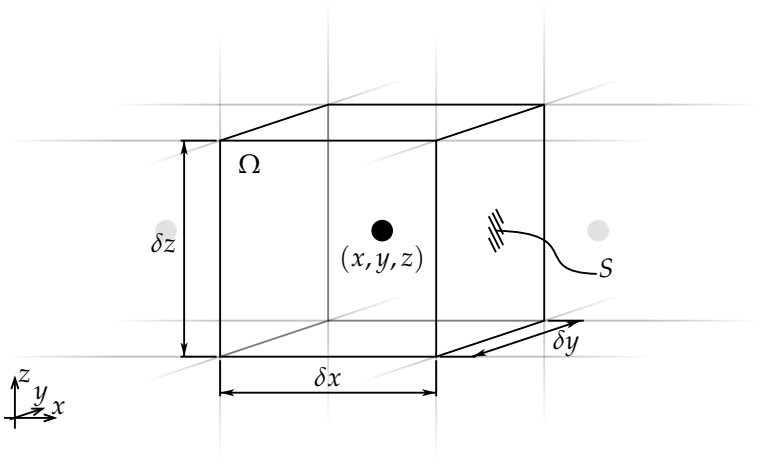


Figure 1.1: Fluid element for analysis

$T := T(x, y, z, t)$. Based on the infinite small size means of the first two terms of a Taylor series expansion are enough to describe changes of these fluid properties, e.g. pressure p:

$$\begin{aligned} p(x+\frac{1}{2}\delta x) &= p(x) + \frac{1}{1!}\frac{\partial p}{\partial x}\left((x+\frac{1}{2}\delta x) - x\right) + \frac{1}{2!}\frac{\partial^2 p}{\partial x^2}\left((x+\frac{1}{2}\delta x) - x\right)^2 \\ &\quad + \frac{1}{3!}\frac{\partial^3 p}{\partial x^3}\left((x+\frac{1}{2}\delta x) - x\right)^3 + \ldots \approx p(x) + \frac{\partial p}{\partial x}\frac{1}{2}\delta x \\ p(x-\frac{1}{2}\delta x) &= p(x) + \frac{1}{1!}\frac{\partial p}{\partial x}\left((x-\frac{1}{2}\delta x) - x\right) + \frac{1}{2!}\frac{\partial^2 p}{\partial x^2}\left((x-\frac{1}{2}\delta x) - x\right)^2 \\ &\quad + \frac{1}{3!}\frac{\partial^3 p}{\partial x^3}\left((x-\frac{1}{2}\delta x) - x\right)^3 + \ldots \approx p(x) - \frac{\partial p}{\partial x}\frac{1}{2}\delta x \end{aligned} \tag{1.1}$$

For the following steps the well known conservation laws of physics form the basis. They are:

- mass conservation (continuity equation)
- momentum conservation (Newton's second law)
- energy conservation (first law of thermodynamics)

1.1.2 Mass Conservation

As relativistic effects don't play any role, because velocities are much smaller than the speed of light ($u, v, w \ll c$), the mass of the fluid has always to be conserved. This is known as the mass conservation equation or continuity equation. In words it would read

rate of change (in-/decrease) of mass inside the fluid element	=	net rate of mass flow into/out of the fluid element

And as equation it reads

$$\frac{\partial m}{\partial t} = \sum_S \dot{m}_{\text{in}} - \sum_S \dot{m}_{\text{out}} \tag{1.2}$$

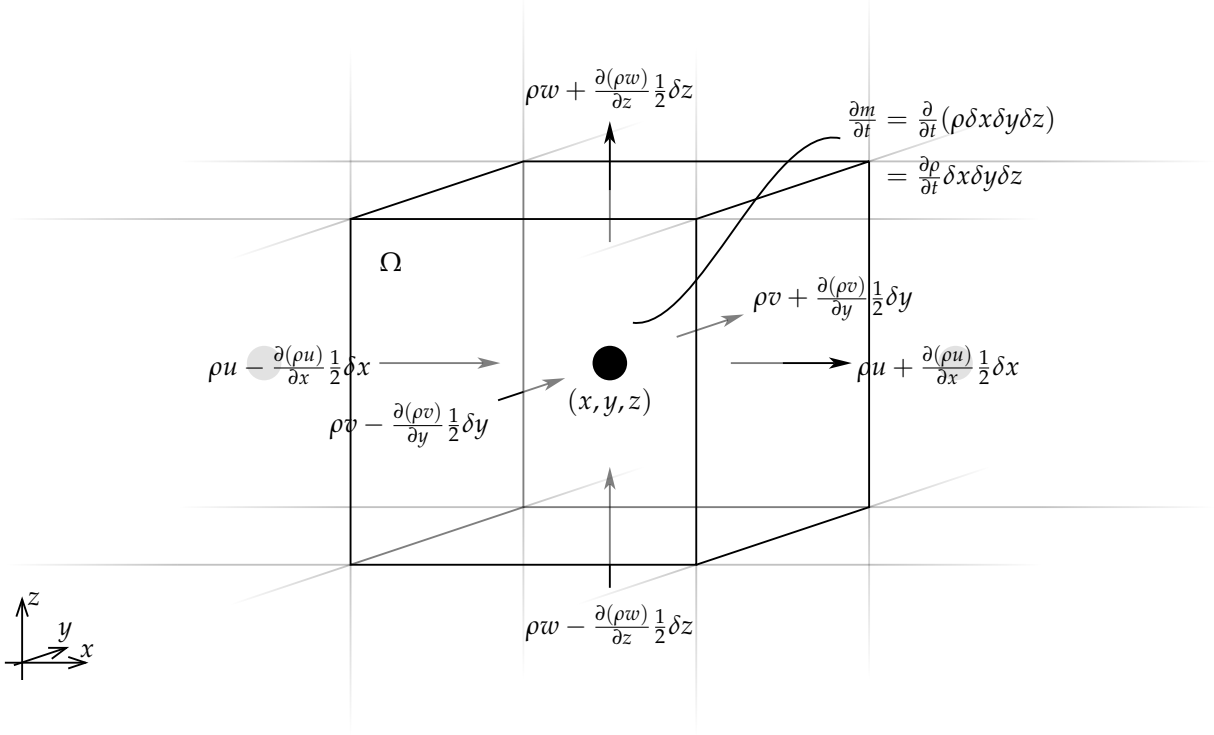


Figure 1.2: Mass flows into and out of a fluid element

All mass fluxes across the boundary S of a fluid element can be seen in Figure 1.2. Below the rate of change of mass inside this fluid element and the outgoing mass flux in x-direction are derived.

$$\frac{\partial m}{\partial t} = \frac{\partial}{\partial}(\rho \delta x \delta y \delta z) = \frac{\partial \rho}{\partial t}\delta x \delta y \delta z \tag{1.3}$$

$$\dot{m}_{\text{out},x} = \dot{m}_x + \frac{\partial \dot{m}_x}{\partial x}\frac{1}{2}\delta x = \left(\frac{\dot{m}_x}{A_x} + \frac{\partial}{\partial x}\left(\frac{\dot{m}_x}{A_x}\right)\frac{1}{2}\delta x\right)\underbrace{\delta y \delta z}_{=:A_x} = \left(\rho u + \frac{\partial(\rho u)}{\partial x}\frac{1}{2}\delta x\right)\delta y \delta z \tag{1.4}$$

By adding all mass fluxes (see equation (1.4) and Figure 1.2) and equation (1.3) into the general mass conservation equation (1.2) and doing some rearrangements you get the special mass conservation equation or continuity equation for a fluid element.

$$\begin{aligned}
\frac{\partial \rho}{\partial t}\delta x \delta y \delta z = \quad & \left(\rho u - \frac{\partial(\rho u)}{\partial x}\frac{1}{2}\delta x\right)\delta y \delta z - \left(\rho u + \frac{\partial(\rho u)}{\partial x}\frac{1}{2}\delta x\right)\delta y \delta z \\
+ \quad & \left(\rho v - \frac{\partial(\rho v)}{\partial y}\frac{1}{2}\delta y\right)\delta x \delta z - \left(\rho v + \frac{\partial(\rho v)}{\partial y}\frac{1}{2}\delta y\right)\delta x \delta z \\
+ \quad & \left(\rho w - \frac{\partial(\rho w)}{\partial z}\frac{1}{2}\delta z\right)\delta x \delta y - \left(\rho w + \frac{\partial(\rho w)}{\partial z}\frac{1}{2}\delta z\right)\delta x \delta y
\end{aligned} \tag{1.5}$$

$$\Leftrightarrow \quad \frac{\partial \rho}{\partial t} = -\frac{\partial(\rho u)}{\partial x} - \frac{\partial(\rho v)}{\partial y} - \frac{\partial(\rho w)}{\partial z} \tag{1.6}$$

$$\Leftrightarrow \quad \frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0 \quad \hat{=} \quad \underbrace{\frac{\partial \rho}{\partial t}}_{*)} + \underbrace{\nabla \bullet (\rho \mathbf{u})}_{**)} = 0 \quad \hat{=} \quad \frac{\mathrm{D}\rho}{\mathrm{D}t} = 0 \tag{1.7}$$

Term *) represents the rate of change of density (mass per unit volume) inside the fluid element and term **) the net flow of mass per unit volume across the boundary of the fluid element. Term **) is called the convective term.

Summary 1.1 *(Mass conservation)*

Unsteady, three-dimensional mass conservation equation or continuity equation for a fluid element in a *compressible* fluid

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho \mathbf{u}) = 0 \tag{1.8}$$

Unsteady, three-dimensional mass conservation equation or continuity equation for a fluid element in a *incompressible* fluid

$$\nabla \bullet \mathbf{u} = 0 \tag{1.9}$$

Hence $\mathbf{u}$ needs to be a divergence free vector field for the incompressible Navier-Stokes Equations.

1.1.3 Momentum Conservation

There are two possible points of view for analyzing the flow of fluids. One is the so called Langrangian approach, where fluid particles are followed on their way. So each property of such a particle is a function of the actual position $\mathbf{x} = (x\ y\ z)^{\mathrm{T}}$ and time t. The development of numerical methods for this approach can be done, but it is not common. Instead the the Eulerian approach is used. Here a fixed region made out of fluid elements is used to develop numerical equations.

The momentum conservation equation respectively Newton's second law in words reads

rate of change (in-/decrease) of momentum of the fluid inside the fluid element	=	sum of forces acting on the fluid inside the fluid element

The rate of change of x-, y- and z-momentum per unit volume of the fluid inside a fluid element equals the total, substantive or material derivation of the momentum per unit volume with respect to time.

$$\frac{\mathrm{D}(\rho u)}{\mathrm{D}t} = \rho\frac{\mathrm{D}u}{\mathrm{D}t} + u \underbrace{\frac{\mathrm{D}\rho}{\mathrm{D}t}}_{=0, \text{ see eq. (1.7)}} = \rho\frac{\mathrm{D}u}{\mathrm{D}t} = \rho\left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial u}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} + \frac{\partial u}{\partial z}\frac{\mathrm{d}z}{\mathrm{d}t}\right) = \dots \tag{1.10}$$

Knowing, that the fluid particles follow the flow, it is $\frac{\mathrm{d}x}{\mathrm{d}t} = u$, $\frac{\mathrm{d}y}{\mathrm{d}t} = v$ and $\frac{\mathrm{d}z}{\mathrm{d}t} = w$.

$$\dots = \rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}\right) = \rho\left(\frac{\partial u}{\partial t} + \mathbf{u} \bullet \nabla u\right) \tag{1.11}$$

There are different types of forces acting on fluid particles. Surface forces, more exact pressure and viscous forces appear explicitly as separate terms in the momentum conservation equations, while body forces, like gravity, centrifugal, Coriolis and electromagnetic forces, are included as source terms.

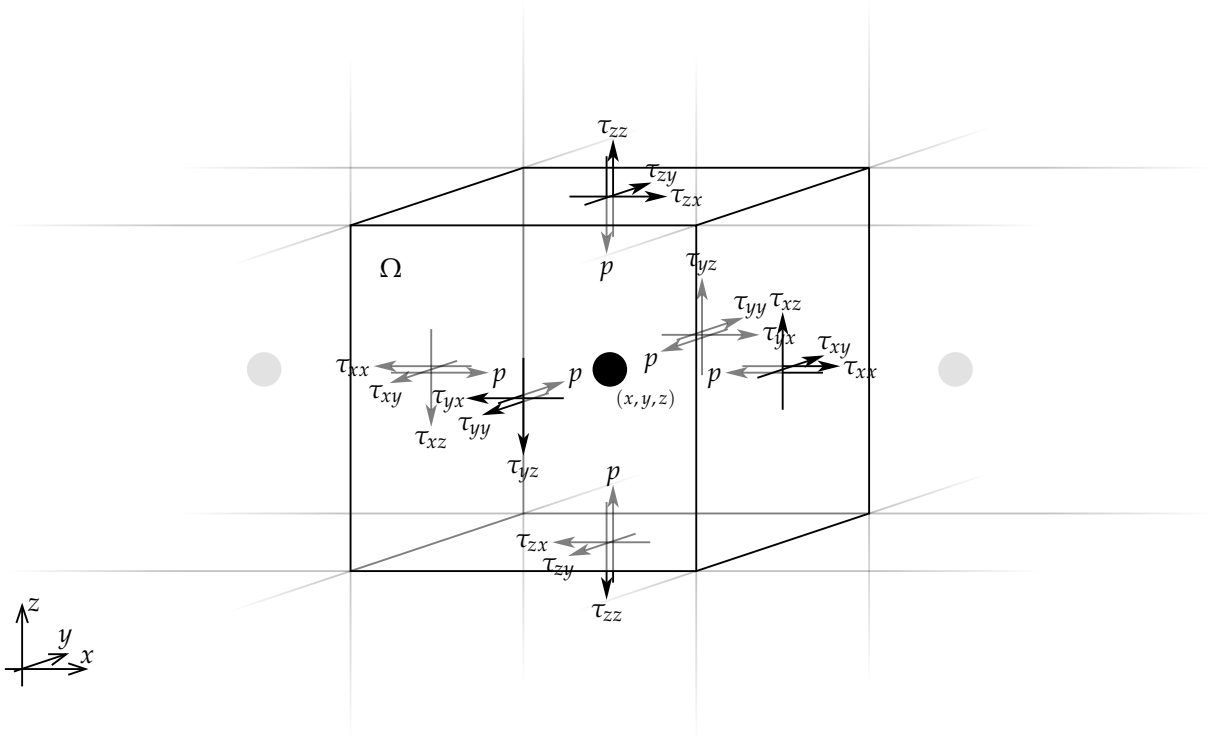


Figure 1.3: Stress components on three faces of a fluid element

The state of stress of a fluid element defined in terms of pressure p, as a normal stress towards the fluid element, and viscous stresses τ_{ij} is shown in Figure 1.3. With this notation i points in the direction of the surface normal and j in the direction of the stress component.

In the following the momentum conservation equation for the x-direction will be exemplary derived. For y- and z-direction only results will be shown, because the procedure of deriving can easily be trans-

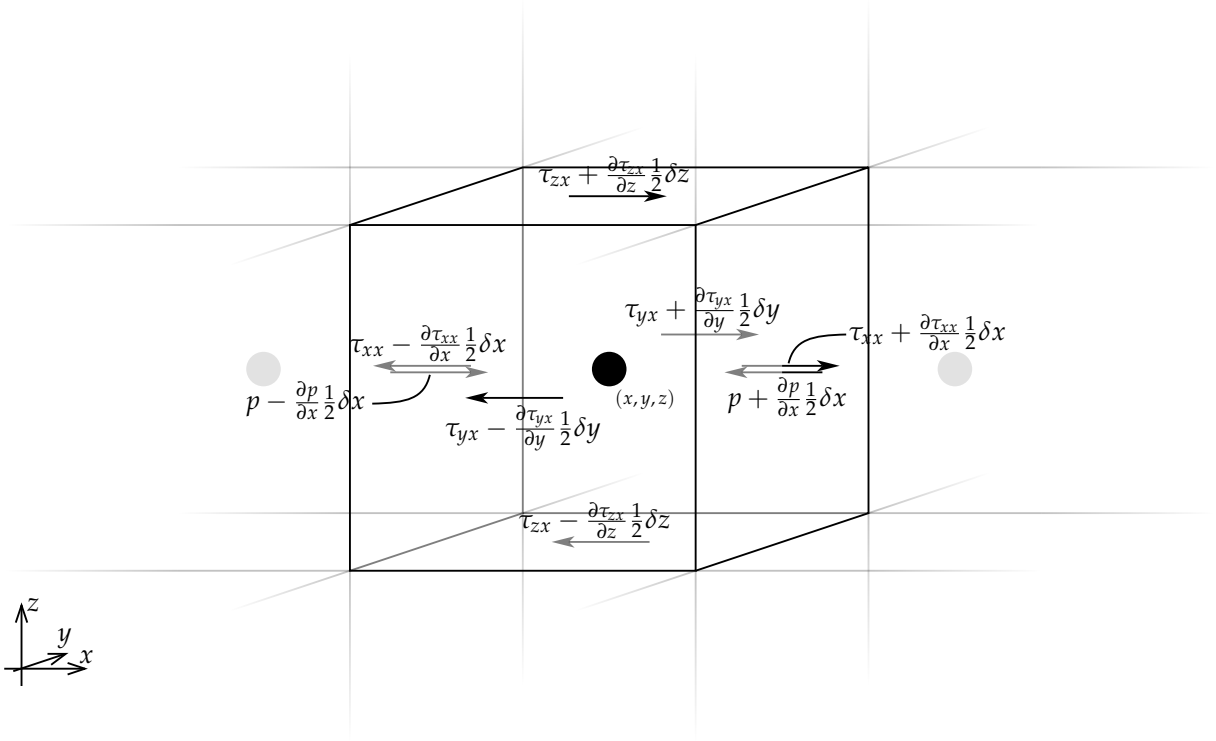


Figure 1.4: Stress components of a fluid element in x-direction

fered from the x-direction. All stresses acting on a fluid element in this direction can be seen in Figure 1.4.

Summation over the left and right face results in

$$\begin{aligned}\left[\left(p - \frac{\partial p}{\partial x}\frac{1}{2}\delta x\right) - \left(\tau_{xx} - \frac{\partial \tau_{xx}}{\partial x}\frac{1}{2}\delta x\right) - \left(p + \frac{\partial p}{\partial x}\frac{1}{2}\delta x\right) + \left(\tau_{xx} + \frac{\partial \tau_{xx}}{\partial x}\frac{1}{2}\delta x\right)\right]\delta y\delta z \\ = \left(-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x}\right)\delta x\delta y\delta z\end{aligned} \tag{1.12}$$

Summation over the front and back face

$$\left[-\left(\tau_{yx} - \frac{\partial \tau_{yx}}{\partial y}\frac{1}{2}\delta y\right) + \left(\tau_{yx} + \frac{\partial \tau_{yx}}{\partial y}\frac{1}{2}\delta y\right)\right]\delta x\delta z = \frac{\partial \tau_{yx}}{\partial y}\delta x\delta y\delta z \tag{1.13}$$

Summation over the upper and lower face

$$\left[-\left(\tau_{zx} - \frac{\partial \tau_{zx}}{\partial z}\frac{1}{2}\delta z\right) + \left(\tau_{zx} + \frac{\partial \tau_{zx}}{\partial z}\frac{1}{2}\delta z\right)\right]\delta x\delta y = \frac{\partial \tau_{zx}}{\partial z}\delta x\delta y\delta z \tag{1.14}$$

Summation over all six faces respectively summation of equations (1.12), (1.13) and (1.14) provides the sum of all surface forces acting on the fluid element.

$$-\frac{\partial p}{\partial x}+\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}+\frac{\partial \tau_{zx}}{\partial z}=-\frac{\partial p}{\partial x}+\nabla \bullet \underbrace{\begin{pmatrix}\tau_{xx} & \tau_{yx} & \tau_{zx}\end{pmatrix}^{T}}_{=:\boldsymbol{\tau}_x} \tag{1.15}$$

Body forces acting in the x-direction are accounted by defining a source term $S_{\mathrm{M}x}$ of x-momentum (Mx) per unit volume and unit time ($[S_{\mathrm{M}x}]=\frac{\mathrm{N}}{\mathrm{m^3 s}}$).

The x-component of the momentum conservation equations comes up with setting the rate of change (equation (1.11)) equal to the sum of surface forces (equation (1.15)) and source term $S_{\mathrm{M}x}$.

$$\rho\frac{\mathrm{D}u}{\mathrm{D}t}=-\frac{\partial p}{\partial x}+\nabla \bullet \boldsymbol{\tau}_x+S_{\mathrm{M}x} \;\hat{=}\; \rho\left(\frac{\partial u}{\partial t}+\mathbf{u}\bullet\nabla u\right)=-\frac{\partial p}{\partial x}+\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}+\frac{\partial \tau_{zx}}{\partial z}+S_{\mathrm{M}x} \tag{1.16}$$

The same way the y-component and the z-component of the momentum conservation equations read

$$\begin{aligned}\rho\frac{\mathrm{D}v}{\mathrm{D}t}&=-\frac{\partial p}{\partial y}+\nabla \bullet \boldsymbol{\tau}_y+S_{\mathrm{M}y} \;\hat{=}\; \rho\left(\frac{\partial v}{\partial t}+\mathbf{u}\bullet\nabla v\right)=-\frac{\partial p}{\partial y}+\frac{\partial \tau_{xy}}{\partial x}+\frac{\partial \tau_{yy}}{\partial y}+\frac{\partial \tau_{zy}}{\partial z}+S_{\mathrm{M}y}\\ \rho\frac{\mathrm{D}w}{\mathrm{D}t}&=-\frac{\partial p}{\partial z}+\nabla \bullet \boldsymbol{\tau}_z+S_{\mathrm{M}z} \;\hat{=}\; \rho\left(\frac{\partial w}{\partial t}+\mathbf{u}\bullet\nabla w\right)=-\frac{\partial p}{\partial z}+\frac{\partial \tau_{xz}}{\partial x}+\frac{\partial \tau_{yz}}{\partial y}+\frac{\partial \tau_{zz}}{\partial z}+S_{\mathrm{M}z}\end{aligned} \tag{1.17}$$

Equation (1.16) and the equations of (1.17) can be assambled to one equation.

$$\begin{aligned}\rho\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t}&=-\nabla p+\nabla\bullet\mathbf{T}+\mathbf{S}_{\mathrm{M}} \;\hat{=}\; \rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\bullet\nabla\mathbf{u}\right)=-\nabla p+\nabla\bullet\mathbf{T}+\mathbf{S}_{\mathrm{M}}\\ &\hat{=}\; \rho\begin{pmatrix}\frac{\partial u}{\partial t}+\mathbf{u}\bullet\nabla u\\ \frac{\partial v}{\partial t}+\mathbf{u}\bullet\nabla v\\ \frac{\partial w}{\partial t}+\mathbf{u}\bullet\nabla w\end{pmatrix}=-\begin{pmatrix}\frac{\partial p}{\partial x}\\ \frac{\partial p}{\partial y}\\ \frac{\partial p}{\partial z}\end{pmatrix}+\begin{pmatrix}\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}+\frac{\partial \tau_{zx}}{\partial z}\\ \frac{\partial \tau_{xy}}{\partial x}+\frac{\partial \tau_{yy}}{\partial y}+\frac{\partial \tau_{zy}}{\partial z}\\ \frac{\partial \tau_{xz}}{\partial x}+\frac{\partial \tau_{yz}}{\partial y}+\frac{\partial \tau_{zz}}{\partial z}\end{pmatrix}+\begin{pmatrix}S_{\mathrm{M}x}\\ S_{\mathrm{M}y}\\ S_{\mathrm{M}z}\end{pmatrix}\end{aligned} \tag{1.18}$$

Where $\mathbf{T}=(\boldsymbol{\tau}_x\ \boldsymbol{\tau}_y\ \boldsymbol{\tau}_z)=\begin{pmatrix}\tau_{xx} & \tau_{xy} & \tau_{xz}\\ \tau_{yx} & \tau_{yy} & \tau_{yz}\\ \tau_{zx} & \tau_{zy} & \tau_{zz}\end{pmatrix}$ and $\mathbf{S}_{\mathrm{M}}$ could for example be gravity in negative z-direction $\mathbf{S}_{\mathrm{M}}=(S_{\mathrm{M}x}\ S_{\mathrm{M}y}\ S_{\mathrm{M}z})^{\mathrm{T}}=(0\,0\ -\rho g)^{\mathrm{T}}$.

Summary 1.2 *(Momentum conservation)*

Momentum conservation equation in x-direction for a fluid element

$$\rho\left(\frac{\partial u}{\partial t}+\mathbf{u}\bullet\nabla u\right)=-\frac{\partial p}{\partial x}+\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}+\frac{\partial \tau_{zx}}{\partial z}+S_{\mathrm{M}x} \tag{1.19}$$

Momentum conservation equation in y-direction for a fluid element

$$\rho\left(\frac{\partial v}{\partial t}+\mathbf{u}\bullet\nabla v\right)=-\frac{\partial p}{\partial y}+\frac{\partial \tau_{xy}}{\partial x}+\frac{\partial \tau_{yy}}{\partial y}+\frac{\partial \tau_{zy}}{\partial z}+S_{\mathrm{M}y} \tag{1.20}$$

Momentum conservation equation in z-direction for a fluid element

$$\rho\left(\frac{\partial w}{\partial t}+\mathbf{u}\bullet\nabla w\right)=-\frac{\partial p}{\partial z}+\frac{\partial \tau_{xz}}{\partial x}+\frac{\partial \tau_{yz}}{\partial y}+\frac{\partial \tau_{zz}}{\partial z}+S_{\mathrm{M}z} \tag{1.21}$$

Three-dimensional momentum conservation equation for a fluid element

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\bullet\nabla \mathbf{u}\right)=-\nabla p+\nabla\bullet\mathbf{T}+\mathbf{S}_{\mathrm{M}} \tag{1.22}$$

1.1.4 Navier–Stokes Equations for a Newtonian Fluid

The above derived momentum conservation equations contain further unknowns: the nine viscous stress components τ_{ij}. In order to get a solvable system a suitable model is required.

In most fluids the viscous stresses are functions of the local deformation rate or strain rate. A general suitable and therefore basically used model is the one of a Newtonian fluid. Here a linear relationship between viscous stresses and local deformation rate exists. More precise this local deformation rate is the sum of the linear and the volumetric deformation rate. The fluid considered also to be isotropic, six of this nine stress components are independent respectively the stress tensor $\mathbf{T}$ is symmetric.

$$\begin{aligned}
\tau_{xx}&=2\mu s_{xx}+\lambda(\nabla\bullet\mathbf{u})=2\mu\frac{\partial u}{\partial x}+\lambda\nabla\bullet\mathbf{u} \qquad \tau_{xy}=\mu s_{xy} \quad = \quad \tau_{yx}=\mu s_{yx}=\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\\
\tau_{yy}&=2\mu s_{yy}+\lambda(\nabla\bullet\mathbf{u})=2\mu\frac{\partial v}{\partial y}+\lambda\nabla\bullet\mathbf{u} \qquad \tau_{xz}=\mu s_{xz} \quad = \quad \tau_{zx}=\mu s_{zx}=\mu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\\
\tau_{zz}&=2\mu s_{zz}+\lambda(\nabla\bullet\mathbf{u})=2\mu\frac{\partial w}{\partial z}+\lambda\nabla\bullet\mathbf{u} \qquad \tau_{yz}=\mu s_{yz} \quad \underbrace{=}_{\text{isotropic fluid}} \quad \tau_{zy}=\mu s_{zy}=\mu\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)
\end{aligned} \tag{1.23}$$

Here μ is the dynamic viscosity, λ is the bulk viscosity, s_{ij} are the linear deformation rates and $\nabla\bullet\mathbf{u}$ is the volumetric deformation rate.

Exemplary the derivation for the x-component of the momentum conservation equations is shown.

$$\begin{aligned}\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}+\frac{\partial \tau_{zx}}{\partial z} &= \frac{\partial}{\partial x}\left[2\mu\frac{\partial u}{\partial x}+\lambda\nabla\bullet\mathbf{u}\right]+\frac{\partial}{\partial y}\left[\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[\mu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right]\\ &= \mu\frac{\partial^2 u}{\partial x^2}+\mu\frac{\partial^2 u}{\partial y^2}+\mu\frac{\partial^2 u}{\partial z^2}+\left[\mu\frac{\partial^2 u}{\partial x\partial x}+\mu\frac{\partial^2 v}{\partial x\partial y}+\mu\frac{\partial^2 w}{\partial x\partial z}+\frac{\partial}{\partial x}(\lambda\nabla\bullet\mathbf{u})\right]\\ &= \mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+\left[\frac{\partial}{\partial x}\left(\mu\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)\right)+\frac{\partial}{\partial x}(\lambda\nabla\bullet\mathbf{u})\right]\\ &= \mu\underbrace{\nabla\bullet\nabla u}_{=\Delta u}+\left[\frac{\partial}{\partial x}\left((\mu+\lambda)\underbrace{\nabla\bullet\mathbf{u}}_{\substack{=0\text{ for incom-}\\ \text{pressible fluids}}}\right)\right]\end{aligned} \tag{1.24}$$

Applying equation (1.24) to the momentum conservation equation in x-direction (1.19) comes to the x-component of the so-called Navier-Stokes equations.

$$\begin{aligned}\rho\left(\frac{\partial u}{\partial t}+\mathbf{u}\bullet\nabla u\right) &= -\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+\underbrace{\left[\frac{\partial}{\partial x}\left((\mu+\lambda)\nabla\bullet\mathbf{u}\right)\right]+S_{\mathrm{M}x}}_{=:S^*_{\mathrm{M}x}}\\ &\hat{=}\quad \rho\frac{\mathrm{D}u}{\mathrm{D}t}=-\frac{\partial p}{\partial x}+\mu\nabla\bullet\nabla u+S^*_{\mathrm{M}x}\end{aligned} \tag{1.25}$$

Similar the y-component

$$\begin{aligned}\rho\left(\frac{\partial v}{\partial t}+\mathbf{u}\bullet\nabla v\right) &= -\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)+\underbrace{\left[\frac{\partial}{\partial y}\left((\mu+\lambda)\nabla\bullet\mathbf{u}\right)\right]+S_{\mathrm{M}y}}_{=:S^*_{\mathrm{M}y}}\\ &\hat{=}\quad \rho\frac{\mathrm{D}v}{\mathrm{D}t}=-\frac{\partial p}{\partial y}+\mu\nabla\bullet\nabla v+S^*_{\mathrm{M}y}\end{aligned} \tag{1.26}$$

and the z-component

$$\begin{aligned}\rho\left(\frac{\partial w}{\partial t}+\mathbf{u}\bullet\nabla w\right) &= -\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)+\underbrace{\left[\frac{\partial}{\partial z}\left((\mu+\lambda)\nabla\bullet\mathbf{u}\right)\right]+S_{\mathrm{M}z}}_{=:S^*_{\mathrm{M}z}}\\ &\hat{=}\quad \rho\frac{\mathrm{D}w}{\mathrm{D}t}=-\frac{\partial p}{\partial z}+\mu\nabla\bullet\nabla w+S^*_{\mathrm{M}z}\end{aligned} \tag{1.27}$$

can be derived. For a incompressible fluid the relation $S^*_{\mathrm{M}i}=S_{\mathrm{M}i}$ holds.

Combination of the equations (1.25), (1.26) and (1.27) gives the well-known notation of the unsteady, three-dimensional Navier-Stokes equations for Newtonian fluids.

$$\rho\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = -\nabla p + \mu\nabla\bullet\nabla\mathbf{u} + \mathbf{S}_\mathrm{M}^* \quad \hat{=} \quad \rho\left(\frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}\bullet\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla\bullet\nabla\mathbf{u} + \mathbf{S}_\mathrm{M}^*$$
$$\hat{=} \quad \rho\begin{pmatrix}\frac{\partial u}{\partial t} + \mathbf{u}\bullet\nabla u\\ \frac{\partial v}{\partial t} + \mathbf{u}\bullet\nabla v\\ \frac{\partial w}{\partial t} + \mathbf{u}\bullet\nabla w\end{pmatrix} = -\begin{pmatrix}\frac{\partial p}{\partial x}\\ \frac{\partial p}{\partial y}\\ \frac{\partial p}{\partial z}\end{pmatrix} + \begin{pmatrix}\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\\ \frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\\ \frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\end{pmatrix} + \begin{pmatrix}S_{\mathrm{M}x}^*\\ S_{\mathrm{M}y}^*\\ S_{\mathrm{M}z}^*\end{pmatrix} \tag{1.28}$$

Providing an incompressible fluid ($\rho = \text{const.}$) there now exist four equations for a system containing four unknowns (p, u, v, w). So this system is mathematically closed and solvable in combination with appropriate initial and boundary conditions.

Summary 1.3 *(Navier-Stokes equations for a Newtonian fluid)*

Navier-Stokes equation for a Newtonian fluid in x-direction

$$\rho\left(\frac{\partial u}{\partial t} + \mathbf{u}\bullet\nabla u\right) = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right) + S_{\mathrm{M}x}^* \tag{1.29}$$

Navier-Stokes equation for a Newtonian fluid in y-direction

$$\rho\left(\frac{\partial v}{\partial t} + \mathbf{u}\bullet\nabla v\right) = -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right) + S_{\mathrm{M}y}^* \tag{1.30}$$

Navier-Stokes equation for a Newtonian fluid in z-direction

$$\rho\left(\frac{\partial w}{\partial t} + \mathbf{u}\bullet\nabla w\right) = -\frac{\partial p}{\partial z} + \mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right) + S_{\mathrm{M}z}^* \tag{1.31}$$

Three-dimensional Navier-Stokes equation for a Newtonian fluid with short explanation of all terms

$$\underbrace{\rho\Bigg(\underbrace{\frac{\partial\mathbf{u}}{\partial t}}_{\substack{\text{unsteady}\\ \text{acceleration}}} + \underbrace{\mathbf{u}\bullet\nabla\mathbf{u}}_{\substack{\text{convective}\\ \text{acceleration}}}\Bigg)}_{\text{rate of change of momentum}} = \underbrace{\underbrace{-\nabla p}_{\substack{\text{pressure}\\ \text{gradient}}} + \underbrace{\mu\nabla\bullet\nabla\mathbf{u}}_{\substack{\text{diffusion term}\\ \text{or viscosity}}}}_{\text{surface forces}} + \underbrace{\mathbf{S}_\mathrm{M}^*}_{\substack{\text{source term}\\ \text{or body forces}}} \tag{1.32}$$

1.1.5 Integral Form of the General Transport Equation

By comparing the continuity equation and the three Navier-Stokes equations it becomes obvious, that there exist significant commonalities between all equations. Introducing the general transport property Φ and the general diffusion coefficient Γ comes up with the general transport equation.

$$\frac{\mathrm{D}(\rho\Phi)}{\mathrm{D}t} = \frac{\partial(\rho\Phi)}{\partial t} + \nabla\bullet(\rho\Phi\mathbf{u}) = \nabla\bullet(\Gamma\nabla\Phi) + \mathbf{S}_\Phi \tag{1.33}$$

Set $\Phi = 1$, $\Gamma = 0$ and $S_\Phi = 0$ for getting the continuity equation and $\Phi = u, v$ or w, $\Gamma = \mu$ and $S_\Phi = S_{\mathrm{M}x,\mathrm{M}y \text{ or } \mathrm{M}z}$ for getting the Navier-Stokes equations.

Integration over a three-dimensional control volume Ω (see Figure 1.1 on page 2) gives

$$\int_\Omega \frac{\partial(\rho\Phi)}{\partial t}\ dV + \int_\Omega \nabla \bullet (\rho\Phi\mathbf{u})\ dV = \int_\Omega \nabla \bullet (\Gamma\nabla\Phi)\ dV + \int_\Omega \mathbf{S}_\Phi\ dV \tag{1.34}$$

Rewriting the convective and diffusive terms as integrals over the entier bounding surface S of the control volume by using Gauss's divergence theorem $\int_\Omega \nabla \bullet \mathbf{a}\ dV = \int_S \mathbf{n} \bullet \mathbf{a}\ dA$ leads to

$$\underbrace{\frac{\partial}{\partial t}\int_\Omega \rho\Phi\ dV}_{1)} + \underbrace{\int_S \mathbf{n} \bullet (\rho\Phi\mathbf{u})\ dA}_{2)} = \underbrace{\int_S \mathbf{n} \bullet (\Gamma\nabla\Phi)\ dA}_{3)} + \underbrace{\int_\Omega \mathbf{S}_\Phi\ dV}_{4)} \tag{1.35}$$

1) describes the rate of change of the total amount of fluid property Φ.
2) describes the net rate of change of Φ due to convection across the bounding surface S. $\mathbf{n} \bullet (\rho\Phi\mathbf{u})$ is the flux component of Φ due to fluid flow along outward normal vector $\mathbf{n}$.
3) describes the net rate of change of Φ due to diffusion across the bounding surface S. $\mathbf{n} \bullet (\Gamma\nabla\Phi)$ is the diffusion flux component of Φ along the outward normal vector $\mathbf{n}$.
4) describes the net rate of change of Φ as a result of sources (or sinks).

Final integration over a small time interval Δt, so from t to $t + \Delta t$, comes up with

$$\int_t^{t+\Delta t} \frac{\partial}{\partial t}\int_\Omega \rho\Phi\ dV\ dt + \int_t^{t+\Delta t}\int_S \mathbf{n} \bullet (\rho\Phi\mathbf{u})\ dA\ dt = \int_t^{t+\Delta t}\int_S \mathbf{n} \bullet (\Gamma\nabla\Phi)\ dA\ dt + \int_t^{t+\Delta t}\int_\Omega \mathbf{S}_\Phi\ dV\ dt \tag{1.36}$$

what can be pointed out as the starting point for computational procedures in the finit volume method.

1.2 Basic Idea of the Finite Volume Method

In the following section the basic idea of the finite volume method (FVM) is presented, which is used by OpenFOAM for its computational fluid dynamics (CFD) calculations. Over all it can be said, that the FVM is a numerical method for solving partial differential equations (PDEs). Like in other numerical methods, as a first step, the fluid domain is discretized. This means, the fluid region is decomposed into a finite number of control volumes, like the one, that can be seen in Figure 1.1 on page 2.

As a next step all conservation equations, named on page 2 and represented by the general transport equation (1.33) with the general transport property Φ, are applied to each of such control volumes. In particularly each conservation equation is integrated over Ω. So the FVM uses the integral form of a PDE as starting point, which has already been mentioned at the end of the previous section. Values are calculated at the computational nodes, located at the center of each control volume, and surface values are interpolated in terms of nodal values.

This procedure results in a linear algebraic equation containing only neighbor nodal values for each conservation quantity in each control volume. Finally the obtained linear algebraic system can be solved by using a linear solver.

The FVM is perhaps one of the simplest to understand and to program, because all appearing terms have a physical and thereby demonstrative meaning. The introduction will be done on the subsequent example of a simple one-dimensional, steady-state convection-diffusion problem for the general transport property Φ.

As already mentioned, step 1 is the discretization of the fluid domain into a finite number of control volumes. This is also known as "mesh generation", "meshing" or "grid generation". The used nomenclature can be seen in Figure 1.5: The capital I marks values at nodal points, while the small i represents values at boundary faces. In practice control volumes are chosen in a way, that near domain edges physical and control volume boundaries coincide. Figure 1.6 shows the simple, uniform mesh used in the one-dimensional example.

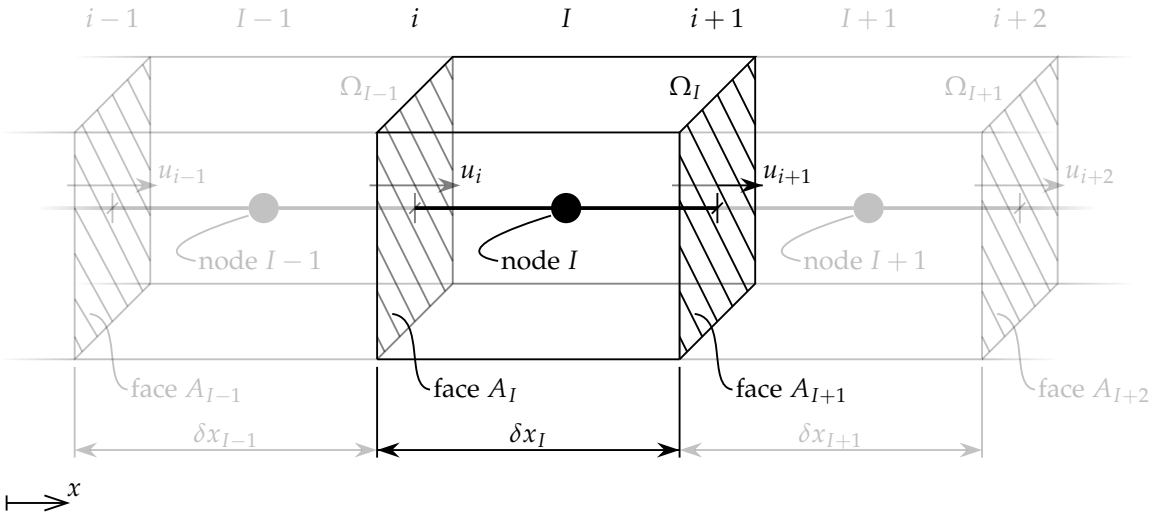


Figure 1.5: Nomenclature of a control volume in discretized, one-dimensional fluid domain

Step 2 is the discretization of the general transport equation, representative for all conservation equations. This means integration over the control volume Ω.

Starting with equation (1.36), derived on page 11 et seq.

$$\int_t^{t+\Delta t} \frac{\partial}{\partial t} \int_\Omega \rho\Phi \; dV \; dt + \int_t^{t+\Delta t} \int_S \mathbf{n} \bullet (\rho\Phi\mathbf{u}) \; dA \; dt = \int_t^{t+\Delta t} \int_S \mathbf{n} \bullet (\Gamma\nabla\Phi) \; dA \; dt + \int_t^{t+\Delta t} \int_\Omega \mathbf{S}_\Phi \; dV \; dt$$

and considering a one-dimensional domain ($v, w, \frac{\partial}{\partial x}, \frac{\partial}{\partial z}, S_{\Phi y}, S_{\Phi z} = 0$) and steady-state ($\frac{\partial}{\partial t} = 0$) leads to

$$\begin{aligned} & \int_S 1(\rho\Phi u)\ dA = \int_S 1\left(\Gamma\frac{\partial\Phi}{\partial x}\right)\ dA + \int_\Omega S_{\Phi x}\ dV \\ \Leftrightarrow \quad & \rho A u\Phi\Big|_i^{i+1} = \Gamma A\frac{\partial\Phi}{\partial x}\Big|_i^{i+1} + \overline{S}_{\Phi x}A\delta x\Big|_I \\ \Leftrightarrow \quad & \rho A u\Phi\Big|_{i+1} - \rho A u\Phi\Big|_i = \Gamma A\frac{\partial\Phi}{\partial x}\Big|_{i+1} - \Gamma A\frac{\partial\Phi}{\partial x}\Big|_i + \overline{S}_{\Phi x}A\delta x\Big|_I \end{aligned} \tag{1.37}$$

The simplest way and well established practice for interpolating the appearing boundary values by nodal point values is linear interpolation. That way gradients are expressed with the central differencing scheme

$$\frac{\partial\Phi}{\partial x}\Big|_i = \frac{\Phi_I - \Phi_{I-1}}{\delta x} \quad \text{and} \quad \frac{\partial\Phi}{\partial x}\Big|_{i+1} = \frac{\Phi_{I+1} - \Phi_I}{\delta x} \tag{1.38}$$

and boundary values – here especially on a uniform grid – become

$$\Phi_i = \frac{\Phi_{I-1} + \Phi_I}{2} \quad \text{and} \quad \Phi_{i+1} = \frac{\Phi_I + \Phi_{I+1}}{2} \tag{1.39}$$

and (e.g. the diffusion coefficient Γ)

$$\Gamma_i = \frac{\Gamma_{I-1} + \Gamma_I}{2} \quad \text{and} \quad \Gamma_{i+1} = \frac{\Gamma_I + \Gamma_{I+1}}{2} \tag{1.40}$$

Applying this to equation (1.37) comes up with a simple linear algebraic equation for node I containing only nodal values of the neighbor nodes $I-1$ and $I+1$ and $\overline{S}_{\Phi x}$, the average value of $S_{\Phi x}$ over the control volume.

$$\begin{aligned} & \rho Au\Big|_{i+1}\frac{\Phi_I + \Phi_{I+1}}{2} - \rho Au\Big|_i\frac{\Phi_{I-1} + \Phi_I}{2} = \Gamma A\Big|_{i+1}\frac{\Phi_{I+1} - \Phi_I}{\delta x} - \Gamma A\Big|_i\frac{\Phi_I - \Phi_{I-1}}{\delta x} + \overline{S}_{\Phi x}A\delta x\Big|_I \\ \Leftrightarrow \quad & \underbrace{\left[\left(-\frac{1}{2}\rho Au - \frac{\Gamma A}{\delta x}\right)\Big|_i\right]}_{=:a_{I-1}}\Phi_{I-1} + \underbrace{\left[\left(-\frac{1}{2}\rho Au + \frac{\Gamma A}{\delta x}\right)\Big|_i + \left(\frac{1}{2}\rho Au + \frac{\Gamma A}{\delta x}\right)\Big|_{i+1}\right]}_{=:a_I}\Phi_I \\ & \qquad + \underbrace{\left[\left(\frac{1}{2}\rho Au - \frac{\Gamma A}{\delta x}\right)\Big|_{i+1}\right]}_{=:a_{I+1}}\Phi_{I+1} = \underbrace{\left[\overline{S}_{\Phi x}A\delta x\Big|_I\right]}_{=:b_I} \\ \Leftrightarrow \quad & a_{I-1}\Phi_{I-1} + a_I\Phi_I + a_{I+1}\Phi_{I+1} = b_I \end{aligned} \tag{1.41}$$

It must be mentioned, that until now, it is assumed, that face velocities u_i are somehow known. More details about the significance of this assumption will be presented in the next section.

Solving the obtained equation system is step 3. In detail equation (1.41) has to be set up for all internal nodes and modified equations have to be derived from the given boundary conditions for all boundary nodes. In the example 2, 3, 4 are internal and 1, 5 are boundary nodes.

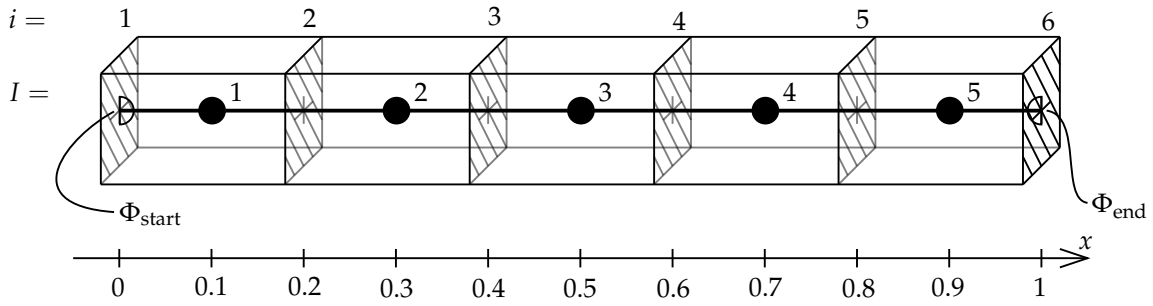


Figure 1.6: Simple, one-dimensional, uniform Mesh used in the example

To calculate a solution concrete numerical values and boundary conditions must be given. For the example the values are $\rho = 1 = \text{const.}$, $A = 4 = \text{const.}$, $u = 1 = \text{const.}$, $\overline{S}_{\Phi x} = 0.375 = \text{const.}$, $\Gamma = 0.5 = \text{const.}$, $l = 1 \Leftrightarrow \delta x = 0.2 = \text{const.}$ and the boundary conditions are $\Phi_{\text{start}} = 2$, $\Phi_{\text{end}} = 10$. With this the linear algebraic equations for internal nodes become

$$\begin{aligned}
&\text{Node 2:} \quad I = 2 \quad \rightarrow \quad a_1\Phi_1 + a_2\Phi_2 + a_3\Phi_3 = b_2 \quad \rightarrow \quad -12\Phi_1 + 20\Phi_2 - 8\Phi_3 = 0.3 \\
&\text{Node 3:} \quad I = 3 \quad \rightarrow \quad a_2\Phi_2 + a_3\Phi_3 + a_4\Phi_4 = b_3 \quad \rightarrow \quad -12\Phi_2 + 20\Phi_3 - 8\Phi_4 = 0.3 \\
&\text{Node 4:} \quad I = 4 \quad \rightarrow \quad a_3\Phi_3 + a_4\Phi_4 + a_5\Phi_5 = b_4 \quad \rightarrow \quad -12\Phi_3 + 20\Phi_4 - 8\Phi_5 = 0.3
\end{aligned} \tag{1.42}$$

Boundary conditions are applied for deriving the modified equations for the boundary nodes 1 and 5. In this example the values of the transport property Φ are explicit given at each end, so no interpolation of the surface values $\Phi_{i=1} = \Phi_{\text{start}}$ and $\Phi_{i=6} = \Phi_{\text{end}}$ is necessary and for the gradients the forward respectively backward differencing scheme is used.

$$\left.\frac{\partial\Phi}{\partial x}\right|_1 = \frac{\Phi_1 - \Phi_{\text{start}}}{\frac{1}{2}\delta x} \quad \text{and} \quad \left.\frac{\partial\Phi}{\partial x}\right|_6 = \frac{\Phi_{\text{end}} - \Phi_5}{\frac{1}{2}\delta x} \tag{1.43}$$

This implies the integrated form of the transport equation for node 1

$$\begin{aligned}
&\left.\rho Au\right|_2 \frac{\Phi_1 + \Phi_2}{2} - \left.\rho Au\right|_1 \Phi_{\text{start}} = \left.\Gamma A\right|_2 \frac{\Phi_2 - \Phi_1}{\delta x} - \left.\Gamma A\right|_1 \frac{\Phi_1 - \Phi_{\text{start}}}{\frac{1}{2}\delta x} + \left.\overline{S}_{\Phi x} A\delta x\right|_{I=1} \\
\Leftrightarrow \quad &\underbrace{\left[\left(\frac{1}{2}\rho Au + \frac{\Gamma A}{\delta x}\right)\Big|_2 + \left.\frac{2\Gamma A}{\delta x}\right|_1\right]}_{=:a_{1,\text{BC}}}\Phi_1 + \underbrace{\left[\left(\frac{1}{2}\rho Au - \frac{\Gamma A}{\delta x}\right)\Big|_2\right]}_{=:a_{2,\text{BC}}}\Phi_2 \\
&\qquad = \underbrace{\left[\left(\rho Au + \frac{2\Gamma A}{\delta x}\right)\Big|_1 \Phi_{\text{start}} + \left.\overline{S}_{\Phi x} A\delta x\right|_{I=1}\right]}_{=:b_{1,\text{BC}}} \\
\Leftrightarrow \quad &\text{Node 2:} \quad I = 1 \quad \rightarrow \quad a_{1,\text{BC}}\Phi_1 + a_{2,\text{BC}}\Phi_2 = b_{1,\text{BC}} \quad \rightarrow \quad 32\Phi_1 - 8\Phi_2 = 48.3
\end{aligned} \tag{1.44}$$

and the one for node 5

$$\begin{aligned}
& \rho Au\Big|_6 \Phi_{\text{end}} - \rho Au\Big|_5 \frac{\Phi_4+\Phi_5}{2} = \Gamma A\Big|_6 \frac{\Phi_{\text{end}}-\Phi_5}{\frac{1}{2}\delta x} - \Gamma A\Big|_5 \frac{\Phi_5-\Phi_4}{\delta x} + \overline{S}_{\Phi x} A\delta x\Big|_{I=5} \\
\Leftrightarrow \quad & \underbrace{\left[\left(-\frac{1}{2}\rho Au - \frac{\Gamma A}{\delta x}\right)\Big|_5\right]}_{=:a_{4,\text{BC}}} \Phi_4 + \underbrace{\left[\left(-\frac{1}{2}\rho Au + \frac{\Gamma A}{\delta x}\right)\Big|_5 + \frac{2\Gamma A}{\delta x}\Big|_6\right]}_{=:a_{5,\text{BC}}} \Phi_5 \\
& \qquad = \underbrace{\left[\left(-\rho Au + \frac{2\Gamma A}{\delta x}\right)\Big|_6 \Phi_{\text{end}} + \overline{S}_{\Phi x} A\delta x\Big|_{I=5}\right]}_{=:b_{5,\text{BC}}} \\
\Leftrightarrow \quad & \text{Node 5:} \quad I=5 \quad \rightarrow \quad a_{4,\text{BC}}\Phi_4 + a_{5,\text{BC}}\Phi_5 = b_{5,\text{BC}} \quad \rightarrow \quad -12\Phi_4 + 28\Phi_5 = 160.3
\end{aligned} \tag{1.45}$$

All five obtained equations for node 1 to 5 can be condensed into one linear algebraic system and e.g. solved with MATLAB or any other tool holding matrix methods.

$$\begin{pmatrix} 32 & -8 & 0 & 0 & 0 \\ -12 & 20 & -8 & 0 & 0 \\ 0 & -12 & 20 & -8 & 0 \\ 0 & 0 & -12 & 20 & -8 \\ 0 & 0 & 0 & -12 & 28 \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \end{pmatrix} = \begin{pmatrix} 48.3 \\ 0.3 \\ 0.3 \\ 0.3 \\ 160.3 \end{pmatrix} \quad \Leftrightarrow \quad \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \end{pmatrix} = \begin{pmatrix} 2.2688 \\ 3.0376 \\ 4.1534 \\ 5.7895 \\ 8.2062 \end{pmatrix} \tag{1.46}$$

In a final step the numerical solution will be compared to the analytical solution to show the accuracy of this method.

Starting again with the general transport equation (1.33) on page 10

$$\frac{\partial(\rho\Phi)}{\partial t} + \nabla \bullet (\rho\Phi\mathbf{u}) = \nabla \bullet (\Gamma\nabla\Phi) + \mathbf{S}_\Phi$$

and applying all simplifications presented on page 13 leads to a linear ordinary differential equation (ODE) of second order with constant coefficients

$$\frac{\partial}{\partial x}(\rho\Phi u) = \frac{\partial}{\partial x}\left(\Gamma\frac{\partial\Phi}{\partial x}\right) + S_{\Phi x} \quad \Leftrightarrow \quad 0 = \Gamma\frac{\partial^2\Phi}{\partial x^2} - \rho u\frac{\partial\Phi}{\partial x} + S_{\Phi x} \tag{1.47}$$

that can easily be solved with any method to get the analytical solution.

$$\begin{aligned}
\Phi(x) &= \frac{\Phi_{\text{end}} - \Phi_{\text{start}}\exp(\lambda_2 l)}{\exp(\lambda_1 l) - \exp(\lambda_2 l)}\exp(\lambda_1 x) + \frac{\Phi_{\text{start}}\exp(\lambda_1 l) - \Phi_{\text{end}}}{\exp(\lambda_1 l) - \exp(\lambda_2 l)}\exp(\lambda_2 x) \\
&\text{with} \quad \lambda_{1,2} = \frac{\rho u}{2\Gamma} \pm \sqrt{\left(\frac{\rho u}{2\Gamma}\right)^2 - \frac{S_{\Phi x}}{\Gamma}} = 1 \pm 0.5 \\
\rightarrow \quad & \Phi(x) \approx 2.3659\exp(1.5x) - 0.3659\exp(0.5x)
\end{aligned} \tag{1.48}$$

Figure 1.7 and Table 1.1 show summarized the comparison between achieved numerical results from the finite volume method and analytical results. Plotted absolute errors are simply the difference be-

tween analytical and numerical solution $e_I = \Phi(x_I) - \Phi_I$, while relative ones are same difference in relation to analytical solution $e_I^* = (\Phi(x_I) - \Phi_I)/\Phi(x_I)$.

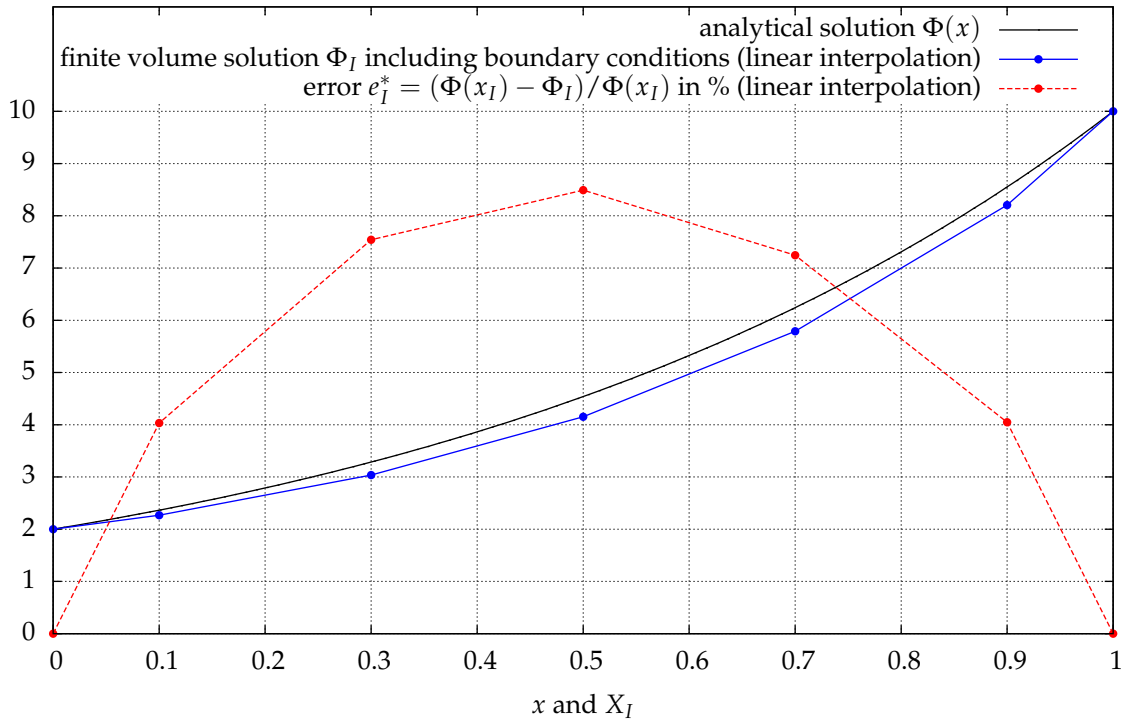


Figure 1.7: Comparison of numerical (finite volume method) and analytical results

Table 1.1: Comparison of numerical (finite volume method) and analytical results

I	x_I	Φ_I **(num.)**	$\Phi(x_I)$ **(anal.)**	**Error** e_I	**Error** e_I^* **in %**
1	0.1	2.2688	2.3641	0.0953	4.0330
2	0.3	3.0376	3.2854	0.2477	7.5408
3	0.5	4.1534	4.5388	0.3854	8.4917
4	0.7	5.7895	6.2417	0.4522	7.2446
5	0.9	8.2062	8.5525	0.3462	4.0485

Summary 1.4 *(Finite volume solution for one-dimensional, steady-state convection-diffusion problem)*

One-dimensional, steady-state convection-diffusion problem for a general transport property Φ

$$\frac{\partial}{\partial x}(\rho\Phi u) = \frac{\partial}{\partial x}\left(\Gamma\frac{\partial\Phi}{\partial x}\right) + S_{\Phi x} \tag{1.49}$$

with boundary conditions Φ_{start} and Φ_{end}

Solution by using the finite volume method

$$\begin{aligned}
&\underbrace{\left[\left(\tfrac{1}{2}\rho Au + \frac{\Gamma A}{\delta x}\right)\Big|_2 + \frac{2\Gamma A}{\delta x}\Big|_1\right]}_{=:a_{1,\text{BC}}}\Phi_1 + \underbrace{\left[\left(\tfrac{1}{2}\rho Au - \frac{\Gamma A}{\delta x}\right)\Big|_2\right]}_{=:a_{2,\text{BC}}}\Phi_2 = \underbrace{\left[\left(\rho Au + \frac{2\Gamma A}{\delta x}\right)\Big|_1\Phi_{\text{start}} + \overline{S}_{\Phi x}A\delta x\Big|_{I=1}\right]}_{=:b_{1,\text{BC}}} \\
&\underbrace{\left[\left(-\tfrac{1}{2}\rho Au - \frac{\Gamma A}{\delta x}\right)\Big|_i\right]}_{=:a_{I-1}}\Phi_{I-1} + \underbrace{\left[\left(-\tfrac{1}{2}\rho Au + \frac{\Gamma A}{\delta x}\right)\Big|_i + \left(\tfrac{1}{2}\rho Au + \frac{\Gamma A}{\delta x}\right)\Big|_{i+1}\right]}_{=:a_I}\Phi_I + \underbrace{\left[\left(\tfrac{1}{2}\rho Au - \frac{\Gamma A}{\delta x}\right)\Big|_{i+1}\right]}_{=:a_{I+1}}\Phi_{I+1} = \underbrace{\left[\overline{S}_{\Phi x}A\delta x\Big|_I\right]}_{=:b_I} \\
&\underbrace{\left[\left(-\tfrac{1}{2}\rho Au - \frac{\Gamma A}{\delta x}\right)\Big|_N\right]}_{=:a_{N-1,\text{BC}}}\Phi_{N-1} + \underbrace{\left[\left(-\tfrac{1}{2}\rho Au + \frac{\Gamma A}{\delta x}\right)\Big|_N + \frac{2\Gamma A}{\delta x}\Big|_{N+1}\right]}_{=:a_{N,\text{BC}}}\Phi_N = \underbrace{\left[\left(-\rho Au + \frac{2\Gamma A}{\delta x}\right)\Big|_{N+1}\Phi_{\text{end}} + \overline{S}_{\Phi x}A\delta x\Big|_{I=N}\right]}_{=:b_{N,\text{BC}}}
\end{aligned} \tag{1.50}$$

Short form of the linear algebraic equation system

$$\begin{aligned}
a_{1,\text{BC}}\Phi_1 + a_{2,\text{BC}}\Phi_2 &= b_{1,\text{BC}} \\
a_{I-1}\Phi_{I-1} + a_I\Phi_I + a_{I+1}\Phi_{I+1} &= b_I \\
a_{N-1,\text{BC}}\Phi_{N-1} + a_{N,\text{BC}}\Phi_N &= b_{N,\text{BC}}
\end{aligned} \tag{1.51}$$

for $i, I = 2, 3, \ldots, (N-2), (N-1)$

1.3 Handling Pressure-Velocity Linkage and Non-Linearities in the Finite Volume Method for Steady-State Flows

Using the finite volume method (FVM) for solving fluid flow equations starts with two problems. The first one appears with the velocity field $\mathbf{u}$. As mentioned in the previous section, solving the general transport equation via the FVM is only possible with assuming, that the velocity field is somehow known. In fact, this is not the case in general. Au contraire the velocity field should be a result of the overall solution process together with the other flow variables. So equations for all three velocity components u, v and w are needed. They can be obtained bei setting the general transport property Φ equal to u, v and w and are better known as the momentum conservation equations alias Navier-Stokes

equations. For three-dimensional, steady-state, laminar flow of an incompressible fluid they read in x-direction

$$\rho\left(\frac{\partial}{\partial x}(uu)+\frac{\partial}{\partial y}(vu)+\frac{\partial}{\partial z}(wu)\right)=-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+S^*_{\mathrm{M}x} \tag{1.52}$$

in y-direction

$$\rho\left(\frac{\partial}{\partial x}(uv)+\frac{\partial}{\partial y}(vv)+\frac{\partial}{\partial z}(wv)\right)=-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)+S^*_{\mathrm{M}y} \tag{1.53}$$

and in z-direction

$$\rho\left(\frac{\partial}{\partial x}(uw)+\frac{\partial}{\partial y}(vw)+\frac{\partial}{\partial z}(ww)\right)=-\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)+S^*_{\mathrm{M}z} \tag{1.54}$$

And the continuity equation reads

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{1.55}$$

Taking a closer look at equation (1.52), (1.53) and (1.54) shows the problem, that comes with the velocity field: The convective terms contain non-linearities, e.g. $\frac{\partial}{\partial x}(uu)$. And in addition the three equations are coupled, because each velocity component appears in each equation.

The pressure field p yields to the second problem, particularly its gradients. They do not appear in the general transport equation, because they are hidden in the source terms $S_{\mathrm{M}i}$. However in problems of engineering importance they form the main momentum source and so they have to be accounted for separately, e.g. $-\frac{\partial p}{\partial x}$. In general fluid flows the pressure field and so also the pressure gradients are unknown and should be part of the solution. For now there is no transport or other special equation for the pressure, but the continuity equation is still unused.

So in compressible flows the continuity equation forms the transport equation for the density ρ and the energy conservation equation the one for the temperature T. With this the pressure can be evaluated by using the equation of state $p = p(\rho, T)$ known from thermodynamics. In incompressible fluids in contrast there is by definition no linkage between p and ρ, because $\rho = \text{const.} \neq \rho(p, T)$. Here instead another constrain is used, the coupling of pressure and velocity: If the correct pressure field is applied to the momentum conservation equations, the obtained velocity field satisfies continuity. This is the basic idea used in algorithms dealing with non-linearity problems and pressure-velocity linkage.

It will now be redescribed on the most basic representative, the SIMPLE algorithm. The acronym SIMPLE is standing for Semi-Implicit Method for Pressure-Linked Equations. It is a predictor-corrector procedure for dealing with both above mentioned problems. The strategy is to evaluate the non-linear convective terms via a guessed velocity field $\hat{\mathbf{u}}$ and to use a guessed pressure field $\hat{p}$ to solve the complete momentum conservation equations. A pressure correction equation is derived evaluating the until now unused continuity equation and taken to update the pressure and velocity fields. The iteration is started

with an initial guess $\hat{\mathbf{u}}$ and $\hat{p}$ and iterated until convergence of all fields, means the fields are iteratively improved to a predefined level. Under-relaxation is used to reduce the stepwise pressure and velocity field correction to get a better stability behavior.

All steps of the SIMPLE algorithm including additional transport properties are shown in Figure 1.8.

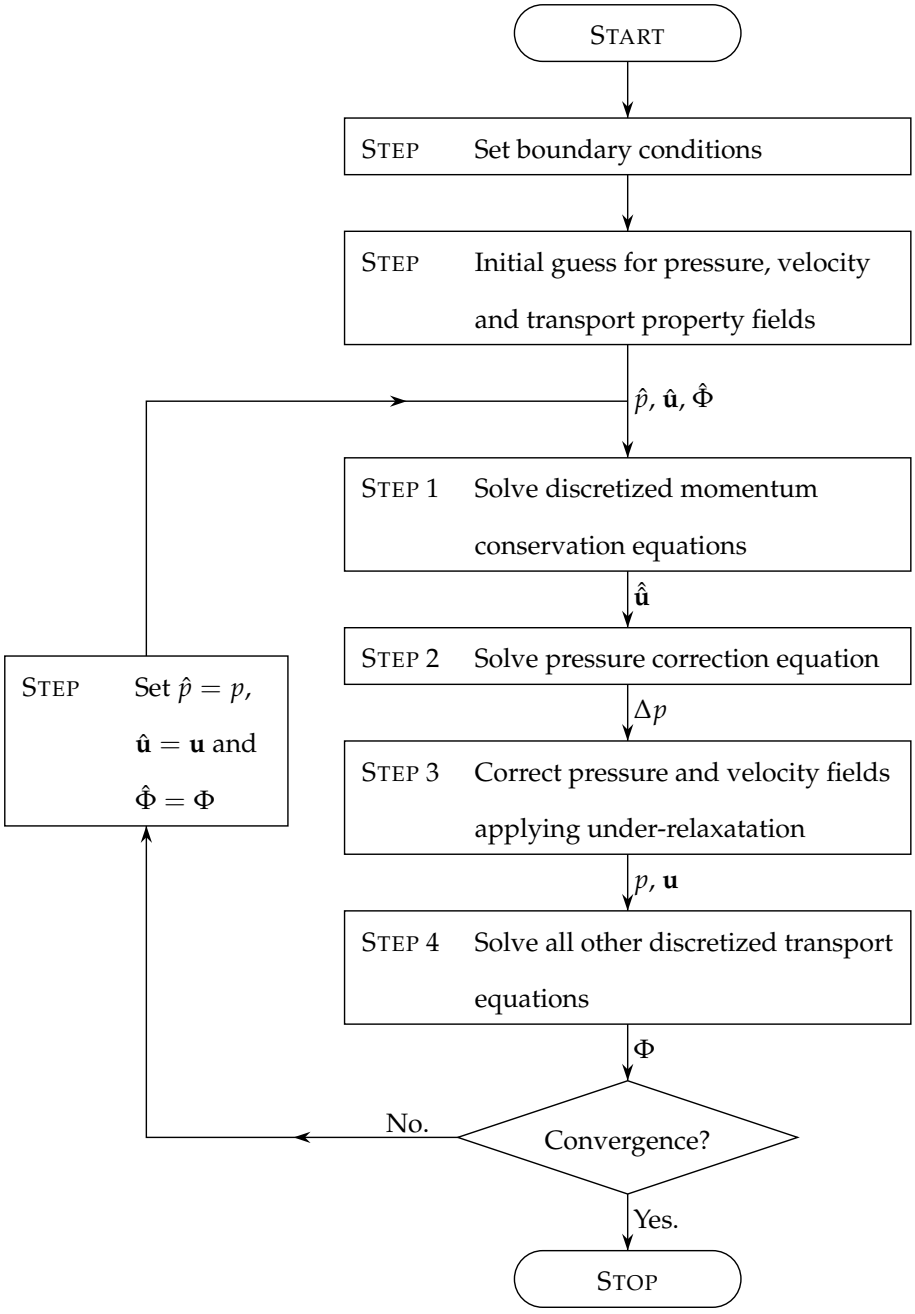


Figure 1.8: The SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) algorithm

2

NREL's Unsteady Aerodynamic Experiment Phase VI

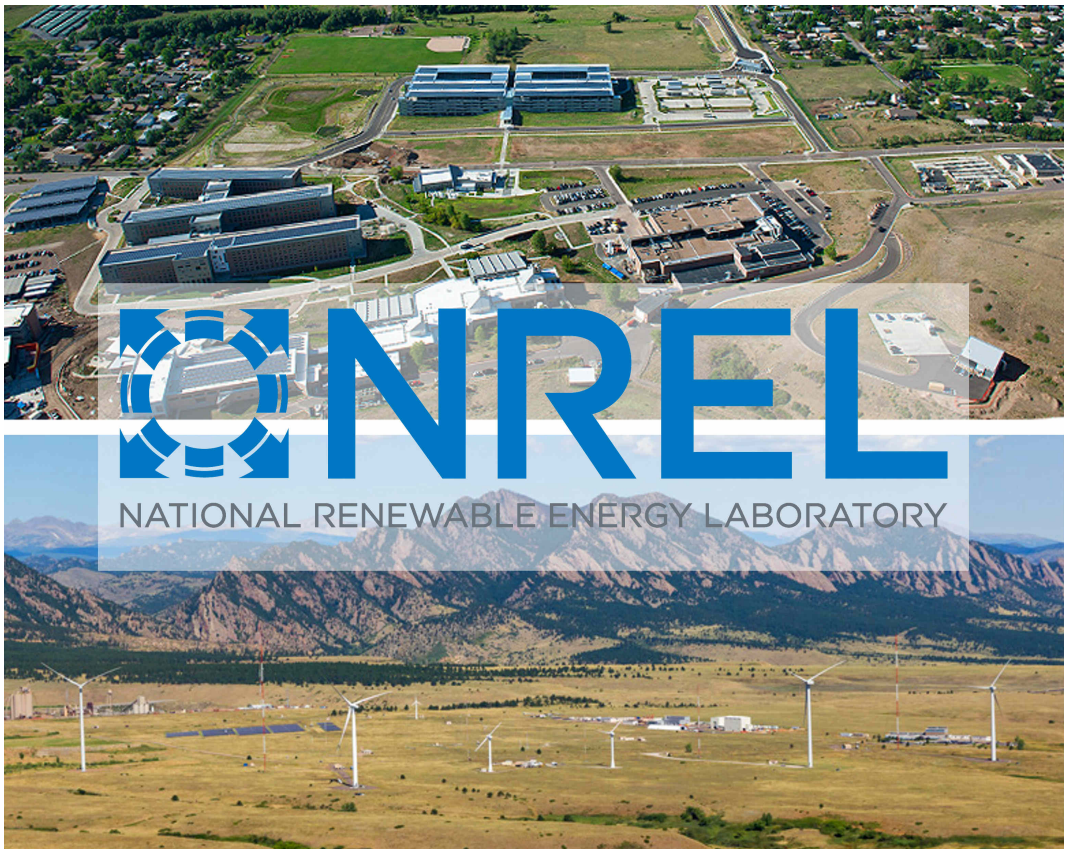


Figure 2.1: National Renewable Energy Laboratory's administrative offices and research laboratories in Golden, Colorado, NREL's logo and its National Wind Technology Center (NWTC) in Louisville, Colorado (taken from [25])

"We focus on creative answers to today's energy challenges." [25] This statement is representative of NREL's principles an main assignment. NREL signifies the National Renewable Energy Laboratory as the primary national laboratory for renewable energy and energy efficiency research and development of the U.S. Department of Energy (DOE). It acquires knowledge and innovations with the aim to achieve the U.S.'s energy and environmental goals. In other words it performs science and engineering to develop technologies for sustainability with energy. One key field of this program are investigations towards wind power generation. Therefore NREL maintains its own test area, the National Wind Technology Center (NWTC), located in Louisville, Colorado for reasons of suitable climatic conditions. All administrative offices and most research laboratories stand in Golden, Colorado. Both locations are close to Denver, Colorado. Further information can be found in [25].

The design of wind turbine blades is based on the results of steady-state, two-dimensional wind tunnel tests on airfoils. The simple expansion from steady-state flow to highly turbulent one and two-dimensions two three-dimensional blade behavior using corrections out of experience is not the answer to today's requirements. Collected data from field tests, such as NREL's Unsteady Aerodynamic Experiments (UAE) Phase I to V started in 1987 confirm this statement. The results show, that three-dimensional effects are prevalent and that wind turbines are subjected to highly dynamic load conditions caused by enormous turbulent inflow anomalies. Further information on this topic, especially to points like boundary layer theory see specialized literature.

The primary objective of NRELS's UAE was to provide information needed to quantify the full-scale, three-dimensional, aerodynamic behavior of horizontal-axis wind turbines (HAWT). Especially its phase VI was targeted at the separation of 3-D effects from the ones caused by atmospheric turbulence. Since this is not possible with field tests, it had to be realized using wind tunnel testing. For this purpose only the open-loop wind tunnel of the National Aeronautics and Space Administration (NASA) Ames Research Center's located in Moffet Field in California's Silicon Valley was able to satisfy the necessary size requirements. It can be seen in Figure 2.2 next to the yellow gantry crane. It offers a test section size of 80 ft (= 24.4 m) x 120 ft (= 36.6 m) and a variable wind speed from 0 up to 50 $\frac{\mathrm{m}}{\mathrm{s}}$ produced by 6 15-bladed fans with a performance of 16,800 kW each. A detailed description and more pictures can be found in [2], [21], [24] and [26].

A two-bladed test wind turbine with a rotor diameter of approximately 10 m was designed, built and pre-analyzed in preparation for the phase VI experiments. It was equipped with extensive measuring technique to ensure all relevant data can be recorded during the test runs and provided later on.

NREL's UAE Phase VI forms the reference in terms of geometry, settings, results etc. for the simulations developed in this thesis. The test matrix shown in [21]'s Table 1 on page 14 gives an overview of the wide range of different test scenarios treated during the wind tunnel tests. Sequence H "Upwind Baseline (F)" defines a pitch angle of $\alpha_p = 3^\circ$. Here both teeter dampers are replaced by rigid links of the same size, so consequently there were cone angles of $\alpha_{c1} = \alpha_{c3} = 0^\circ$ and a teeter angle of $\alpha_t = 0^\circ$. Since the tunnel wind speed was ranged from $u_\infty = 5\ \frac{\mathrm{m}}{\mathrm{s}}$ up to 25 $\frac{\mathrm{m}}{\mathrm{s}}$, for lower wind speeds yaw angles

Figure 2.2: 80 ft x 120 ft (24.4 m x 36.6 m) wind tunnel used in NREL's Unsteady Aerodynamic Experiment Phase VI located at NASA Ames Research Center in Moffett Field, Silicon Valley, California (taken from [33])

Figure 2.3: NREL's test wind turbine as focus of the Unsteady Aerodynamic Experiment Phase VI in NASA Ames Research Center's wind tunnel (taken from [24])

from $\alpha_y = -30°$ to $+180°$ and for higher ones from $\alpha_y = -30°$ to $+30°$ were adjusted. The rotor always operated at a nominal speed of 72 $\text{min}^{-1} = 1.2\ \text{s}^{-1} \mathrel{\hat{=}} 432\ \frac{°}{\text{s}} = 7.54\ \frac{\text{rad}}{\text{s}}$. All simulations will use the sequence H runs with lowest wind speed ($u_\infty = 5\ \frac{\text{m}}{\text{s}}$) and pure upwind configuration ($\alpha_y = 0°$) exclusively. In this case pure downwind configuration would be $\alpha_y = 180°$. Choosing the lowest available inflow velocity accounts for the wish to act in pre-stall, since stall begins with a velocity of $u_\infty = 7\ \frac{\text{m}}{\text{s}}$ [9] and requires special treatment.

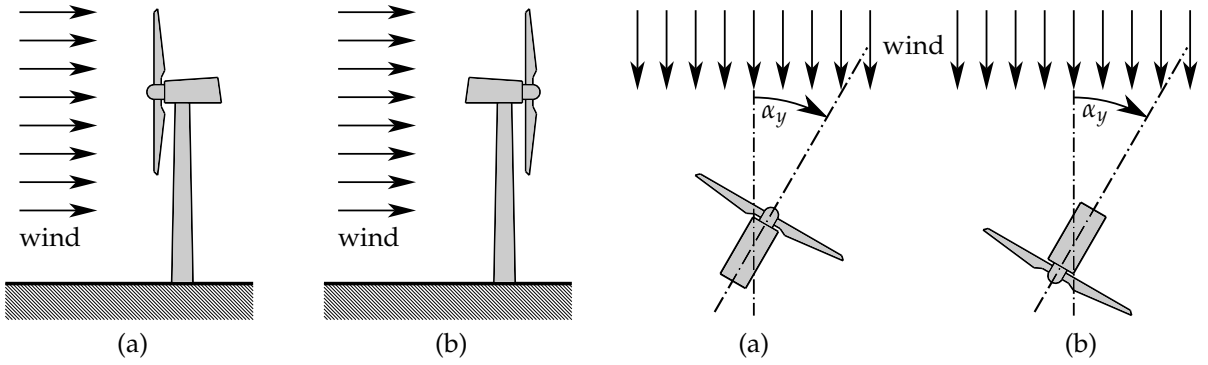


Figure 2.4: Upwind (a) and downwind (b) configuration of a wind turbine with definition of the yaw angle α_y

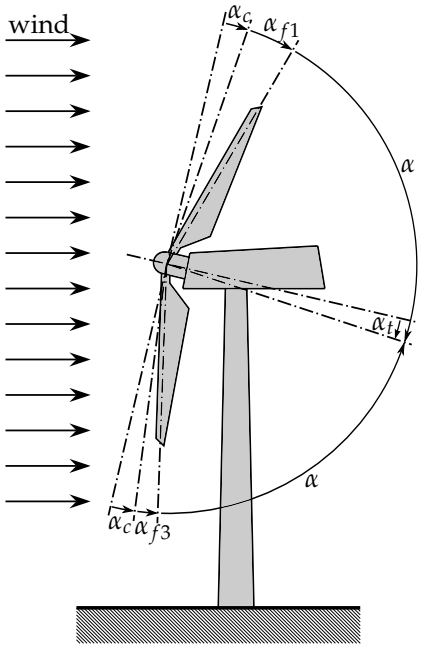


Figure 2.5: Cone angle α_c due to presetting, flap angles $\alpha_{1,3}$ due to wind load and teeter angle α_t of a wind turbine

2.1 Geometry Specifications

All listed records and websites in the bibliography dealing with NREL's UAE Phase VI do not give away any CAD data files neither for hub, nacelle, tower nor for the blades. So a first task was to extract all get-

table information setting any geometrical specifications on the wind turbine. Best source therefore was [21] with its technical drawings however containing only sparse details about physical dimensions. So a remedy was provided by setting a scale in each drawing using the real printing's dimensions and available dimensions in the drawing. At worst the same procedure was adapted to pictures from websites to get missing specifications, where no alternative option was available. All extracted and interpolated geometrical specifications, as used for generating the CAD data files, and the resulting complete geometry of the wind turbine can be seen in Figure 2.6 and 2.7.

Tower including nacelle, hub, blade 1 and 3 and additional geometries for the meshing process, which will be described in detail later on, were designed as parts in CATIA V5. Tower and rotor, the assembly of both blades and hub, were assembled to the full wind turbine geometry. In addition surface geometries were derived from all parts. So part and surface geometries are available, which can be exported in different file types, such as .stp or .stl, for use with different meshing tools. The assembly of the complete wind turbine was only useful for generating images. All parts are referenced to the same common origin. It is set in the center of the hub, where the pitch axes of the blades, meaning its longitudinal axes, and the rotation axes of the rotor are crossing. The sense of this procedure lies in simplifying the assembly and import process in meshing tools, since there is no position correction, translation or rotation, of the inserted parts required.

More effort had to be spend on designing the blades. Detailed descriptions hereto using the example of NREL's 5MW wind turbine can be found in [15] and [35], while in this thesis only a brief introduction of the procedure will be given. The central idea is to simply describe a three-dimensional blade surface geometry by specifying cross-sections through the blade normal to the blade's pitch axes and the corresponding distance (radius) to the rotational axes of the rotor. Each such cross-section is set via an airfoil profile. Since those come with a standard length of 1, origin in the leading edge and no twist (see Figure 2.9), they have to be modified to fit the blades requirements. First of all the profile is shifted along its chord line, which is the straight connection between leading and trailing edge, by the value of aerodynamic axes. Assumed to be positive therewith the new position of the origin inside the profile and ditto the pitch axes position are set. Next the profile has to be scaled using the value of the real chord length. Normally both directions are scaled with this factor except in transition areas. Here perhaps a different scaling in the non-chord direction is necessary via the parameter thickness. The last step is then to rotate the whole profile around its origin by the value of the local twist angle. Convention here is to rotate in a way the leading edge moves down and trailing edge up with a positive twist angle.

Airfoils are scheduled via a list of point coordinates for the upper and lower surfaces. So a closed curve is created via connecting all new obtained points with a spline starting and ending at the trailing edge. Concerning this sharp trailing edges require a double point at this position. At least all existing curves each lying in a different level with the distance of radius to the rotational axis, have to be connected via further splines to reach a closed three-dimensional surface. All named physical values are explained in Figure 2.8.

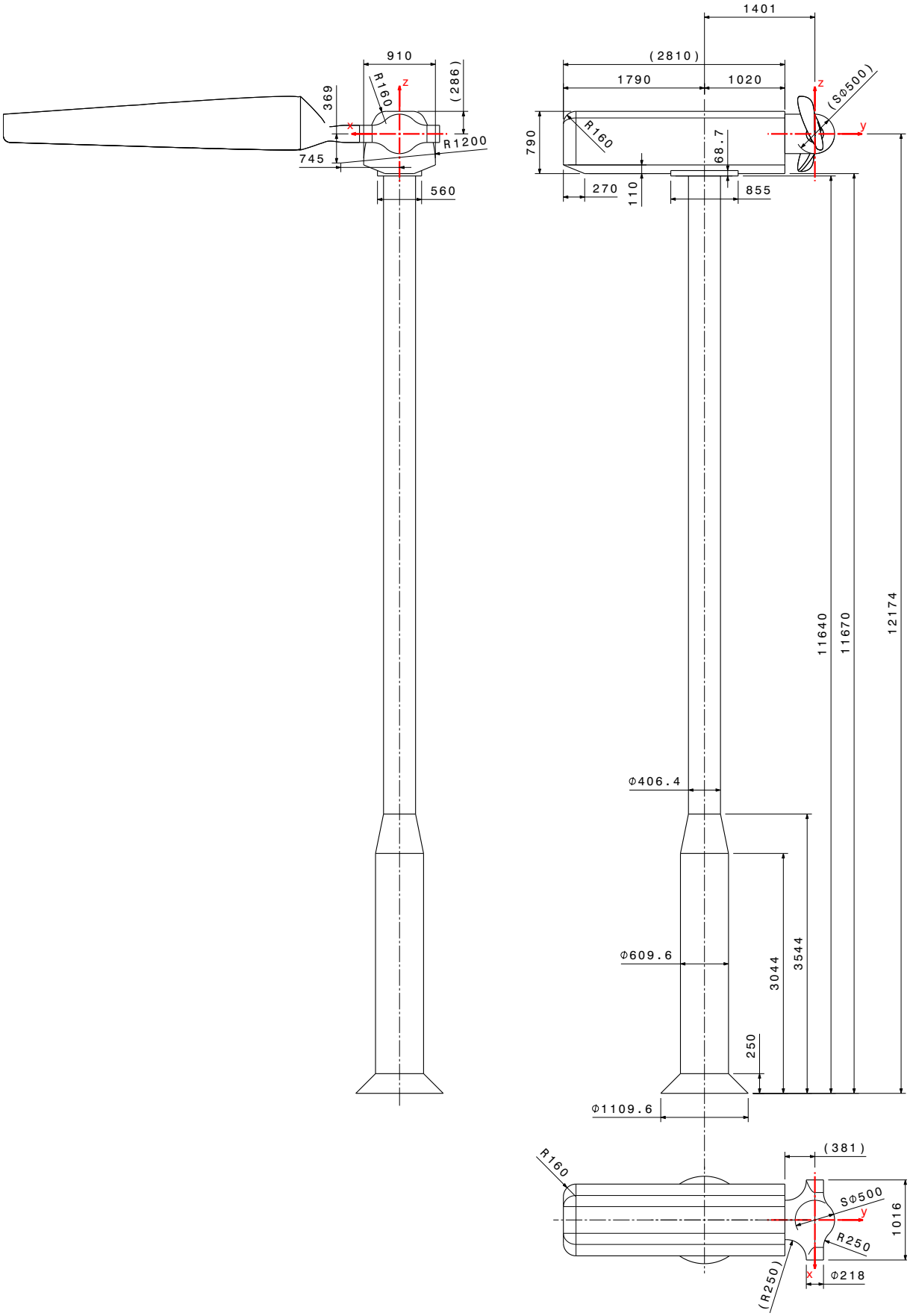


Figure 2.6: Two-dimensional views of the designed wind turbine with all extracted and approximated geometrical specifications

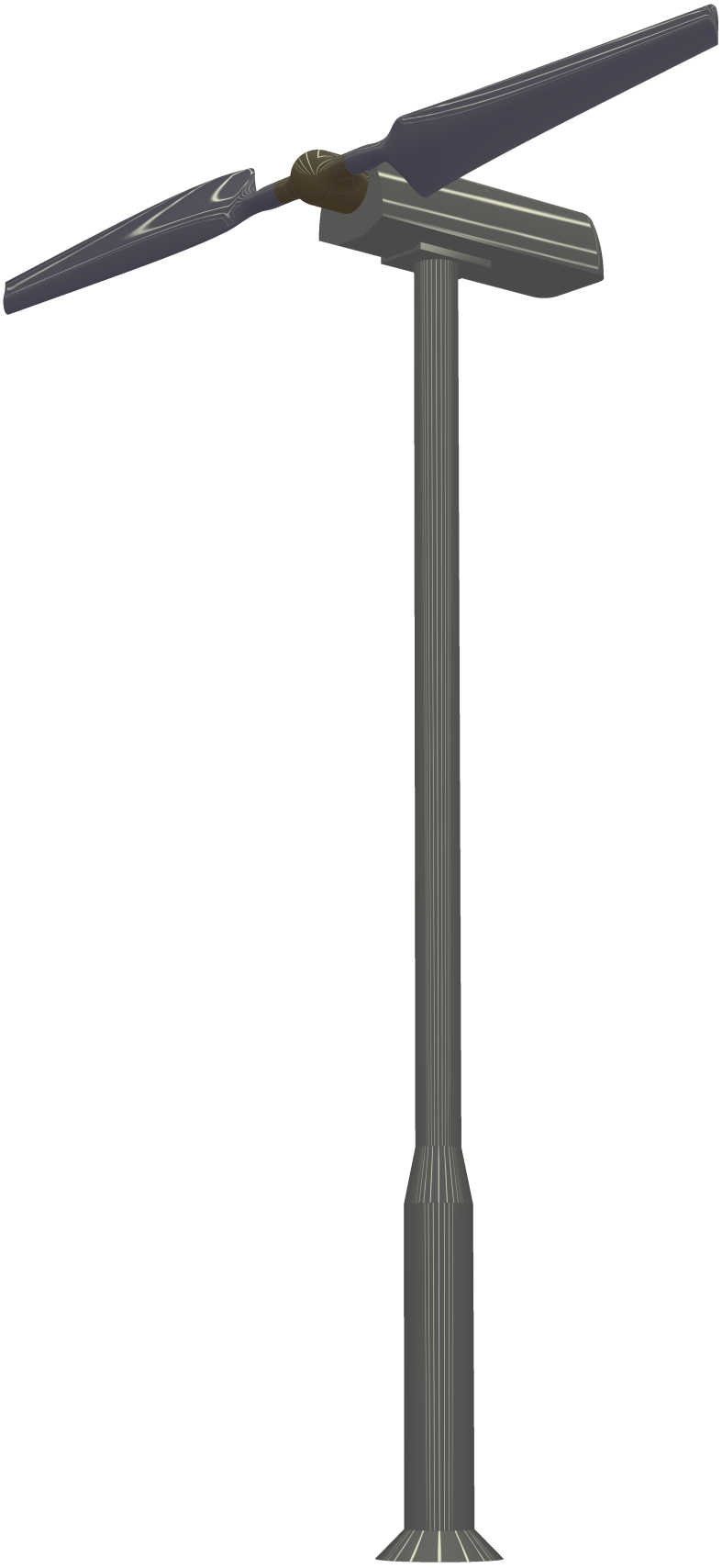


Figure 2.7: Three-dimensional view of the designed wind turbine used in all simulations

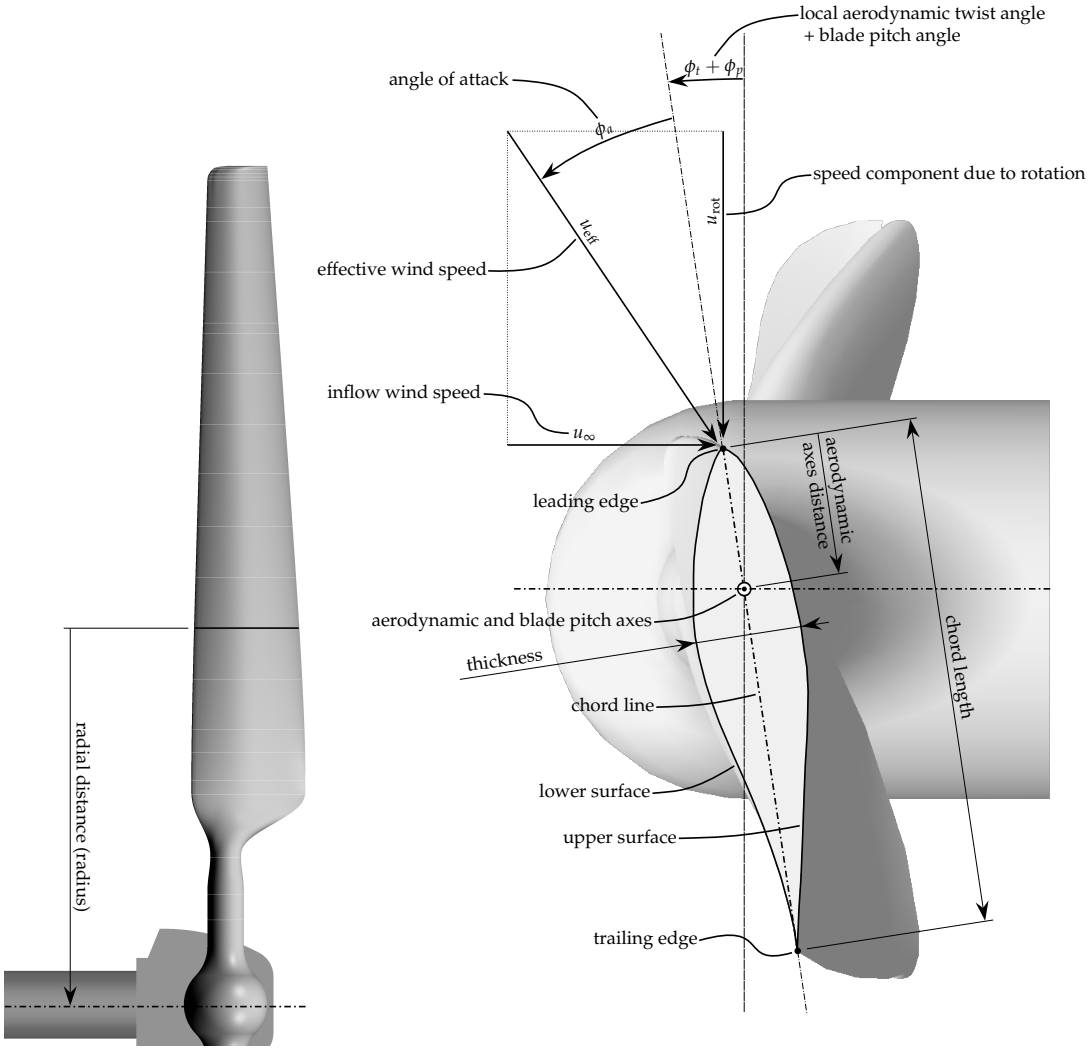


Figure 2.8: Used notation for defining the cross-sections

For NREL's UAE Phase VI wind turbine blade the S809 airfoil exclusively was used. Profile coordinate data of upper and lower surfaces are given in Table 2.1 and Figure 2.9 shows the interpolated profile shape including given points. Aerodynamic characteristics as result of wind tunnel tests on the S809 airfoil, what can be interpreted as two-dimensional flow behavior, can be referred to in Appendix A on pages 69 – 74 in [21]. Cross-section data and thus the blade geometry definition can be seen also in Table 2.1. Black and gray indicates data taken from [21], while the red data was added to define the tip cap. It was calculated using geometry specifications from [21] and improved by try-and-error to fit existing pictures of the short tip cap. Gray data was not used during the blade design process to obtain a smoother surface. The resulting profiles in CATIA V5 are shown in Figure 2.10. To get those the MATLAB script presented in Appendix A is importing S809 point data, manipulating it by using geometry specifications, both from edited Tables 2.1 (see Appendix A), and is exporting the resulting coordinate data to a Microsoft Excel file. This detour enables the option to use the Microsoft Excel macro from CATIA V5's tutorials to create points, connect them automatically via curves (splines) and make the complete surface via loft (also splines). Sometimes problems arise from the automatic curve and surface creation, which then have to be fixed manually. Most of the time loft creation fails, because of non-closed curves, different peripheral senses of the curves or other inappropriate settings. Sometimes points are too close, so curve creation fails. To add additional double endpoints, delete too close points and define splines point-by-point manually will help. Normally the macro was also only used til step 2, the automatic spline definition, since one way or another a full and no surface model was designed. Therefore it has to be switched from the Generative Shape Design environment, where the macro only works, to Part Design during the design process in CATIA.

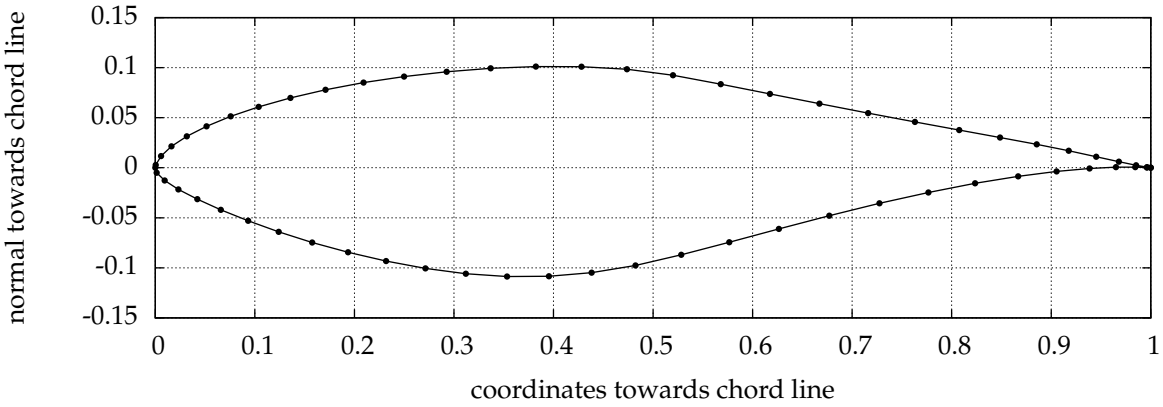


Figure 2.9: Illustrated normalized profile point data of the exclusively used S809 airfoil

For the simulations the boundary of the entire fluid domain is defined coincident with the real physical boundaries given by the wind tunnel walls, inlet and outlet. So the dimensions of the NASA Ames Research Center wind tunnel including the position of the NREL test wind turbine inside of it have to be known (see Figure 2.11).

Table 2.1: Normalized profil point data of the exclusively used S809 airfoil (left) and profile definition data for the blades of NREL's test wind turbine (right) (taken from [21])

Lower surface		Upper surface	
x/Chord in 1	*y*/Chord in 1	*x*/Chord in 1	*y*/Chord in 1
1.00000	0.00000	0.00037	0.00275
0.99612	0.00024	0.00575	0.01166
0.98446	0.00065	0.01626	0.02133
0.96509	0.00054	0.03158	0.03136
0.93852	-0.00075	0.05147	0.04143
0.90545	-0.00370	0.07568	0.05132
0.86677	-0.00859	0.10390	0.06082
0.82348	-0.01559	0.13580	0.06972
0.77668	-0.02466	0.17103	0.07786
0.72752	-0.03558	0.20920	0.08505
0.67710	-0.04792	0.24987	0.09113
0.62649	-0.06112	0.29259	0.09594
0.57663	-0.07442	0.33689	0.09933
0.52837	-0.08697	0.38223	0.10109
0.48234	-0.09756	0.42809	0.10101
0.43832	-0.10484	0.47384	0.09843
0.39541	-0.10842	0.52005	0.09237
0.35328	-0.10866	0.56801	0.08356
0.31188	-0.10589	0.61747	0.07379
0.27129	-0.10060	0.66718	0.06403
0.23175	-0.09326	0.71606	0.05462
0.19362	-0.08447	0.76314	0.04578
0.15752	-0.07467	0.80756	0.03761
0.12397	-0.06408	0.84854	0.03017
0.09325	-0.05301	0.88537	0.02335
0.06579	-0.04199	0.91763	0.01694
0.04223	-0.03144	0.94523	0.01101
0.02321	-0.02162	0.96799	0.00600
0.00933	-0.01272	0.98528	0.00245
0.00140	-0.00498	0.99623	0.00054
0.00000	0.00000	1.00000	0.00000

Node ID	Airfoil	Radius in m	Chord in m	Thickn. in m	Twist in degree	Aero axis in 1
1	Cylinder	0.508	0.218	0.218	0.0	0.50
2	Cylinder	0.660	0.218	0.218	0.0	0.50
3	Cylinder	0.883	0.183	0.183	0.0	0.50
4	Cyl./S809	1.008	0.349	0.163	6.7	0.359
5	Cyl./S809	1.067	0.441	0.154	9.9	0.335
6	Cyl./S809	1.133	0.544	0.154	13.4	0.319
7	S809	1.257	0.737	0.154	20.040	0.30
8	S809	1.343	0.728	—	18.074	0.30
9	S809	1.510	0.711	—	14.292	0.30
10	S809	1.648	0.697	—	11.909	0.30
11	S809	1.952	0.666	—	7.979	0.30
12	S809	2.257	0.636	—	5.308	0.30
13	S809	2.343	0.627	—	4.715	0.30
14	S809	2.562	0.605	—	3.425	0.30
15	S809	2.867	0.574	—	2.083	0.30
16	S809	3.172	0.543	—	1.150	0.30
17	S809	3.185	0.542	—	1.115	0.30
18	S809	3.476	0.512	—	0.494	0.30
19	S809	3.781	0.482	—	-0.015	0.30
20	S809	4.023	0.457	—	-0.381	0.30
21	S809	4.086	0.451	—	-0.475	0.30
22	S809	4.391	0.420	—	-0.920	0.30
23	S809	4.696	0.389	—	-1.352	0.30
24	S809	4.780	0.381	—	-1.469	0.30
24.1	S809	4.938000	0.365050	0.365050	-1.686288	0.300000
24.2	S809	4.960750	0.360511	0.365000	-1.717574	0.295647
24.3	S809	4.983500	0.349601	0.360000	-1.748861	0.278263
24.4	S809	5.006250	0.329376	0.300000	-1.780148	0.238827
24.5	S809	5.017625	0.312436	0.230000	-1.795791	0.200131
24.6	S809	5.029000	0.267577	0.030000	-1.811434	0.069038
25	S809	5.000	0.358	—	-1.775	0.30
26	S809	5.305	0.328	—	-2.191	0.30
27	S809	5.532	0.305	—	-2.500	0.30

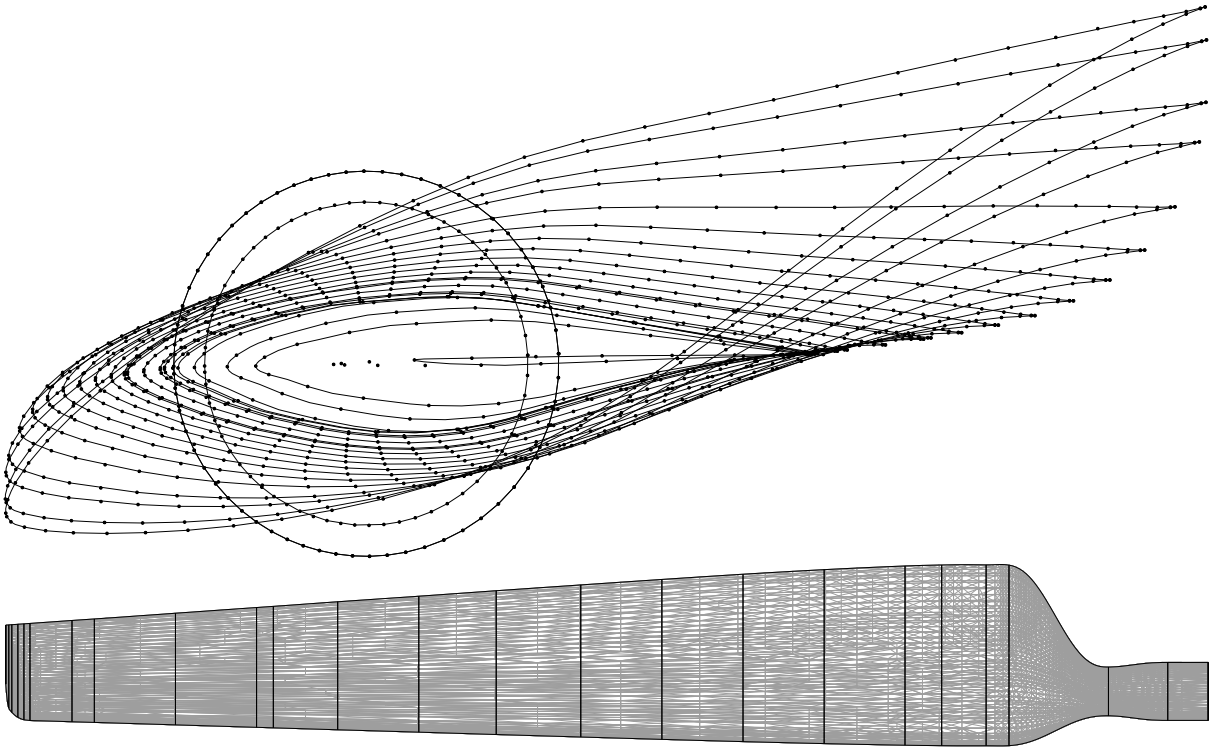


Figure 2.10: Blade design – points and splines in CATIA V5 generated and imported via MATLAB script and Microsoft Excel macro

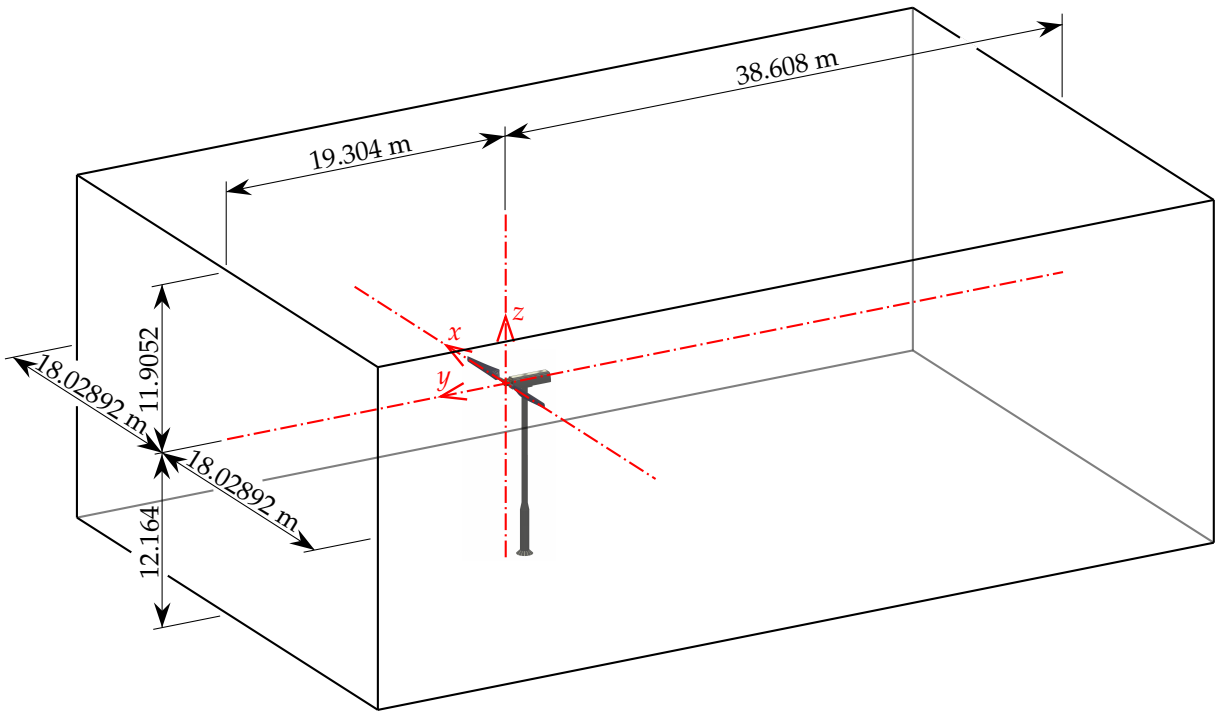


Figure 2.11: Dimensions of NASA's wind tunnel and position of NREL's wind turbine inside

2.2 Available Test Data and its Utilization

As already mentioned "the purpose of [...] [NREL's UAE Phase VI] wind tunnel test was to acquire accurate quantitative aerodynamic and structural measurements on a wind turbine, geometrically and dynamically representative of full scale machines, in environment free from pronounced inflow anomalies." [21] Here great importance was spent on specifying the required measurement data types, which should later on be used for development and validation of engineering models of recent plants for wind energy generation. The variety of test sequences produced an enormous output of measurement data. It was stored completely and is provided for different purposes of use via an online database. It is accessible via the url `https://www.nrel.gov/extranet/uaewtdata/seq.html` [26] requiring log in data from NREL.

In the following a simple introduction of the online database structure will be given, supported by screen shots seen in Figure 2.12. It is required as step (1) to log in using log in data from NREL. Based on the naming in Table 1 on page 14 in [21] and its more detailed descriptions on pages 13 to 24 in [21] the selection of the test sequence will be done next (2). Then (3) all executed test runs for a specific sequence are listed in a table via nominal wind tunnel velocity (line) and yaw angle (row). Zero, one or two numbers in each cell indicate the number of selectable repetitions. Subsequently (4) all required measurement data channels, that should be output, can be enabled. Short explanations to each channel will be reached by just clicking on its naming (5). For detailed ones see [21]. At least the output options have to be set. Here the entire time history ("Entire raw channel selection"), the average over the complete run time ("Channel average"), the average over every rotor cycle ("Cycle average") or the "Azimuth average" of every enabled channel can be requested. The data will be given as pure ASCII text on a new web page ("ASCII Text") or as compressed file (*.zip) also with ASCII data ("ZIP Compressed") via clicking "Submit". Clicking "Reset" will reset the complete page and remove all selections. (6) shows the data output of some example channels using the "Entire raw channel selection" and "ASCII Text" options.

One main component of the extended simulations presented in the following chapter is the rotor blades to be treated as flexible structures. As a consequence they get deformed under the effect of surface pressure caused by the fluid flow. In this simulations the structural deformation is on the fluid side realized via mesh motion in consequence of moving blades' boundary patches. Hereby the maximum mesh motion corresponds to the maximum blade deformation, which normally appears at the blades' tip ($R = 5.029$ m) and forms a critical value for evaluating the fluid mesh quality, also presented below. So consequently an estimation of the (maximum) blade deformation becomes necessary. This is done via the simple model of a piecewise defined cantilevered Euler-Bernoulli beam seen in Figure 2.13. Its resp. the real blades' stiffness properties are summarized in Table 2.2 extracted from Table A-9 on page 76 in [21]. Fixed-end moments M_A were measured during all test runs for both blades respectively in

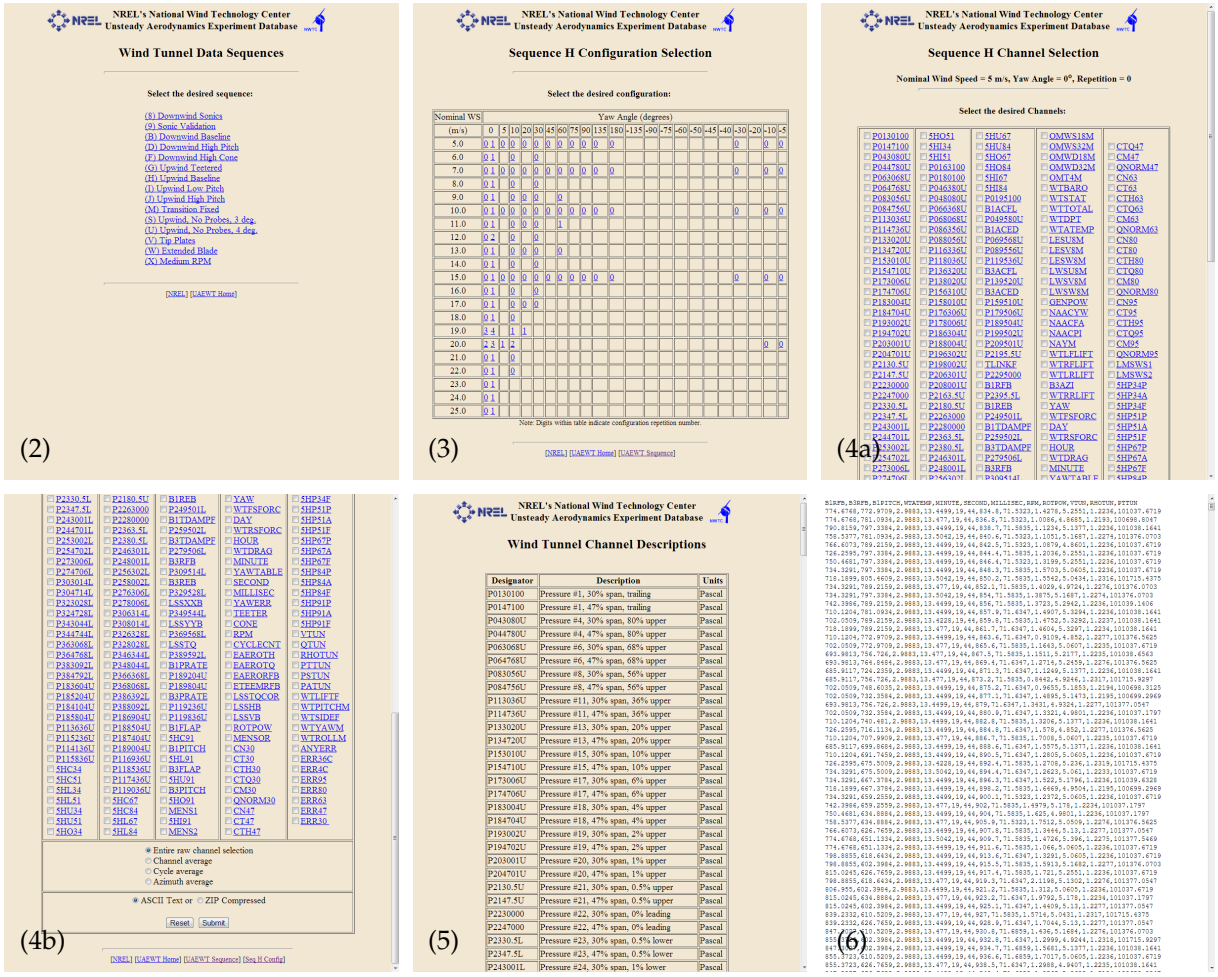


Figure 2.12: User's guide for the online database of NREL's UAE Phase VI: (1) Log in. (2) Select test sequence. (3) Select yaw angle and wind tunnel velocity. (4) Select data channels and output options. (5) Click on data channel name to get to the wind tunnel channel description. (6) Output example using "Entire raw channel selection" and "ASCII Text". (screen shots of [26])

the y- and z-direction. The labels behind the >-symbols indicate the appropriate measurement channels the data is stored in.

- fixed-end moment of blade 1 in y-direction $M_{A,1,y}$ = blade 1 root flap bending moment >B1RFB
- fixed-end moment of blade 3 in y-direction $M_{A,3,y}$ = blade 3 root flap bending moment >B3RFB
- fixed-end moment of blade 1 in z-direction $M_{A,1,z}$ = blade 1 root edge bending moment >B1REB
- fixed-end moment of blade 3 in z-direction $M_{A,3,z}$ = blade 3 root edge bending moment >B3REB

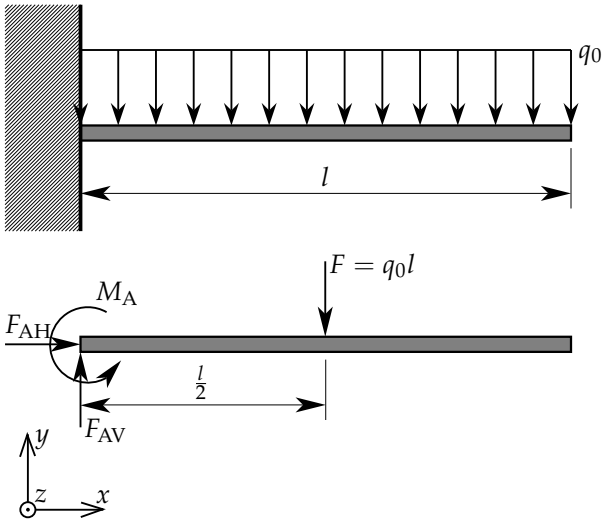


Figure 2.13: Cantilevered Euler-Bernoulli beam with constant line load for estimation of the blade deformation resp. mesh motion

According to the main flow around a blade, the resulting surface pressure distribution and the lower stiffness properties in this direction, the behavior in y-direction is of greatest interest. Here the assumption of a constant line load $q(x) = q_0$ causing the fixed-end moment is made. It seams quite reasonable, since the surface area is shrinking with increasing radius while the inflow velocity in contrast is increasing due to the increasing rotational component. A dummy load F at the position $x = \frac{l}{2}$ causes the same fixed-end moment as the line load itself.

$$M_A = \int_0^l xq(x)\ dx = \int_0^l xq_0\ dx = \frac{1}{2}l^2 q_0 = F\frac{l}{2} = q_0 l\frac{l}{2} \tag{2.1}$$

So the line load can be estimated to

$$q(x) = q_0 = \frac{2M_A}{l^2} \tag{2.2}$$

Figures B.1 to B.6 in Appendix B show the measured flap bending moments for both blades and wind tunnel velocities of 5, 15 and 25 $\frac{m}{s}$. Maximum root flap bending moments of about 920, 3520 and

Table 2.2: Stiffness properties of NREL's test wind turbine blade used for estimation of the blade deformation respective mesh motion (taken from [21])

Radius x **in** m	**Torsional stiffnes** GI_T **in** Nm^2	**Axial stiffnes** EA **in** N	**Edgewise stiffnes** EI_{zz} **in** Nm^2	**Flapwise stiffnes** EI_{yy} **in** Nm^2
5.000	2.73×10^5	9.36×10^7	6.71×10^5	6.80×10^4
4.780	3.34×10^5	1.09×10^8	8.45×10^5	9.05×10^4
4.696	3.57×10^5	1.15×10^8	9.14×10^5	9.97×10^4
4.391	4.57×10^5	1.38×10^8	1.21×10^6	1.40×10^5
4.086	5.73×10^5	1.61×10^8	1.56×10^6	1.89×10^5
4.023	5.97×10^5	1.66×10^8	1.63×10^6	2.00×10^5
3.781	7.07×10^5	1.85×10^8	1.97×10^6	2.49×10^5
3.476	8.56×10^5	2.10×10^8	2.44×10^6	3.20×10^5
3.185	1.03×10^6	2.35×10^8	2.97×10^6	4.03×10^5
3.172	1.03×10^6	2.36×10^8	2.99×10^6	4.07×10^5
2.867	1.22×10^6	2.56×10^8	3.59×10^6	4.79×10^5
2.562	1.45×10^6	2.92×10^8	4.38×10^6	6.08×10^5
2.343	1.63×10^6	3.21×10^8	5.02×10^6	7.20×10^5
2.257	2.19×10^6	3.52×10^8	6.23×10^6	8.05×10^5
1.952	2.45×10^6	3.57×10^8	6.49×10^6	9.16×10^5
1.648	2.74×10^6	3.67×10^8	6.69×10^6	1.07×10^6
1.510	2.89×10^6	3.84×10^8	6.92×10^6	1.17×10^6
1.343	3.09×10^6	4.04×10^8	7.20×10^6	1.30×10^6
1.257	3.18×10^6	4.07×10^8	7.22×10^6	1.35×10^6
0.883	4.02×10^5	3.95×10^8	1.65×10^6	1.65×10^6
0.660	2.14×10^6	1.03×10^9	4.77×10^6	4.77×10^6
0.610	3.12×10^6	1.33×10^9	6.10×10^6	6.10×10^6
0.559	3.68×10^6	1.48×10^9	6.85×10^6	6.85×10^6
0.508	3.00×10^6	1.01×10^9	4.06×10^6	4.06×10^6
0.483	1.31×10^7	6.69×10^9	1.72×10^7	1.72×10^7
0.369	3.16×10^5	1.04×10^9	4.16×10^5	4.16×10^5

6600 Nm appear, which lead to line loads of $q(x) = q_0 \approx 73$, 279 and 522 $\frac{\text{N}}{\text{m}}$. Simulation of the Euler-Bernoulli beam with Abaqus/CAE 6.12-1 applying the stiffness properties for y-direction from Table 2.2, which means the flapwise stiffness properties (EI_{yy}), and the estimated line load values, results in the deformations seen in Figure 2.14. Since this values are based on a static model, deformations for the dynamic case, which are about twice as big $u_{\text{dyn}} \approx 2u_{\text{stat}}$, have to be taken into account. This results in maximum dynamic deformations $u_{\text{dyn,max}} \approx -5.1$, -19.5 and -36.4 mm.

In order to obtain a statement on the expected quality of later simulation results, evaluations concerning the quality of mesh and set simulation parameters have to be performed. This especially is important in regard to the *N*-code co-simulation, which is extremely intensive in computational power and time. The evaluations will be done in two steps. In step A the settings are evaluated to yield the right surface pressure distribution of the blades, which indicates the right blade deformation from the fluid side. This is done via comparing normalized pressure coefficients c_p, meaning surface pressure relative to static pressure, measured at specific locations on the blade surface to those calculated using results of simpler pre-simulations. Figure 2.15 shows those locations, where pressure coefficients will be evaluated.

Step B is the comparison between the resulting low speed shaft torques from measurement ($T_{\text{LSS,meas}}$) and simulation ($T_{\text{LSS,sim}}$). The direct measurement is stored in channel >LSSTQ, the corrected one in channel >LSSTQCOR. The correction terms can be seen in [21] on page 58. $T_{\text{LSS,sim}}$ is output from

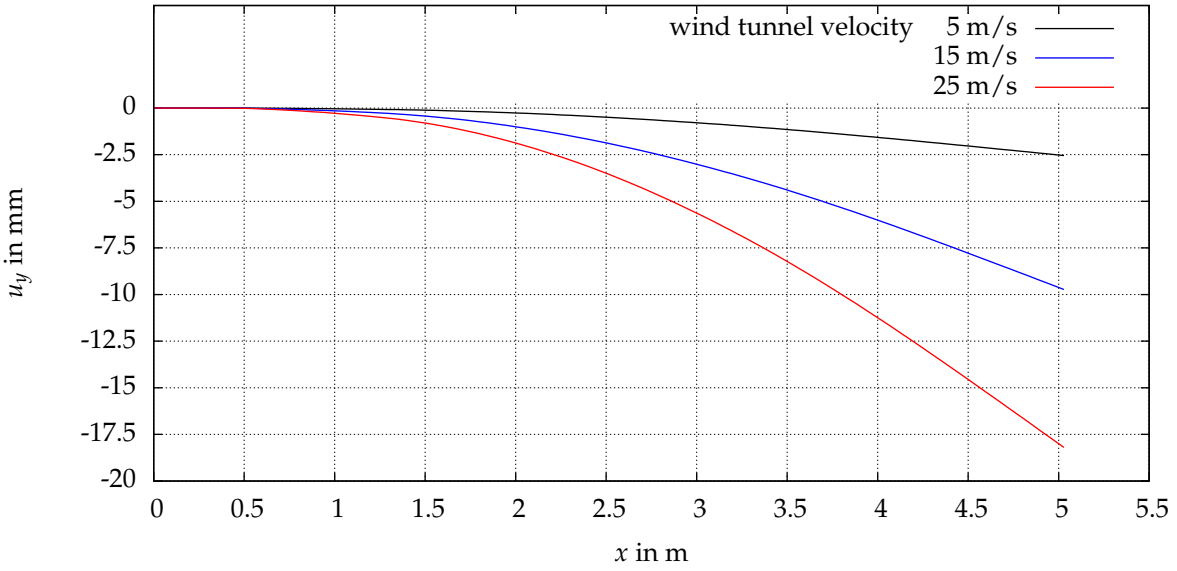


Figure 2.14: Static deformation of the Euler-Bernoulli beam calculated using Abaqus/CAE 6.12-1

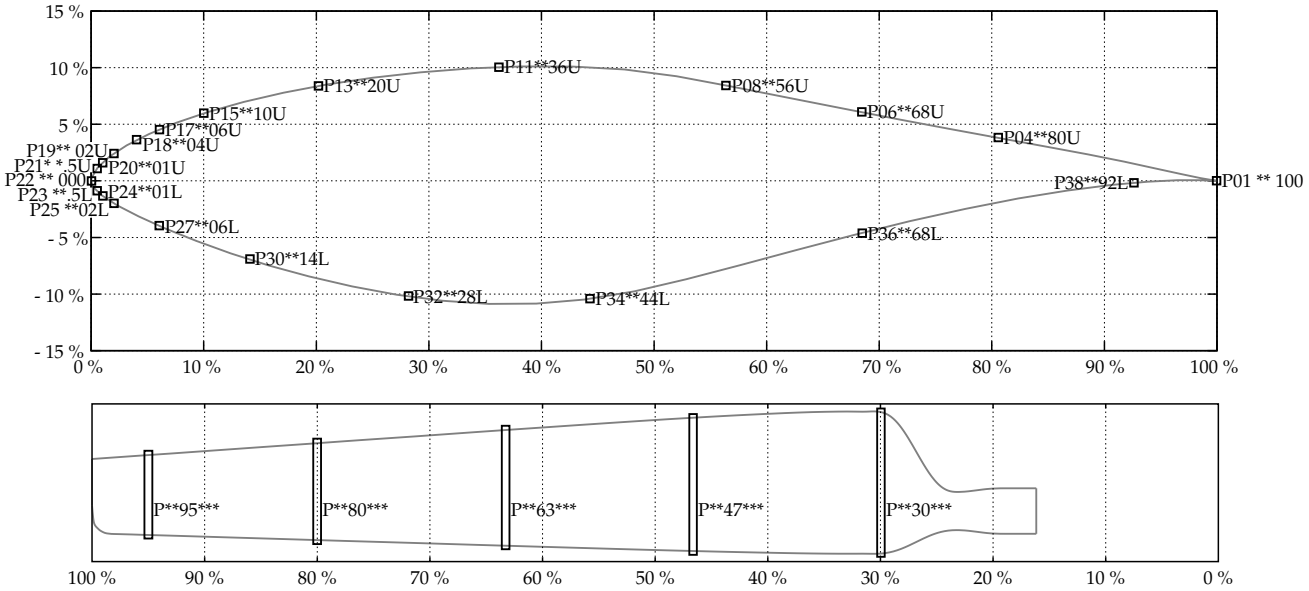


Figure 2.15: Pressure taps for evaluation of normalized pressure coefficients c_p are located at 30, 46.6, 63.3, 80 and 95 % radius (0 m ≙ 0 % and 5.029 m ≙ 100 % and at 100, 80, 68, 56, 36, 20, 10, 6, 4, 2, 1 and 0.5 % chord on the upper and at 0, 0.5, 1, 2, 6, 14, 28, 44, 68 and 92 % chord on the lower surface of NREL's test wind turbine blade. E.g. pressure tape number 21, at 46.6 % radius and 0.5 % chord on the upper surface would be stored in channel >P2147.5U (values taken from [21])

OpenFOAM during the solving process by simply including the tool "forces". Detailed information on this will be given in the next chapter. The calculated time history of $T_{\text{LSS,meas}}$ for different wind tunnel velocities (5, 15 and 25 $\frac{\text{m}}{\text{s}}$) can be seen in Figures B.7 to B.12 in Appendix B.

As it was mentioned before, the introduced evaluation steps A and B will be done on simpler simulation cases. This has the advantage of less computational power requirements and time. The cases are

- a steady-state simulation of the quasi-rotating turbine excluding tower using the `MRFSimpleFoam` solver (later step 1/5)
- and a transient simulation of the full, rotating, ridgid turbine using the `pimpleDyMFoam` solver (later step 2/5).

Both are pure CFD simulations. Details will be presented in the following chapter.

3

N-Code Co-Simulation of NREL's UAE Phase VI Wind Turbine

To achieve the aim of accelerating the development-to-market time in modern product design, new, more accurate simulation techniques become more and more important. In this context various terms like "fluid-structure interaction (FSI)", "multiphysics", "N-code coupling" or "co-simulation" appear. To avoid confusion a short introduction to the common terminology should be given here, starting on the simplest example of a fluid-structure interaction problem. It is quite easy to imagine, e.g. a flag fluttering in the wind. This problem bases on two main physical phenomena, a flexible structure alias flag and fluid flow alias wind. Generally speaking at one moment the fluid is seeing a structure of defined shape. Fluid flow around this structure is developing and causing a surface pressure field loading the structure. As a consequence the flexible structure is deforming. This again results in changes in the fluid flow, what ends in new structural behavior and so on. It becomes clear, that the fluid is influencing the structure and the other way round. The term "multiphysics" is describing such problems, where more than one physical phenomenon is involved. Solving just one of both phenomena, like traditional engineering approaches did, is not possible here. One option to get a solution is to describe the complete fluid-structure system in one set of differential equations to be solved. This solution process is called monolithic and is afterwards only applicable to this special type of problem again. So to be much more flexible and reusable another method is used. Simulation tools solving computational solid mechanics (CSM) meaning pure structural problems, e.g. Abaqus or Ansys, just as pure computational fluid dynamics (CFD), e.g. StarCCM+ or OpenFOAM are available. Key step is to combine those tools in order to solve the coupled problem containing CSM and CFD. In the starting example above, there are two codes ($N = 2$) involved. Imaging e.g. a controller enforcing a specific structural behavior depending on the actual fluid flow and structural deformation, it becomes clear, there can be quite fast more than two

or three codes – N codes – requiring a structured coupling procedure. This is named "N-code coupling" or "co-simulation".

3.1 Enhanced MultiPhysics Interface Research Engine (EMPIRE)

EMPIRE is a tool for doing co-simulation of multiple (N) codes, so it exactly satisfies the above mentioned requirements for a structured coupling procedure handling N codes. EMPIRE is the abbreviation of Enhanced MultiPhysics Interface Research Engine and is developed by Stefan Sicklinger and Tianyang Wang at the Chair for Structural Analysis (Prof. Dr.-Ing. K.-U. Bletzinger) at Technische Universität München. The vision behind EMPIRE is to be a flexible and efficient tool for doing co-simulation with multiple codes including parallel applications. It should have a open data structure and interface smoothly with clients.

EMPIRE is based on a so called client-server approach, which is well known from computer networks or the world wide web. The server (from lat. servire) called EMPEROR is the connection between all clients (from lat. cliens). Clients, just another name for solvers or codes in this case, are requesting data from the server for doing their calculations and responding the result data to the server. With this basic functionality a N-code co-simulation can be realized. The second main part of EMPIRE is the EMPIRE API. API is short for Application Programming Interface. The EMPIRE API ensures the interaction between all clients and the server. Therefore its input is based on a .xml file. Clients connect via Socket-like MPI-2.2 communication. It is written in C++ and wrapped in C for better interoperability. All MPI calls are handled internally, so they are not visible for the user. The server EMPEROR has also an input based on a .xml file, which is read at the beginning of the co-simulation. With this the complete coupling scenario is setup. EMPEROR can do asynchronous listening (OpenMP). Clients are connected via Socket-like MPI-2.2 communication. Open Meta-database data structure is used. EMPEROR is including very efficient and accurate mapping algorithms based on (Dual) Mortar methods. Mapping gets necessary in case of non-matching grids at the interface between different clients. Properties on one interface side (mesh A) have to be transformed into appropriate properties on the opposite interface side (mesh B).

The turbulent, full-scale fluid-structure-signal co-simulation of NREL's UAE Phase VI wind turbine should demonstrate the functionality and accuracy of the coupling tool EMPIRE. This includes validation via comparison of obtained simulation results and existing measurement data. The ability of EMPIRE to handle huge amounts of data should also be shown.

Basically a fluid-structure-signal co-simulation combines, to put it another way couples three modeled and simulated physical phenomena. The main one is the fluid flow through the wind tunnel around the wind turbine forcing rotation, with that generation of of electric power and deformation of the flexible rotor blades. All other components, e.g. tower with nacelle, are assumed to be ideal rigid. So the physical boundaries of the rotor with its blades state the coupling interface, where the fluid forces acting on the blade surface have to be mapped to the structure. The open source tool OpenFOAM was selected to simulate the physics of the fluid.

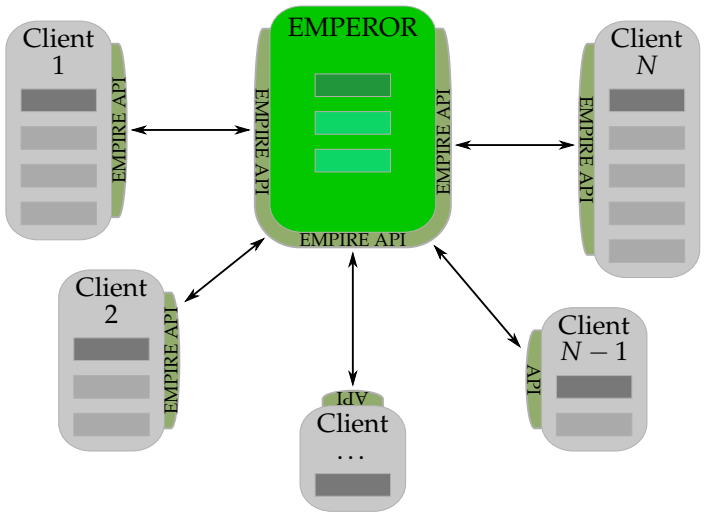


Figure 3.1: The Enhanced MultiPhysics Interface Research Engine (EMPIRE) is based on a so called client-server approach, where the server EMPEROR is connected to N client codes

The next separately considered physical phenomenon is the structural behavior of the rotor blades. For this the in-house FEM tool Carat++ will be used, modeling the blades as shell segments with the stiffness properties from Table 2.2 on page 35. The blades are loaded by the mapped fluid forces. Due to that, they are deforming. This deformation results in a point displacement, which is mapped back to the fluid mesh at the interface, where the fluid domain boundaries of the rotor are moving and causing changes in the fluid flow.

In order to realize a third client code, since until now there are only two of them with the fluid-structure interaction problem, the parts in the nacelle including gearbox, bearing and generator are also modeled as autonomous system. This results in a ordinary differential equation (ODE). According to the normal physical behavior the black-box "Gearbox + Generator" would receive a rotational speed from the structural solver and react with, respectively send back a resistive torque arising from the generator and friction. For purposes of better coupling performance, meaning better stability and convergence, another coupling scheme is used. The "Gearbox + Generator" black-box will receive the moment from the fluid forces out of Carat++ and send a actual rotational speed to OpenFOAM. Moment and rotational speed are no field properties, but rather pure signals. The numerical computing environment MATLAB is coupled to calculate the rotational speed signal out of the moment signal based on the simple ODE. The basic and detailed concept of the entire coupling scenario can be seen in Figure 3.2.

Combination of overall four main tools, each containing different sub-tools, requires a stepwise evolution to the full co-simulation case. Only that way it is possible to guarantee the best simulation performance including a optimized and tested combination of sub-tools and all settings. Debugging of errors or problems would be nearly impossible without this stepwise procedure. In the evolutionary history the last three of five steps are containing the coupling tool EMPIRE. Since step 5/5 stands for the full co-simulation case, which was already presented above, the remaining steps will be introduced subsequently in reverse order.

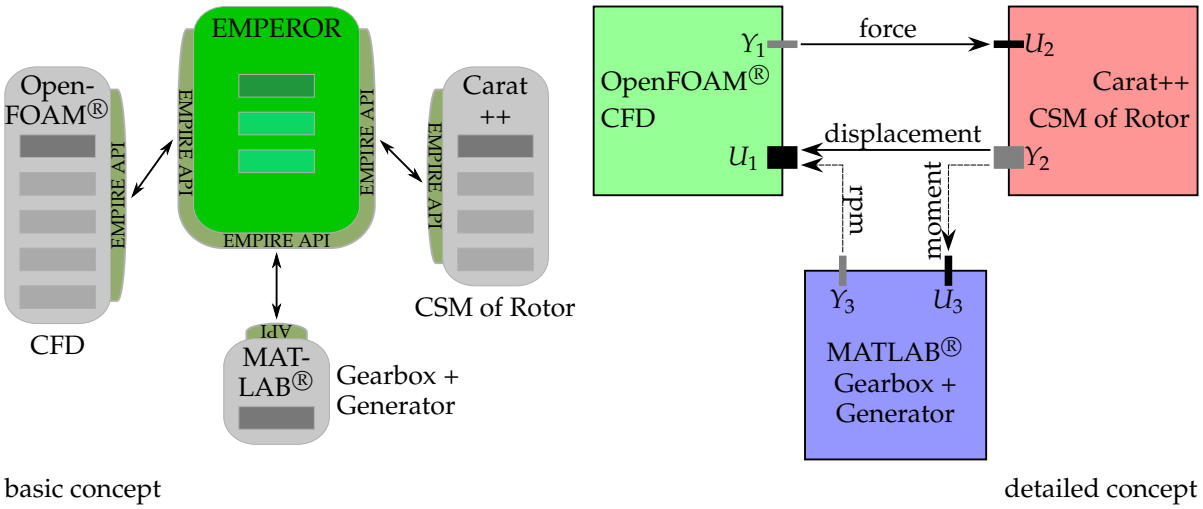


Figure 3.2: Basic and detailed concept for coupling OpenFOAM, Carat++ and MATLAB to do a full fluid-structure-signal co-simulation of NREL's UAE Phase VI wind turbine with EMPIRE

Doing a pure fluid-structure interaction co-simulation without simulating the behavior of gearbox, bearings and generator, means without coupling MATLAB, is step 4/5. In this case the rotational speed of the rotor is set via OpenFOAM input file to the constant value of 72 min^{-1}.

A even simpler step 3/5 is to do only one-way coupling. That means Carat++ is replaced by the client meshClientTurbomachinery, which does not receive any input (except the actual time step), but is sending a user predefined point displacement for the blade surface. This coupling scheme is used to figure out the possible maximum of mesh deformation on the fluid side and to check the mesh quality during deformation. A parabolic point displacement only in negative y-direction increasing linearly with every time step is therefore sent by meshClientTurbomachinery. It is zero at the blades' root and has its maximum at the blades' tip approximating the real blade deformation. A simplified version of the file AbstractDataCreator.h, where the calculation of the actual point displacements is stated, can be found in Appendix C on page 93.

Steps 2/5 and 1/5 will be introduced in the next section, since they are pure CFD simulations not making use of the EMPIRE co-simulation environment.

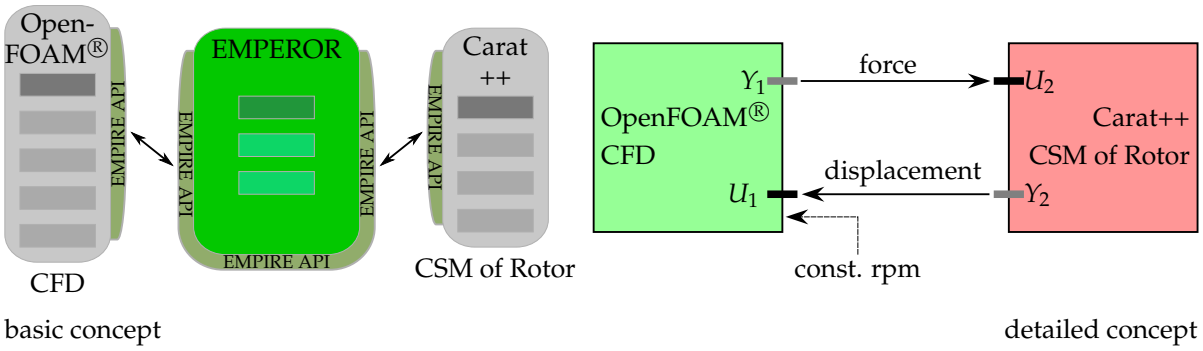


Figure 3.3: Basic and detailed concept for coupling OpenFOAM and Carat++ with EMPIRE to do a fluid-structure interaction co-simulation of NREL's UAE Phase VI wind turbine rotating at constant rpm

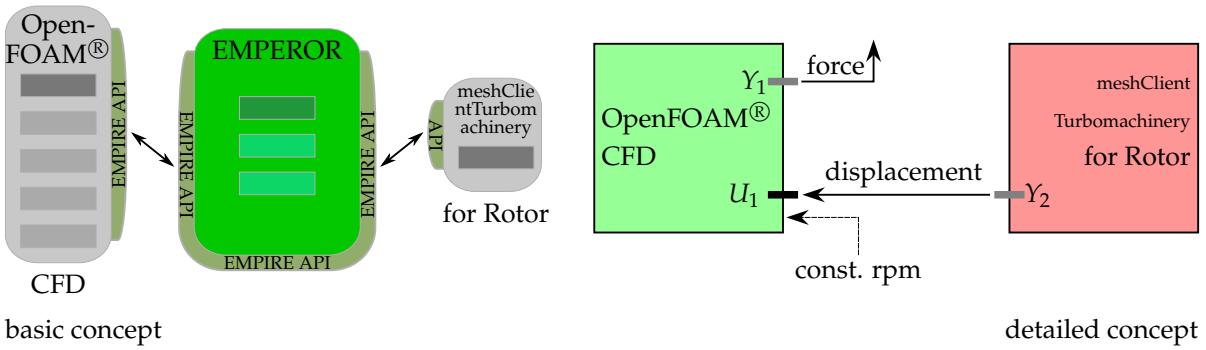


Figure 3.4: Basic and detailed concept for coupling OpenFOAM and meshClientTurbomachinery with EMPIRE to use predefined, parabolic point displacements for mesh checks

3.2 The CFD Client OpenFOAM

OpenFOAM (Open Source Field Operation and Manipulation) is first and foremost a `C++` Library, used for creating numerical solvers, pre- and post-processing tools for the solution of continuum mechanics problems, here with a focus on computational fluid dynamics (CFD). OpenFOAM is a free and open source software. Its code is released under the GNU General Public License, produced by OpenCFD Ltd. at ESI Group, which is owner of the OpenFOAM trademark, and distributed by the OpenFOAM Foundation. The object oriented structure holds the opportunity of quite easy customized extensions. And, as already mentioned in the first chapter, the implemented numerics use the finite volume method (FVM) on structured and unstructured meshes.

The basics, including the use of OpenFOAM, its case folder structure, the minimum required files, the mesh definition, available solvers etc., are introduced quite well in its user guide [13]. Especially the provided tutorials are recommended in purpose of getting started with the software package. But at this point it should also be mentioned, that one big handy cap is the lack of documentation, which makes it hard to find detailed information on more special features. Here e.g. discussion boards can help.

3.2.1 Implementation of Rotation – OpenFOAM's Arbitrary Mesh Interface (AMI)

OpenFOAM can deal with rotating objects next to static ones. There exist different ways of handling motion of some boundaries during CFD simulations. In OpenFOAM the so called Arbitrary Mesh Interface (AMI) is implemented. Thereby in general two sub meshes are moving relatively to each other at a sliding interface, which is nothing more then coincident boundaries on both meshes. OpenFOAM provides an application as cyclic AMI. Therefore the fluid domain bounded by the wind tunnel walls is divided into two sub-meshes, also called regions. The situation can be seen in Figure 3.5. A static sub-mesh, here named Stator, contains tower and nacelle, while a second rotating one houses the Rotor (hub, blade 1 and 3). Part of the sliding interface are the Rotor's outer boundary (here AMI1) and the Stator's inner boundary (here AMI2).

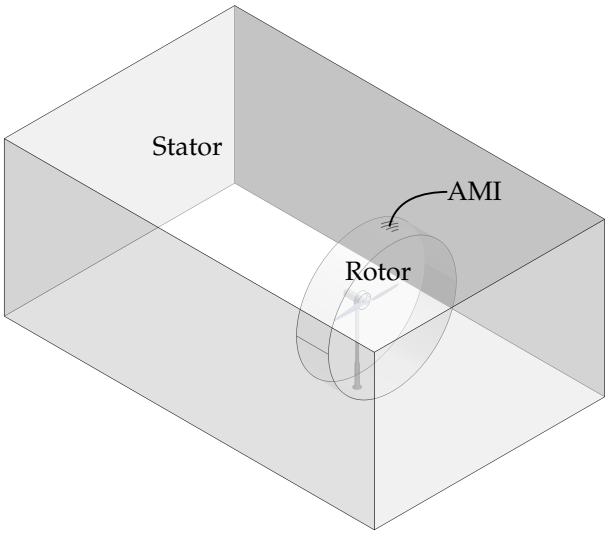


Figure 3.5: Arbitrary Mesh Interface (AMI) between the static sub-domain (Stator) containing tower and nacelle and the rotating sub-domain (Rotor) containing hub, blade 1 and 3

To extend a OpenFOAM case to one containing AMI, some adjustments have to be done. They all can be reproduced doing the recommended tutorials [14]

`$FOAM_TUTORIALS/incompressible/pimpleDyMFoam/mixerVessel2DAMI`
(2D spinning rotor and stationary stator) and

`$FOAM_TUTORIALS/incompressible/pimpleDyMFoam/propeller`
(3D propeller, meshed with snappyHexMesh with feature line conformance).

First of all the single fluid domain has to be split into two independent domains Stator and Rotor. It is essential for the Rotor to be rotationally symmetric to its rotational axes, so that the sliding interface can be provided during the whole simulation. Necessary modifications to obtain two regions will be introduced in the next subsection for the third party tool STAR-CCM+. Next the rotor-sided interface boundary patch AMI1 must be typed as `cyclicAMI` with the corresponding cyclic AMI neighbor patch AMI2 (here). The following lines must exist in the `$EMPIRE_CASE/OF/constant/polyMesh/boundary` file:

Listing 3.1: AMI1 part of $EMPIRE_CASE/OF/constant/polyMesh/boundary

```
1  AMI1
2  {
3      type            cyclicAMI;
4      nFaces          (number);
5      startFace       (number);
6      matchTolerance  0.0001;
7      neighbourPatch  AMI2;
8      transform       noOrdering;
9  }
```

The sliding interface boundary patch of the Stator AMI2 has to be defined respectively.

Listing 3.2: AMI2 part of $EMPIRE_CASE/OF/constant/polyMesh/boundary

```
    AMI2
    {
        type            cyclicAMI;
        nFaces          (number);
        startFace       (number);
        matchTolerance  0.0001;
        neighbourPatch  AMI1;
        transform       noOrdering;
    }
```

The necessary modifications to define the boundary patches during the meshing process will also be presented in the following subsection.

OpenFOAM takes all specifications concerning the AMI from the `dynamicMeshDict` dictionary, which must exist in the `$EMPIRE_CASE/OF/constant/` folder.

Listing 3.3: $EMPIRE_CASE/OF/constant/dynamicMeshDict (AMI only)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dynamicFvMesh   solidBodyMotionFvMesh;

motionSolverLibs ( "libfvMotionSolvers.so" );

solidBodyMotionFvMeshCoeffs
{
    cellZone        Rotor;

    solidBodyMotionFunction  rotatingMotion;
    rotatingMotionCoeffs
    {
        CofG           (0 0 0);
        radialVelocity (0 432 0); // in deg/s
    }
}

// ************************************************************************* //
```

As it can be seen, the name of the rotating `cellZone` (region or sub-mesh) has to be set, as well as the center of rotation (`CofG`) and the axis and speed of rotation as vector (`radialVelocity`). In the latter case values in degree per second $\left(\frac{^\circ}{s}\right)$ are required.

After generating the mesh and before solving, the OpenFOAM pre-processing tool `topoSet` has to be executed. It requires the following `topoSetDict` in the `$EMPIRE_CASE/OF/system/` folder, which simply contains the commands to add the faces of both AMI patches (`"AMI.*"`) to a new `faceSet` AMI, which is used by the solver.

Listing 3.4: $EMPIRE_CASE/OF/system/topoSetDict (AMI only)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    AMI;
        type    faceSet;
        action  new;
        source  patchToFace;
        sourceInfo
        {
            name  "AMI.*";
        }
    }
);

// ************************************************************************* //
```

It should also be mentioned, that there are no special requirements on both meshes touching the sliding interface. So e.g. non-matching grids (see Figure 3.10 on page 53) or even different cell types, like tetrahedrals on one and hexahedrals on the other side, are possible.

For solving OpenFOAM's `pimpleDyMFoam` solver is used. It is a extended `pimpleFoam` solver for handling dynamic meshes (`*DyM*`). Step 2/5 of the presented simpler simulations is to do a transient stand-alone CFD simulation of the full wind turbine using the `pimpleDyMFoam` solver in combination with the AMI. All parts are ideal rigid for this purpose and the rotor is rotating at its constant rpm rate of 72 min^{-1}.

Step 1/5 is to do also a stand-alone but steady-state CFD simulation of the rotating turbine. This is possible with the `MRFSimpleFoam` solver, which is adding Coriolis forces corresponding to a quasi-rotation. `MRFSimpleFoam` can handle AMIs, so the mesh can remain the same at this point. This way it can be ensured, that the most critical submesh of the Rotor stays the same for all simulation cases. So useful tests regarding the mesh quality can be provided using the `MRFSimpleFoam` solver and results can be mapped to a transient PIMPLE case more easily. Only the surrounding Stator sub-mesh has to undergo some changes, since the entire mesh has to be rotationally symmetric. The Rotor mesh automatically satisfies this requirement, while the Stator mesh must be edited accordingly. Therefore the tower is removed and the cubic wind tunnel is replaced by a cylindrical "wind tunnel" of adequate dimensions. Taking a normal `simpleFoam` case from the tutorials, the file `MRFZones` has to be added to the `$EMPIRE_CASE/OF/constant/` folder. It contains fixed patches (`nonRotatingPatches`), the `origin`, the rotational `axes` and the rotational speed. Attention has to be spend on the difference, that an angular velocity ω (`omega`) in radiant per second $\left(\frac{\text{rad}}{\text{s}}\right)$ is required here.

Listing 3.5: $EMPIRE_CASE/OF/constant/MRFZones

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      MRFZones;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

1
(
    Rotor
    {
        // Fixed patches (by default they 'move' with the MRF zone)
        nonRotatingPatches ();

        origin    (0 0 0);
        axis      (0 1 0);
        omega     constant 7.5398; // in rad/s
    }
)

// ************************************************************************* //
```

3.2.2 Mesh Generation Using CD-adapco's STAR-CCM+

STAR-CCM+ is a product from the multinational computer software company CD-adapco, which has its headquarters in Melville, New York in the USA. CCM is standing for Computational Continuum Mechanics. The software tool was introduced in 2004 and used as pure meshing tool for preprocessing of the simulation cases presented in this thesis. Available was version 7.06. High quality requirements, complex geometry and a lack of documentation for OpenFOAM's meshing tools are reasons, why STAR-CCM+ was selected.

Appendix D is providing detailed information about all used settings etc. In this subsection the basic ideas and the basic procedure should be introduced next to a discussion of critical settings. STAR-CCM+'s user guide [5] contains everything necessary for getting started with the software and also for advanced usage.

The mesh will be built using seven CAD geometries designed in CATIA V5:

- *WindTunnel*.CATPart/.stp
- *Tower*.CATPart/.stp
- *AMI*.CATPart/.stp
- *HelpCylinder*.CATPart/.stp
- *Hub*.CATPart/.stp
- *Blade1*.CATPart/.stp

- *Blade3*.CATPart/.stp

They can be seen in Figure 3.6, while Figure 3.7 provides detailed information about dimensions and position in the global coordinate system of until now not characterized parts. For details on the missing tower, blades and hub see chapter 2. Hub, blade 1 and 3, in combination the rotor, are treated as autonomous parts with the advantage of having independent boundary patches, which can be used for defining individual surface mesh settings and for advanced use of the forces tool, which will be introduced later.

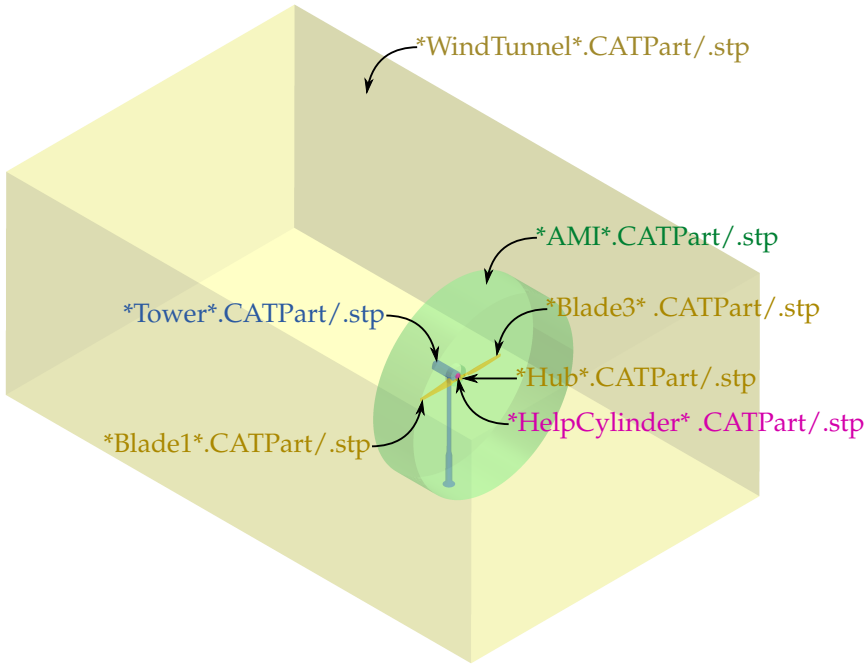


Figure 3.6: The Stator domain will be generated substracting *AMI*, *HelpCylinder* and *Tower* from *WindTunnel*, the Rotor domain subtracting *Blade1*, *Hub*, *Blade3* and *HelpCylinder* from *AMI*

The seven CAD geometries are imported in STAR-CCM+ via .stp format (missing license for CATIA files) in the tab "Geometry" as new "3D-CAD Models" and then transformed into parts. The final Stator part representing the static fluid domain is built by subtracting *AMI*, *HelpCylinder* and *Tower* from *WindTunnel*, what is provided as Boolean operation in the tab "Parts", as well as the final Rotor part representing the rotating domain by subtracting *Blade1*, *Hub*, *Blade3* and *HelpCylinder* from *AMI*. Surfaces are edited, so that they fit to the final boundary patches:

- stator
  - inlet
  - outlet
  - statorTower
  - statorWall
  - AMI2

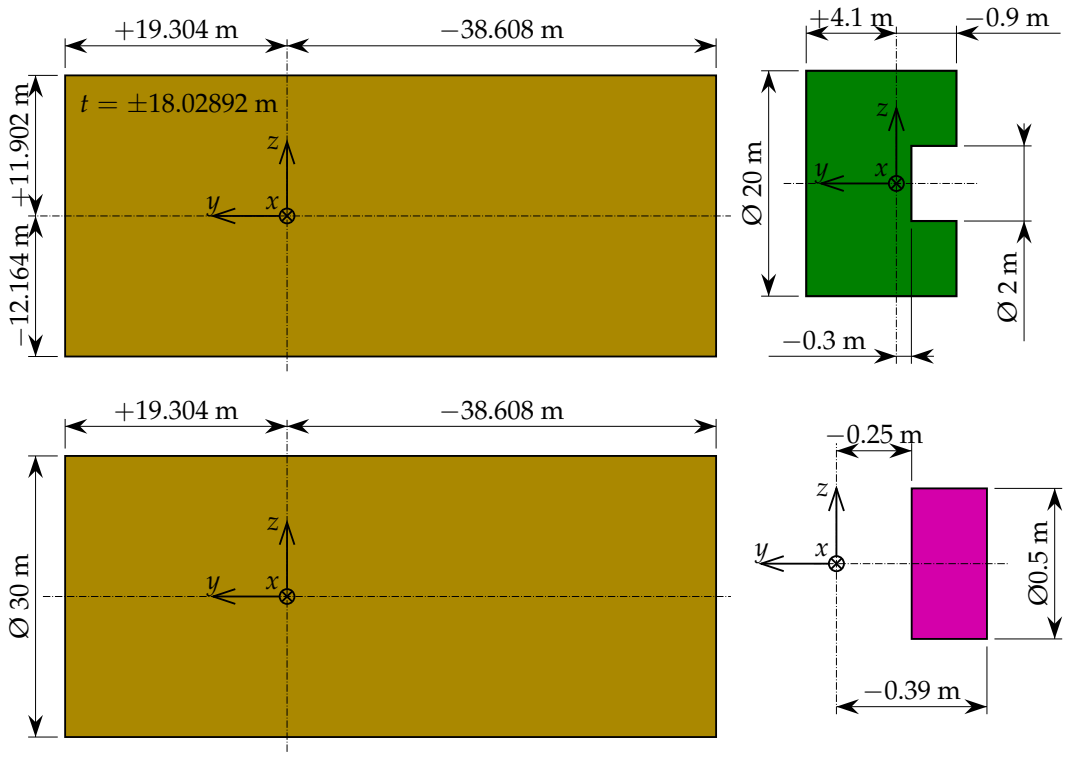


Figure 3.7: Dimensions and coordinate positions for the additional required CAD geometries *WindTunnel*, *AMI*, *WindTunnelCylinder* and *HelpCylinder*

- rotor
  - rotorBlade1
  - rotorHub
  - rotorBlade2
  - AMI1

Naming tower and wind tunnel wall as statorTower and statorWall provides the opportunity of shorter boundary condition file description (`"stator.*"`) in OpenFOAM. The same applies to rotorBlade1, rotorHub and rotorBlade3 (`"rotor.*"`).

Next both final parts are each assigned to a region. Perhaps some renaming and editing of boundaries becomes necessary at this point. A more important task is to edit the existing feature curves, which depend on the settings chosen during the CAD data import. Only really essential feature curves should remain, since they can effect negatively the local size of the surface mesh. The coarser the mesh the more importance comes to editing or deleting feature curves. So e.g. for both blades only the trailing edge feature curve was left, since here a sharp edge is required for modeling the right aerodynamic behavior.

For independent meshing of both regions, each has to be referred to a mesh continuum, which can be created under the tab "Continua". Here all surface and volume mesh specific settings have to be made. Once again they can be read for the later presented case in Appendix D. Detailed explanations

to all values can be found in [5]. Settings made in this tab are global ones for the related mesh region. Local changes, e.g. to get a coarser surface mesh at the AMI than at the blades surface – both belong to the same mesh region and continuum – can be done in the "Region – Boundaries" tab via enabling custom values.

After the meshing procedure the resulting mesh is exported as .ccm file containing all regions and boundaries but no results. The conversion from the .ccm format to the OpenFOAM mesh format is done via the OpenFOAM tool `ccm26ToFoam`. The tool needs the library `libccmio`. The .ccm file should be copied into the main OpenFOAM case folder and `ccm26ToFoam` be executed in the same directory together with the full file name. One inconvenient task is to edit the AMI boundary definitions in the `$EMPIRE_CASE/OF/constant/polyMesh/boundary` file using the following help file `boundaryEdit`. (number) must be replaced by the actual stored value from the real `boundary` file.

Listing 3.6: $EMPIRE_CASE/OF/constant/polyMesh/boundaryEdit

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundaryEdit;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// * * * * * * * * * * For editing real file "boundary"  * * * * * * * * * * //
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    AMI1
    {
        type            cyclicAMI;
        nFaces          (number);
        startFace       (number);
        matchTolerance  0.0001;
        neighbourPatch  AMI2;
        transform       noOrdering;
    }

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// * * * * * * * * * * For editing real file "boundary"  * * * * * * * * * * //
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    AMI2
    {
        type            cyclicAMI;
        nFaces          (number);
        startFace       (number);
        matchTolerance  0.0001;
        neighbourPatch  AMI1;
        transform       noOrdering;
    }

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// * * * * * * * * * * For editing real file "boundary"  * * * * * * * * * * //
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// ************************************************************************* //
```

The region names Stator and Rotor are not adopted during the conversion process. So afterwards they are called `cellZone_1` and `cellZone_2`. The correspondence can be identified using e.g. the tool `setSet`, which prints detailed mesh information for this purpose.

Next to internal quality checks in STAR-CCM+, OpenFOAM's tool `checkMesh` can be used to check the mesh quality after import. Most detected errors could be fixed by simply increasing the "Optimization Cycles" number in STAR-CCM+ after enabling the "Mesh Expansion Control" in the corresponding mesh continuum tab. `checkMesh`'s result (`checkMesh.log`) for the afterwards presented mesh is seen at the end of Appendix D.

The resulting mesh using the presented CAD geometries and the referenced settings can be seen in Figures 3.8 to 3.14.

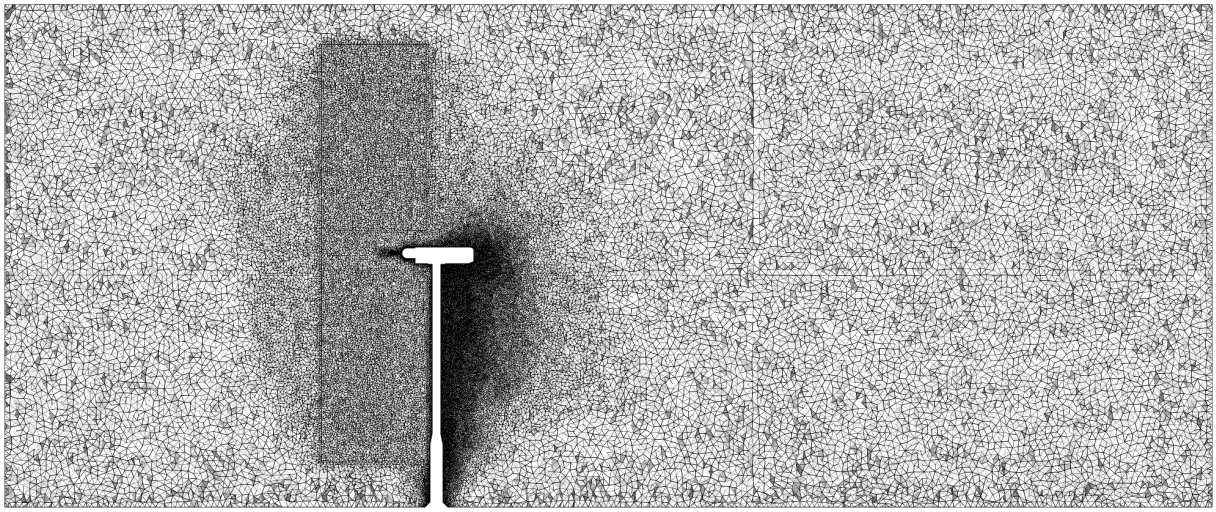


Figure 3.8: Tetrahedrals on the cutting plane $x = 0$ through the entire meshed fluid domain. The static and rotating domain with the Arbitrary Mesh Interface (AMI) and lines due to decomposing of the fluid domain for parallel meshing can be seen

3.2.3 Mesh Motion Based on the Arbitrary Lagrangian Eulerian Method (ALE)

The steps 3/5, 4/5 and the final one 5/5 are treating the wind turbine blades as flexible structures to model a more accurate physical behavior and to realize an additional client for coupling. Taking a closer look, the movement of every blade surface point is due to superposition of pure rigid body rotation and structural deformation mainly in negative y-direction during rotation. Thus for the simulations the blade behavior is split into this two components. As presented in section 3.2.1 the rigid body rotation is realized via OpenFOAM's implementation of the AMI, having a static and a rotating domain. Modeling the second component, the structural deformation means getting back a displacement vector for every related boundary point from the CSM client. Here the Arbitrary Lagrangian Eulerian method, short ALE, is used. The fluid mesh is described as a quasi-material. So if one mesh point is moving, then the whole mesh is deforming accordingly. This effects, that all mesh affinities (points, edges, faces, etc.) are

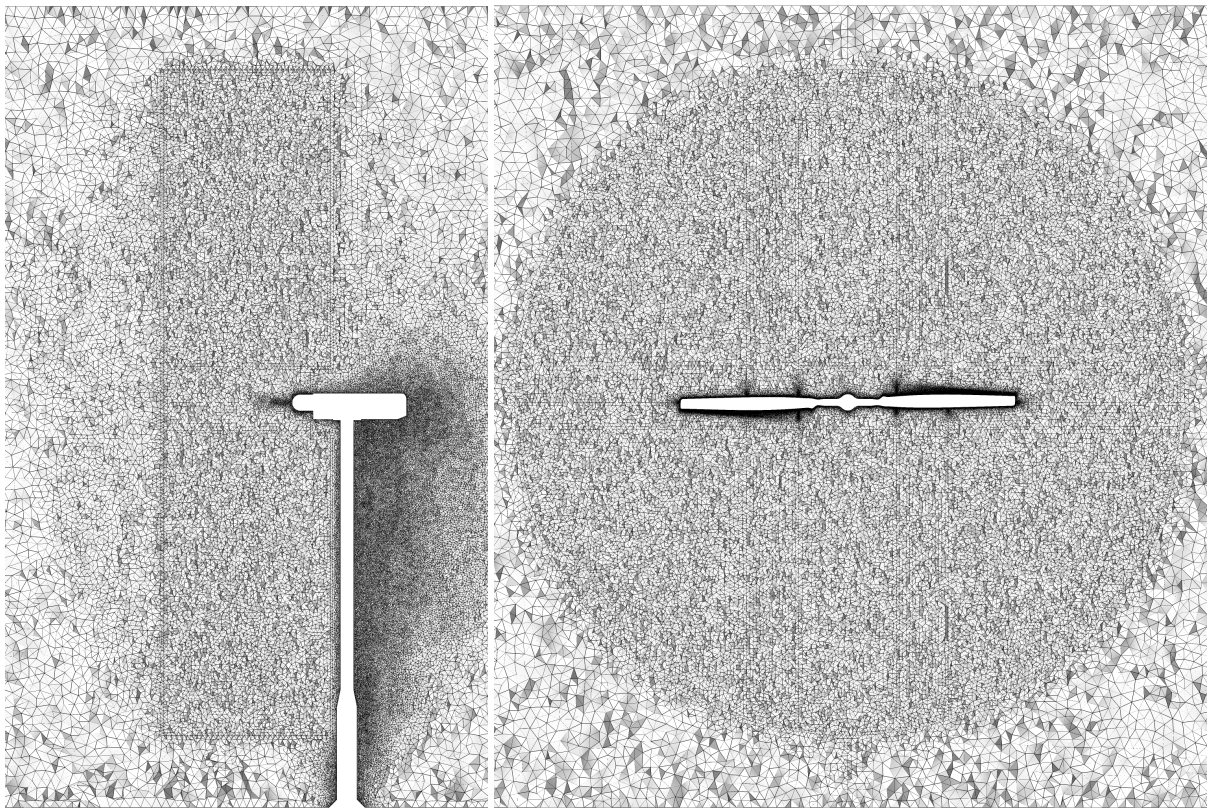


Figure 3.9: Tetrahedrals on the cutting planes $x = 0$ (left) and $y = 0$ through the meshed fluid domain with focus on the rotating domain

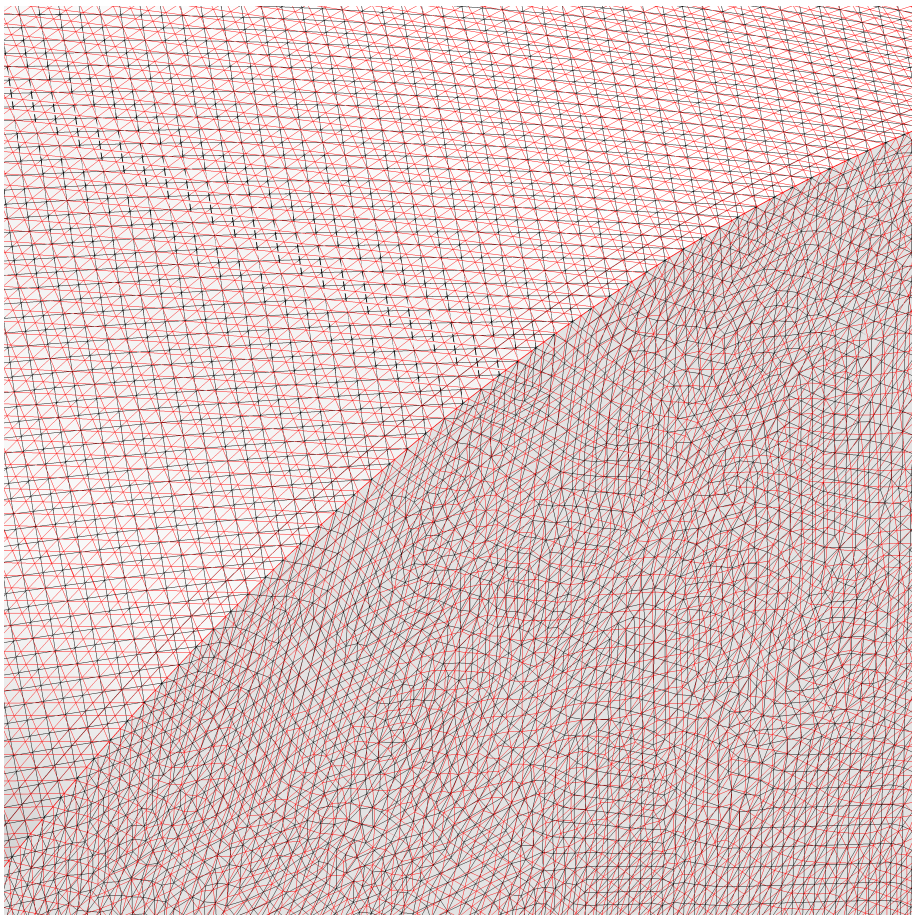


Figure 3.10: Non-matching grids of the rotating (red) and static domain at the Arbitrary Mesh Interface (AMI)

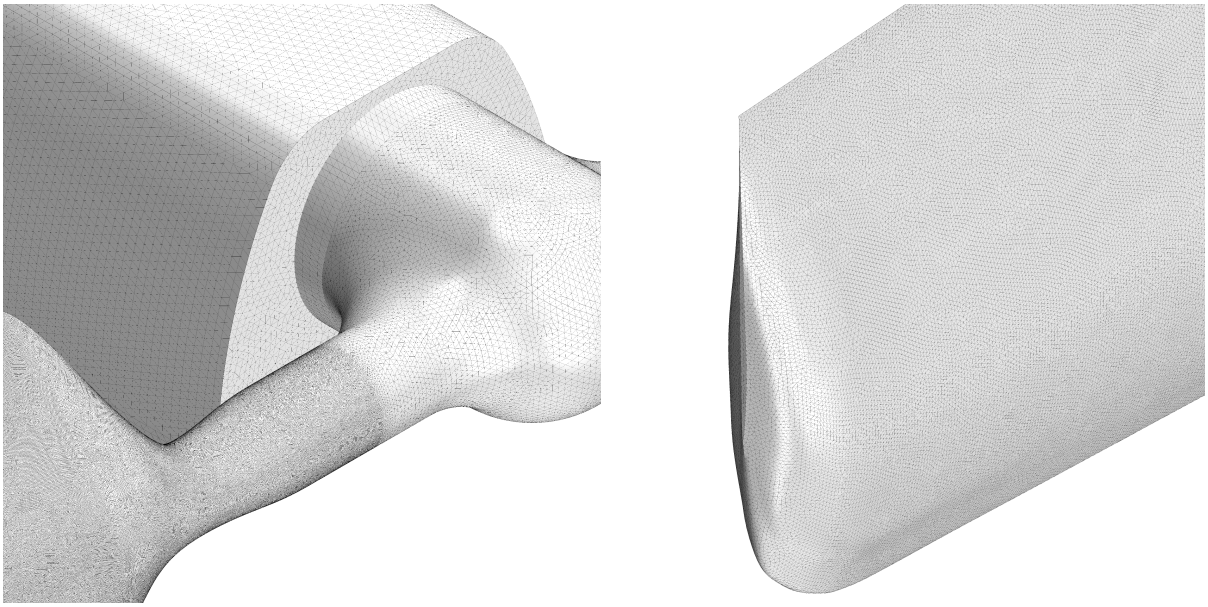


Figure 3.11: Comparison between the surface meshes of blade 1, hub and tower (left) and detailed view of the surface mesh at the tip of blade 1

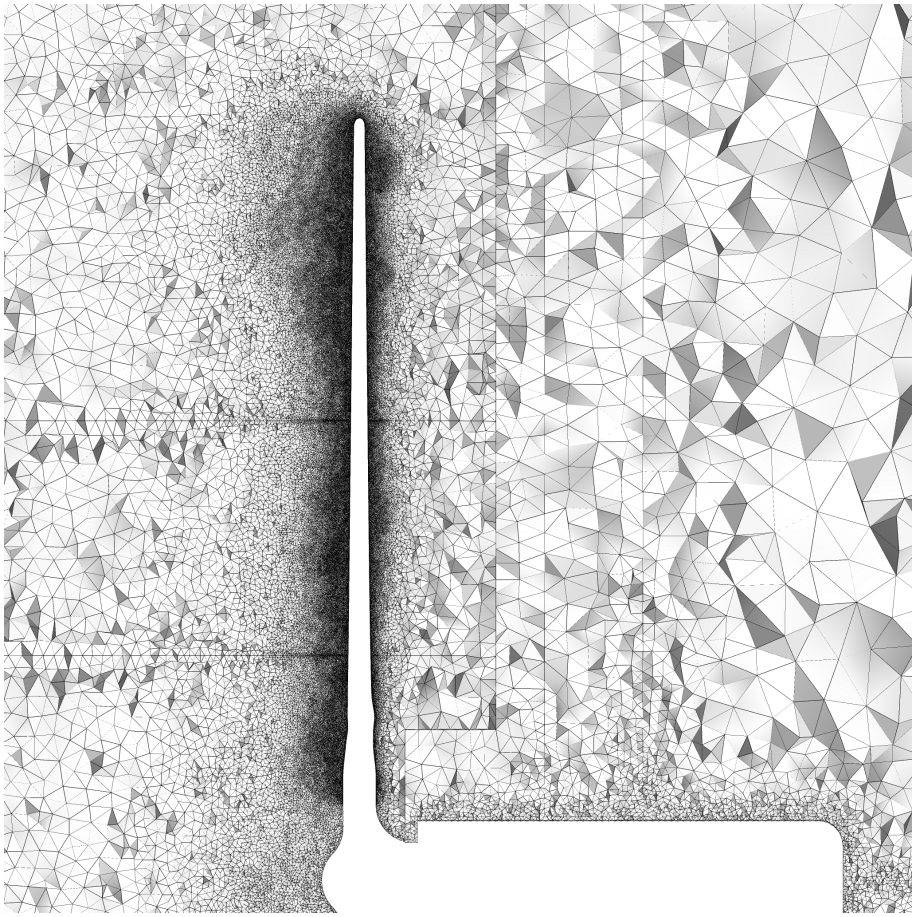


Figure 3.12: Tetrahedrals on the cutting plane $z = 0$ through the meshed fluid domain with focus on blade 1. Both horizontal lines are due to decomposing of the fluid domain for parallel meshing. The other line is the Arbitrary Mesh Interface (AMI)

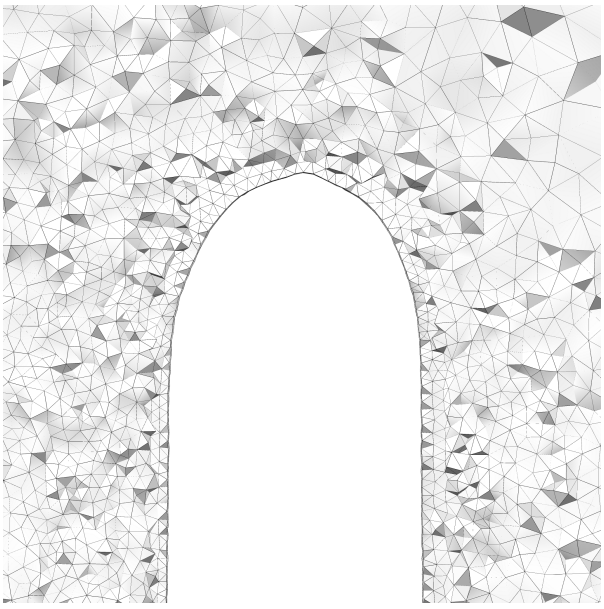


Figure 3.13: Tetrahedrals on the cutting plane $z = 0$ through the meshed fluid domain with focus on the cells at the tip of blade 1

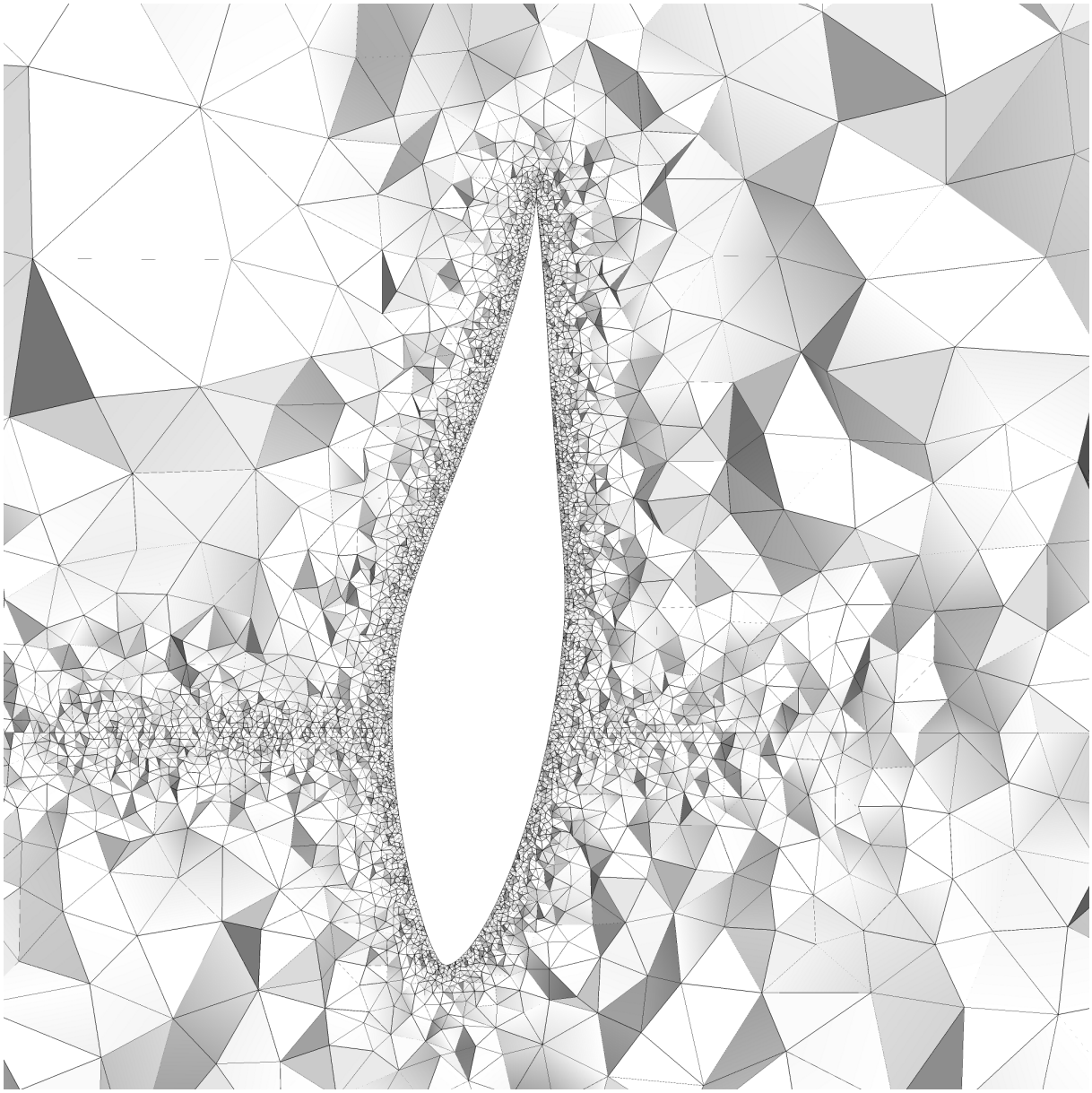


Figure 3.14: Tetrahedrals on the cutting plane $x = 2.5145$ m (50 % radius) through the meshed fluid domain with focus on the volume mesh around blade 1. The horizontal line is due to decomposing of the fluid domain for parallel meshing

preserved. The mesh deformation can be influenced by setting the diffusivity value, which describes the mesh "material's" stiffness.

So for the steps 3/5 to 5/5 the rotor behavior will be realized as follows: The whole fluid domain is split into a static and a rotating one using the AMI concept. The rpm rate of the rotating domain arises from the rotational speed of the rotor. In consequence the rotor is standing still inside the rotating domain and the ALE concept is applied to realize the deformation of both blades.

The implementation of this concept in OpenFOAM requires some manipulations. Normally OpenFOAM can only handle one feature, AMI or ALE, at a time. The `dynamicMeshDict` for a pure AMI case is seen in Listing 3.3 on page 45, while here the edition for a pure ALE case is printed.

Listing 3.7: $EMPIRE_CASE/OF/constant/dynamicMeshDict (ALE only)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dynamicFvMesh   dynamicMotionSolverFvMesh;

motionSolverLibs  ("libfvMotionSolvers.so");

solver          displacementLaplacian;

diffusivity     uniform;

// ************************************************************************* //
```

For the presented concept a combination of both features is required. This was realized by implementing a new dynamic mesh solver called `CoSimulationMotionSolverFvMesh`.

Listing 3.8: $EMPIRE_CASE/OF/constant/dynamicMeshDict (AMI + ALE)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dynamicFvMesh   CoSimulationMotionSolverFvMesh;

motionSolverLibs ("libfvMotionSolvers.so");

CoSimulationMotionSolverFvMeshCoeffs
{
    cellZone                cellZone_1; // Rotor
```

```
    solidBodyMotionFunction  rotatingMotion;
    rotatingMotionCoeffs
    {
        CofG            (0 0 0);
        radialVelocity  (0 432 0); // in deg/s
    }
}

solver           displacementLaplacian;

diffusivity      uniform;

frozenDiffusion  off;

frozenPointsZone  frozen;

// ************************************************************************* //
```

So all AMI parameters can be set next to ALE settings. At this stage a constant rpm rate value is expected, what will later be changed to reading the actual value out of the coupling at each time. For the ALE method only the `displacementLaplacian` solver is selectable. It represents a simplified "material" model, which obeys on a simple Laplacian equation. For setting the local mesh stiffness different diffusivity models, implemented in OpenFOAM, can be used. They will be discussed in detail at the end of this section.

Next to adding the `dynamicMeshDict` dictionary to the `$EMPIRE_CASE/OF/constant/` folder additional changes have to be made. Boundary and initial conditions for the mesh deformation have to be set creating the `$EMPIRE_CASE/OF/0/pointDisplacement` file. For solving the coupled simulation an extended version `pimpleDyMFsiFoam` of the normal `pimpleDyMFoam` solver was implemented. It includes the EMPIRE API for connection during the coupling procedure. This way point displacements for a in the `$EMPIRE_CASE/OF/constant/EmpireDict` defined boundary patch are received and corresponding surface forces are sent. The `$EMPIRE_CASE/empireOF.xml` input file is also necessary. The `topoSetDict` presented for the pure AMI case located in the `$EMPIRE_CASE/OF/system/` folder also has to be extended. Defined is a fixed amount of points, that must not be moved during the simulation. These must contain all Stator and AMI points.

Listing 3.9: $EMPIRE_CASE/OF/system/topoSetDict (AMI + ALE)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    AMI;
        type    faceSet;
        action  new;
        source  patchToFace;
        sourceInfo
```

```
26          {
27              name     "AMI.*";
28          }
29      }
30      {
31          name    frozen;
32          type    pointSet;
33          action  new;
34          source  cellToPoint;
35          sourceInfo
36          {
37              set      cellZone_2; // Stator
38              option   all;
39          }
40      }
41      {
42          name    frozen;
43          type    pointSet;
44          action  add;
45          source  faceToPoint;
46          sourceInfo
47          {
48              set      AMI;
49              option   all;
50          }
51      }
52  );
53
54  // ************************************************************************* //
```

A `pointSet` named frozen is created, which must be transformed to a `pointZone` afterwards, since this type is required by the `pimpleDyMFsiFoam` solver. It is done via executing `setsToZones -noFlipMap` after `topoSet`. The command to consider a "frozen" `pointZone` must be set at the end of the `dynamicMeshDict` file. The entire case structure for step 3/5 including all files is seen in Appendix E.

To document the effects of all available diffusivity models, OpenFOAM's documentation does not provide this, detailed tests with a simplified step 3/5 case were done. It was called own3DPropeller-CSMDummy. The rotor geometry was simplified to a rod completely located inside the Rotor domain and the tower was removed. The extreme coarse mesh was built using OpenFOAM's `blockMesh` utility. So a clear structured mesh was realized providing the option of simpler visual evaluation. Similar dimensions were used to guarantee a basic transferability of obtained results to the right step 3/5 case. The deformation of the "rotor" was simplified to a complete translation in y-direction, means a constant value for all points of maximum 1 m $\hat{=}$ 100 % was sent back by the meshClientTurbomachinery. Therefore the out commanded parts of AbstractDataCreator.h seen in Appendix C have been activated. Results are discussed in the Figures 3.15, 3.16 and 3.17. It should be noticed, that the views are limited to the rotational domain and that simulations crashed before reaching the maximum deformation, if $\Delta y/y$ values are smaller than 100 %.

3.2.4 Turbulence Modeling and Evaluation of Surface Forces etc.

Turbulence is modeled using Reynolds-averaged Navier Stokes equations. This is better known as RANS turbulence model. In OpenFOAM it can be selected by setting `RASModel` as `simulationType` in the `$EMPIRE_CASE/OF/constant/turbulenceProperties` file. RAS is short for Reynolds-averaged Simulations, OpenFOAM's name for the RANS turbulence models. OpenFOAM offers a large

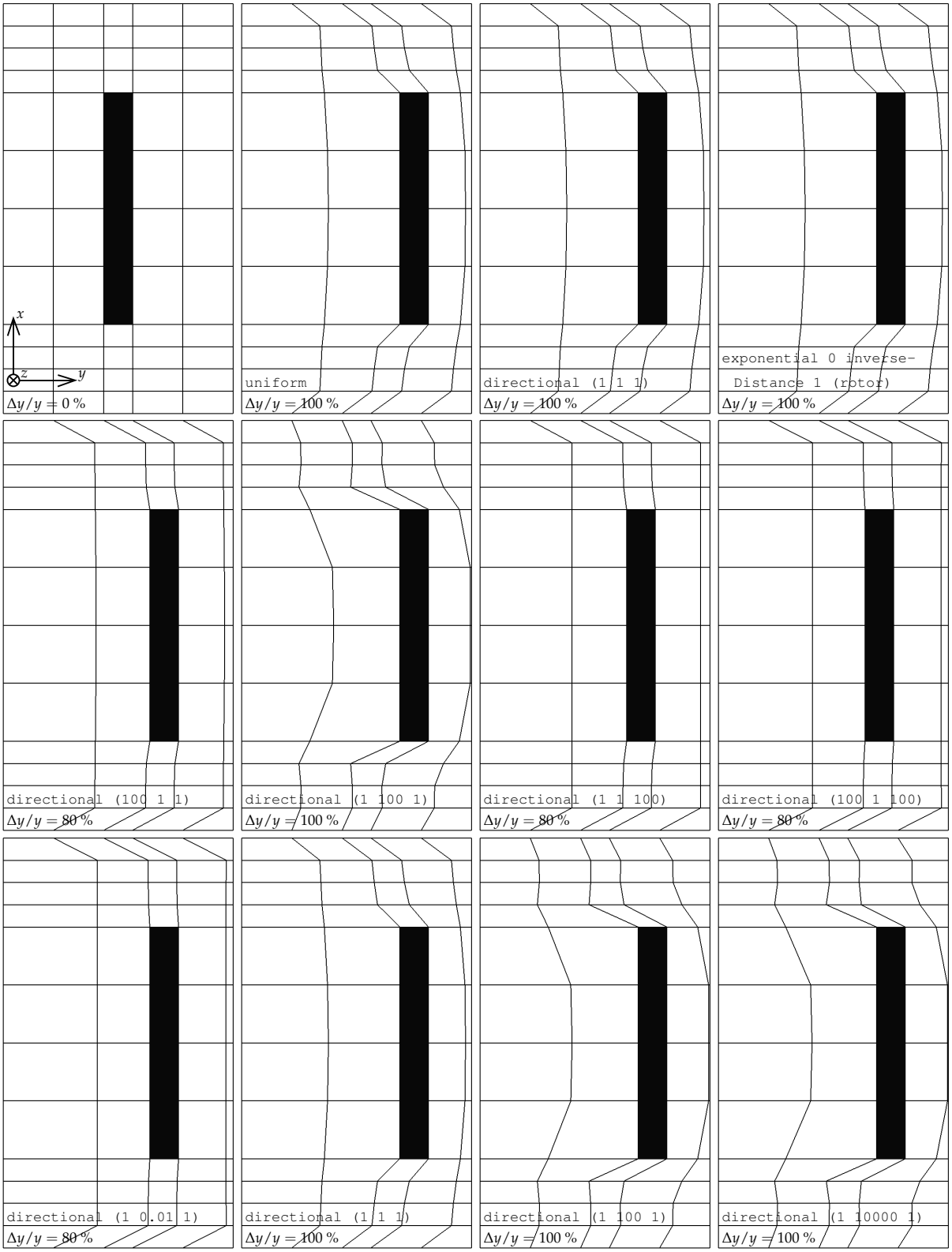


Figure 3.15: Results of own3DPropellerCSMDummy for testing available diffusivity models in OpenFOAM (1):
Line 1 shows `uniform`, `directional (1 1 1)`, `exponential 0 inverseDistance 1 (rotor)` and `exponential 0 inverseDistance 1 (stator)` are similar. Line 2 points out `directional` coefficients in x- or z-direction have almost the same effect, since main motion direction is y. The effect of increased `directional` coefficients in y-direction can be seen in line 3

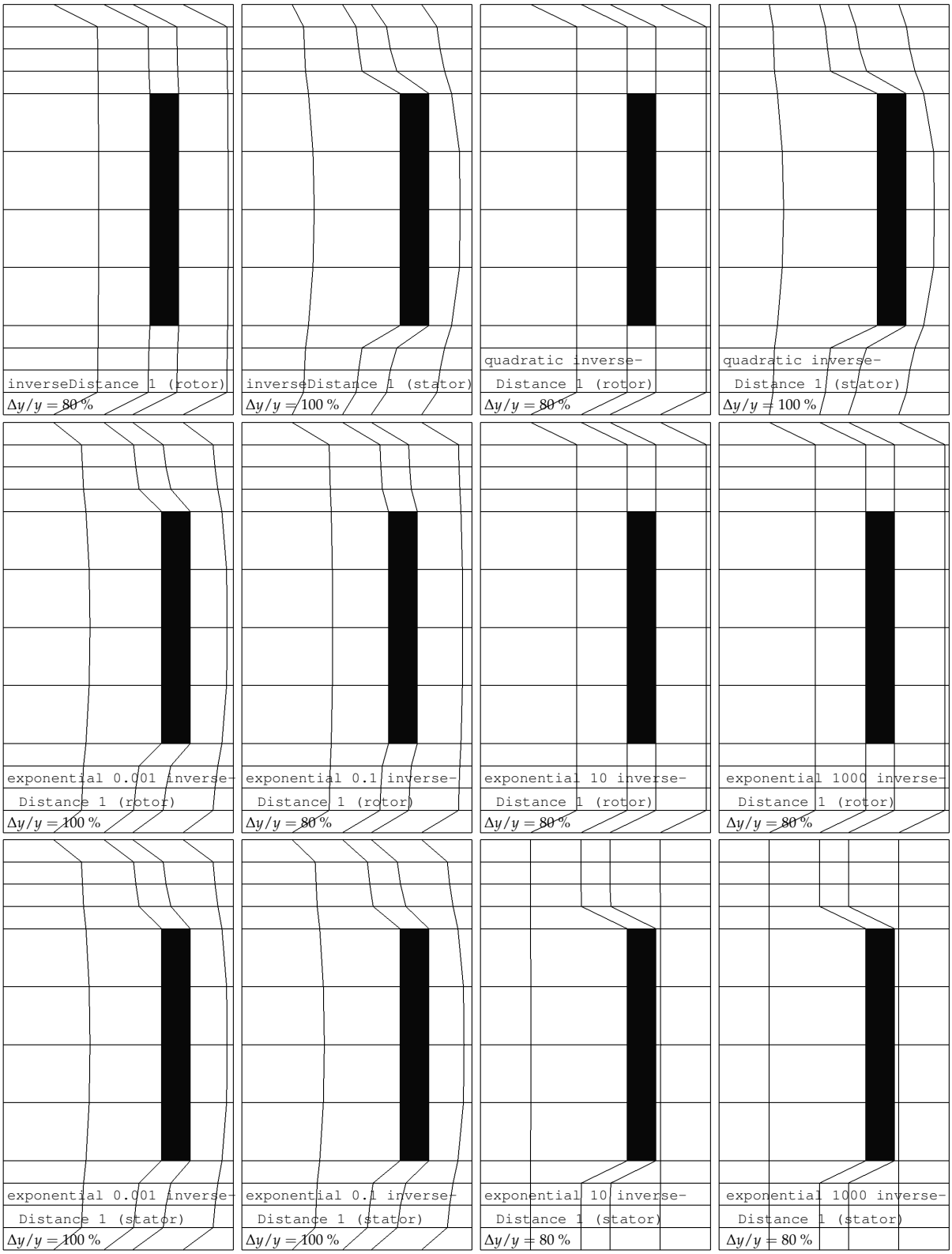


Figure 3.16: Results of own3DPropellerCSMDummy for testing available diffusivity models in OpenFOAM (2):
Line 1 shows the differences between `linear` or `quadratic inverseDistance` to `rotor` or `stator`. The effect of increased `exponential` coefficients to `rotor` or `stator` is seen in lines 2 and 3

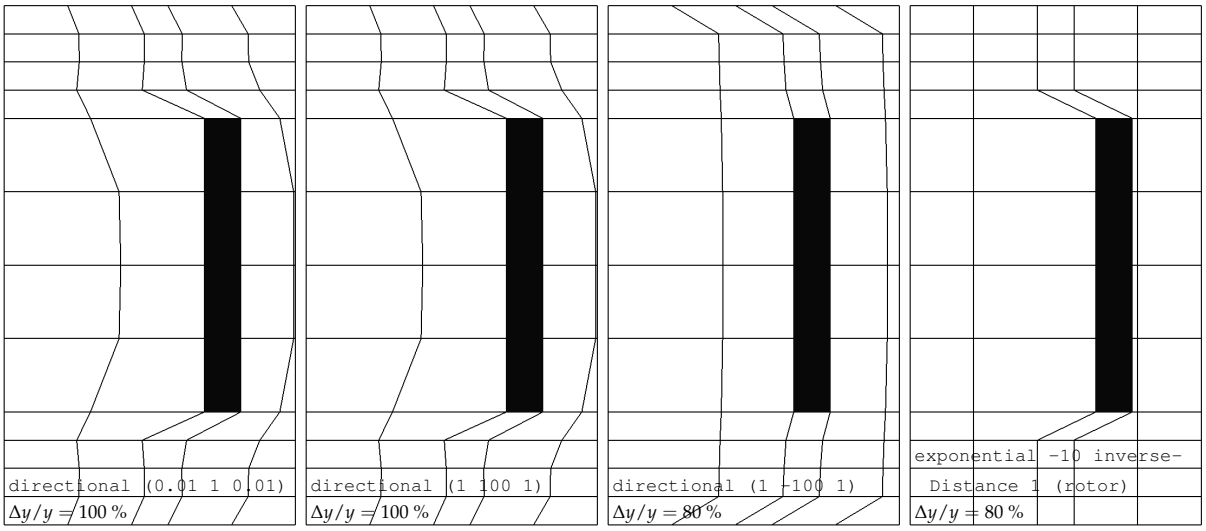


Figure 3.17: Results of own3DPropellerCSMDummy for testing available diffusivity models in OpenFOAM (3):
With `directional` only relative differences between values have any effect. Using negative values is inverting the effects

library of RANS turbulence models, summarized e.g. on `http://www.openfoam.com/features/RAS.php`. For this purpose the `RASModel kOmegaSST` was selected via the `RASProperties` file located in the `$EMPIRE_CASE/OF/constant/` folder. Detailed information to the RANS k-ω-SST turbulence model can be found on `http://www.cfd-online.com/Wiki/SST_k-omega_model` or in [34, p. 91 ff].

Initial and inlet boundary values for k and ω are required for the simulations. It can be red in [6], that the "Turbulence intensity (streamwise component) is typically no greater than 0.5 %". Referencing to [1] the turbulence intensity I is defined with the root-mean-square of the velocity fluctuations u' and the mean free stream velocity $\overline{u}$.

$$I = \frac{u'}{\overline{u}} \tag{3.1}$$

The turbulence length scale l is a physical quantity for the size of large eddies in turbulent flows. It can be calculated via an empirical relationship containing a characteristic length L, which for internal flows can be set as hydraulic diameter d_h. For the rectangular shaped wind tunnel the hydraulic diameter is

$$d_h = \frac{4A}{U} = \frac{2bh}{b+h} = 28.87 \text{ m} \tag{3.2}$$

This results in a turbulence length scale of

$$l = 0.07L = 0.07d_h = 2.021 \text{ m} \tag{3.3}$$

For standard RANS turbulence models and variations the turbulent kinetic energy k reads

$$k = \frac{3}{2}(\overline{u}I)^2 = 0.0009375\ \frac{\mathrm{m}^2}{\mathrm{s}} \tag{3.4}$$

and the turbulent dissipation rate ε

$$\varepsilon = \frac{0.1643k^{1.5}}{l} = 0.000002334\ \frac{\mathrm{m}^2}{\mathrm{s}^3} \tag{3.5}$$

With this the specific dissipation rate ω becomes

$$\omega = \frac{\varepsilon}{k} = 0.002489\ \frac{1}{\mathrm{s}} \tag{3.6}$$

for the k-ω-SST model.

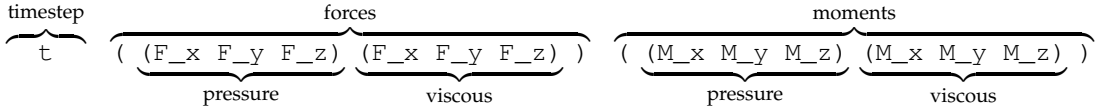
For evaluation of the surface forces due to fluid flow, which are loading parts like blades, tower etc., the run-time post-processing tool forces can be integrated. Therefore the following lines simply have to be added to the `$EMPIRE_CASE/OF/system/controlDict` and adjusted for each patch or group of patches someone is interested in.

Listing 3.10: $EMPIRE_CASE/OF/system/controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

[...]

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

functions
{
    forcesRotor
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches ("rotor.*");
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 0 0);

            outputControl     timeStep;
            outputInterval    1;
    }

    forcesStatorTower
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches (statorTower);
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 -1.401 -12.164); // with reference to root
```

```
45
46              outputControl     timeStep;
47              outputInterval    1;
48      }
49  }
50
51  // ************************************************************************* //
```

In this example folders `$EMPIRE_CASE/OF/forcesRotor/` and `$EMPIRE_CASE/OF/forcesStatorTower/` will be created storing the output files for the group of patches `"rotor.*"`, which represents the entire rotor, and `statorTower`. The structure of the output files is as follows.



Additional interesting tools are `yPlusRAS` for evaluating the non-dimensional wall distance $y+$ and `Co` for calculating the Courant-Friedrichs-Lewy (CFL) number for each time step.

3.3 Extracts of the Simulation Results

3.3.1 Step 1/5 – Steady-state, Pure CFD Simulation of Quasi-Rotating Turbine

Detailed descriptions of simulation step 1/5 concerning structure and aims have been given in previous sections. The simulation was run on 12 cores $\times$ 12 nodes $=$ 144 cores on the CCCSA[1] using a mesh similar to the presented one. The rotating domain is exactly the same, while the static one only differs in the used CAD geometries (cylindrical "wind tunnel" and no tower) but uses identical meshing parameters. The full case structure (folders and files) and all OpenFOAM dictionaries containing initial and boundary conditions as well as all other settings are printed in Appendix E. 5840 iterations were done until convergence was reached. Only the last iteration output was reconstructed and stored.

Listing 3.11 shows the first and last lines of the file written by OpenFOAM's runtime post-processing tool forces.

Listing 3.11: $EMPIRE_CASE/OF/forcesRotor/0/forces.dat

```
1 # Time  forces(pressure, viscous) moment(pressure, viscous)
2 1 (((-21.50751 -407778.1 -94.26324) (-1.554737e-05 -1.336612 0.0005010323)) ((-185.1514 -425763.5 1764.514)
      (-6.500869e-05 -15.81094 -0.001081759)))
3 2 (((5.459842 118247.4 11.62695) (-0.008794198 12.49909 0.03556726)) ((53.46144 125826.1 -546.488)
      (0.001023978 -0.9929004 0.01372964)))
4 3 (((-910.7024 1358483 831.3459) (-0.0002530239 12.53929 0.07043972)) ((189.6956 1191757 -389.2877)
      (-0.002169577 5.555696 0.02445003)))
5 4 (((-357.4434 900380.7 872.7153) (-0.003767295 4.285001 0.05499132)) ((18.0778 770277.8 476.1936)
      (-0.001455948 2.425854 0.004979752)))
6 5 (((320.8226 -345080.4 -467.9927) (-0.003950646 -2.576337 0.03672611)) ((-89.18284 -293975.1 917.7759)
      (-0.003454887 -5.498579 0.01152773)))
7 6 (((325.7734 -501690.6 910.3741) (0.006675572 -2.24143 0.0291424)) ((-41.54943 -420252.6 -1486.756)
      (-0.001038873 -9.284118 0.01715174)))
8 7 (((1.663545 58426.17 439.9702) (0.009782588 1.050038 0.02483369)) ((12.93046 51985.52 -858.3709)
      (-0.000437157 -8.298967 0.01659759)))
9 8 (((-94.54208 227761.4 -475.0339) (-0.001013052 2.034458 0.0264744)) ((26.82292 191461.1 588.689)
      (-0.00134011 -6.729351 0.002500129)))
```

[1]The CCCSA is the Compute Cluster of the Chair for Structural Analysis. It has 14 compute nodes, each with two Sockets (2x Intel E5-2667). As interconnect a QDR InfiniBand is used. The E5-2667 is a six-core Sandy Bridge-EP CPU, hence 168 CPUs are available in total.

```
9 (((8.538126 6873.662 -426.8004) (-0.003785352 0.9519546 0.03009518)) ((2.139069 7192.961 595.9179)
    (-0.002218444 -7.1538 -0.004722653)))
10  (((24.58096 -101661 308.4132) (-0.006033731 0.2426372 0.03315586)) ((-7.558577 -81815.69 -320.4547)
    (-0.002294528 -8.323915 0.00508023)))

[...]

5831  (((0.2150052 -860.6512 -1.492614) (-0.0003713517 -0.8254131 0.008689564)) ((0.09669951 375.5506
    -2.616494) (-0.0008372611 -17.68448 0.0005596755)))
5832  (((0.2149524 -860.6504 -1.493572) (-0.0003699706 -0.8254171 0.008686745)) ((0.09698376 375.5509
    -2.618259) (-0.0008370657 -17.68449 0.0005563564)))
5833  (((0.2148756 -860.6495 -1.494397) (-0.0003685268 -0.8254214 0.008685381)) ((0.09727129 375.5509
    -2.619341) (-0.0008367502 -17.68448 0.0005564307)))
5834  (((0.2147891 -860.6488 -1.495193) (-0.0003662133 -0.8254253 0.008681771)) ((0.09757823 375.5504
    -2.619312) (-0.000836447 -17.68448 0.0005565244)))
5835  (((0.2147028 -860.6479 -1.496016) (-0.0003644576 -0.8254287 0.008679785)) ((0.09787465 375.5501
    -2.619223) (-0.0008360923 -17.68448 0.0005586451)))
5836  (((0.2146228 -860.6472 -1.496888) (-0.0003622476 -0.8254317 0.008677596)) ((0.0981603 375.5497
    -2.619429) (-0.0008358709 -17.68448 0.0005585455)))
5837  (((0.2145442 -860.6461 -1.497666) (-0.0003601949 -0.8254349 0.008675997)) ((0.09841614 375.5492
    -2.620731) (-0.0008357033 -17.68447 0.0005554207)))
5838  (((0.2144668 -860.6452 -1.498235) (-0.0003570651 -0.8254377 0.008673849)) ((0.09866723 375.5491
    -2.622227) (-0.0008353555 -17.68447 0.0005599744)))
5839  (((0.2144065 -860.6444 -1.498834) (-0.0003556541 -0.8254399 0.008670852)) ((0.09894338 375.5489
    -2.622553) (-0.0008351773 -17.68446 0.0005628843)))
5840  (((0.2143465 -860.6435 -1.499433) (-0.0003526077 -0.8254437 0.008669539)) ((0.09921569 375.5485
    -2.622792) (-0.000835019 -17.68446 0.0005643102)))
```

The measurement data of NREL's experiments (cf. Fig. B.1 and B.2, Appendix B, pp. 80) are summarized to four representative values: minimum, maximum, mean value and standard deviation. This is necessary, because the real time history deals with big fluctuations, which can not be matched to a specific position of the wind turbine rotor. So the created range between minimum and maximum will be compared to the simulation result for the low speed shaft torque. It is calculated via summation of M_y due to pressure forces and M_y due to viscous forces. Figure 3.18 contains all results. Deviations of 20.5217 % resp. 21.5750 % from the mean values were achieved.

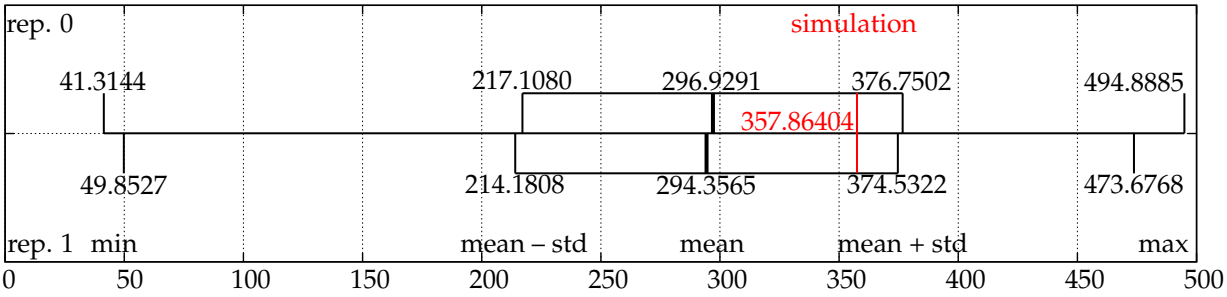


Figure 3.18: NREL's experimental data for the low speed shaft torque vs. simulation output of runtime post-processing tool forces (step 1/5, `MRFSimpleFoam`, 5840 iterations)

To indicate the reaching of right simulation results a simple visual validation was done at first. Therefore simulated pressure and velocity fields at 80 % radius = 4.0232 m presented in [17] were compared to the own ones. To be able to do this, the same color legend including pressure and velocity ranges was set. OpenFOAM uses for its calculations pressure normalized with the density $\left(p^* = \frac{p}{\rho}\right)$, so values had to be converted first.

$$p^*_{\text{min}} = \frac{p_{\text{min}}}{\rho} = \frac{-0.3\ \text{kPa}}{1.23\ \frac{\text{kg}}{\text{m}^3}} \approx -244\ \frac{\text{m}^2}{\text{s}^2} \qquad p^*_{\text{max}} = \frac{p_{\text{max}}}{\rho} = \frac{0.25\ \text{kPa}}{1.23\ \frac{\text{kg}}{\text{m}^3}} \approx 203\ \frac{\text{m}^2}{\text{s}^2} \tag{3.7}$$

Furthermore the `MRFSimpleFoam` solver outputs absolute velocity values $\mathbf{u}_{\text{abs}}$. But relative velocity values $\mathbf{u}_{\text{rel}}$ are required for the comparison, so the velocity field also had to be converted first using the "Calculator" filter in `paraFoam`.

$$\begin{aligned} \mathbf{u}_{\text{rel}} &= \mathbf{u}_{\text{abs}} - (\boldsymbol{\omega} \times \mathbf{x}) \quad \text{with} \quad \boldsymbol{\omega} = \left(0, 7.5489\ \frac{\text{rad}}{\text{s}}, 0\right)^T \\ \Rightarrow \quad |\mathbf{u}_{\text{rel}}| &= \sqrt{(u_{\text{abs}} - \omega_y z)^2 + v_{\text{abs}}^2 + (w_{\text{abs}} + \omega_y x)^2} \end{aligned} \tag{3.8}$$

Figure 3.19 shows the results.

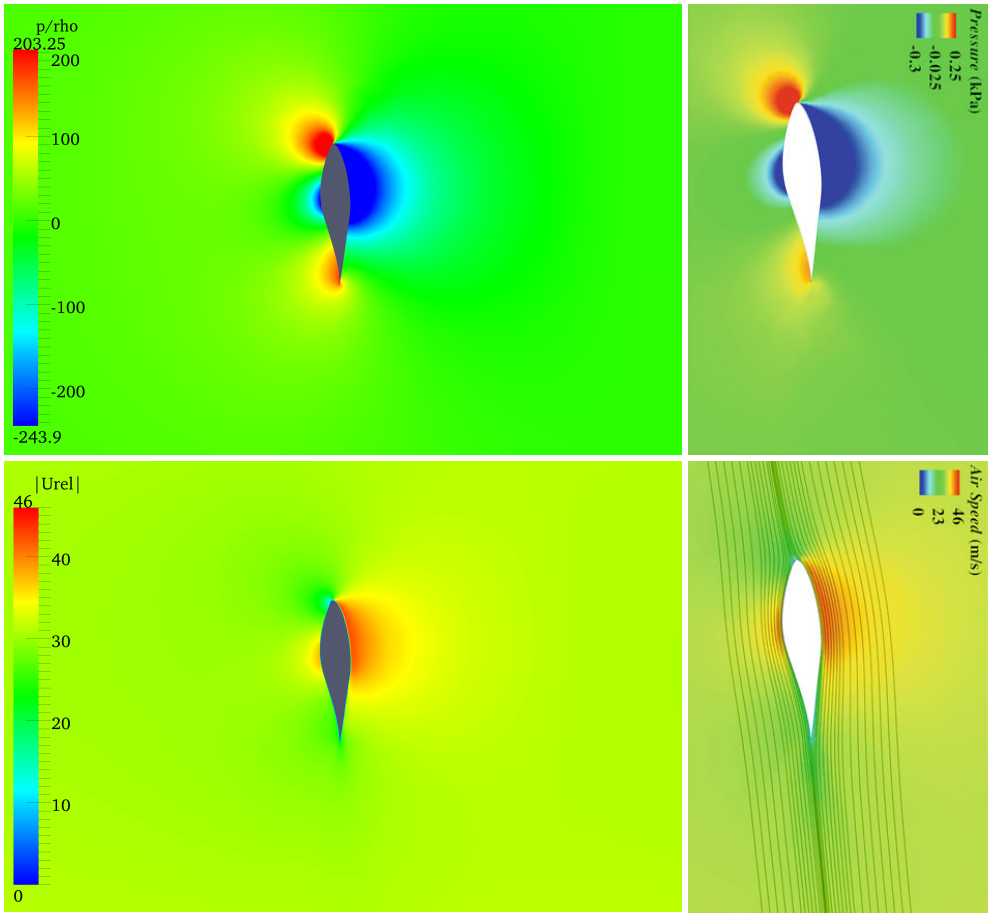


Figure 3.19: Pressure and velocity field around a blade at 80 % radius = 4.0232 m: Own simulation results (step 1/5, `MRFSimpleFoam`, 5840 iterations) vs. results taken fron [17] (left)

Most importance comes to evaluation of the normalized pressure coefficients c_p. Measurement positions are illustrated in Figure 2.15 on page 36. To extract the appropriate data from the simulation results, some semiautomatic manipulations have to be done. Cutting planes on the corresponding radial positions are used to export the surface pressure values (only patch blade 1 is enabled) from `paraFoam` to

.csv files. These files are used to calculate c_p over % chord with a MATLAB script. Normalized was with relation to the maximum achieved value.

$$c_p = \frac{p_{\text{cut}}}{\max\left(p_{\text{cut}}\right)} \tag{3.9}$$

Measurement data is illustrated again via minimum, maximum, mean value and its standard deviation. As an example the results for position 46.6 % radius $=$ 2.3435 m can be seen in Figure 3.20, while Appendix F holds figures for all five radial positions.

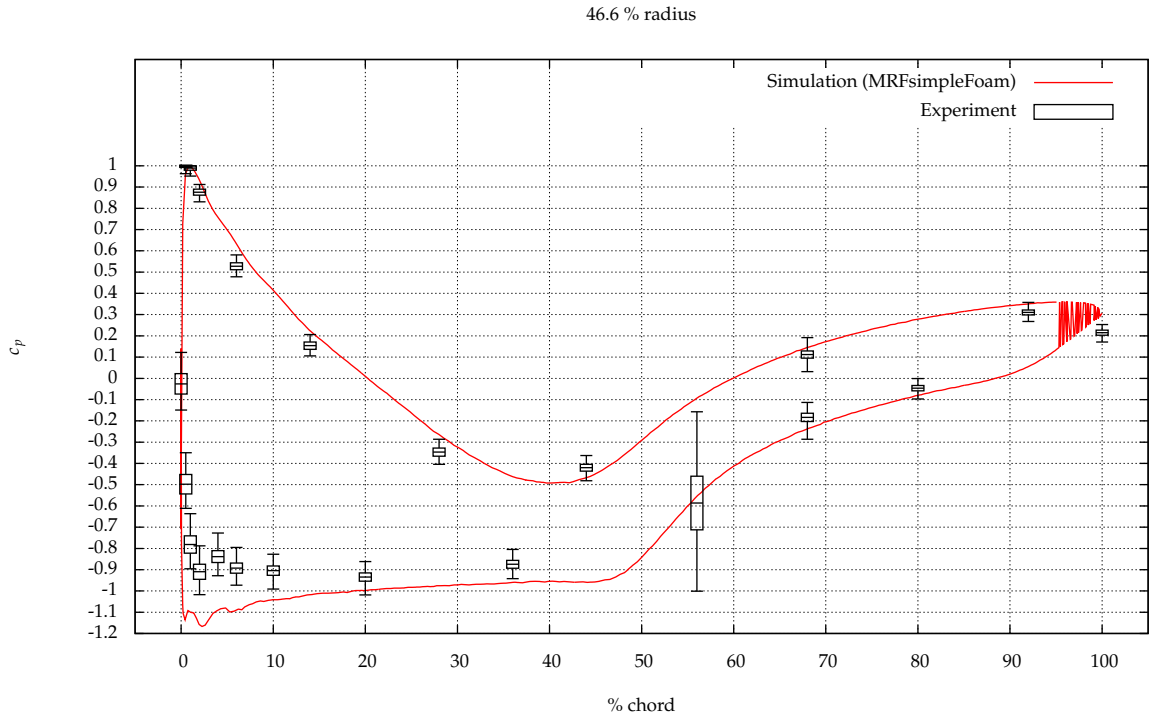


Figure 3.20: Normalized pressure coefficients at 46.6 % radius $=$ 2.3435 m: Simulation results (step 1/5, `MRFSimpleFoam`, 5840 iterations) vs. NREL UAE Phase VI (sequence H, yaw angle 0°, wind tunnel velocity 5 $\frac{\text{m}}{\text{s}}$, repetition 0) measurements. Measurement data is represented by minimum, maximum and mean value with its standard deviation

It should be noticed, that upper (lower) values represent the lower (upper) blade surface, since here overpressure (vacuum) dominates and results in positive (negative) c_p values. Furthermore could the big min.-max. range at 56 % upper surface be due to break-off in this region.

This last evaluation is used for further investigations with different mesh and solver settings, which are still in progress. The coarsest possible mesh with still correct pressure field should be figured out this way.

3.3.2 Step 2/5 – Transient, Pure CFD Simulation of Full, Rotating Turbine

A transient, pure CFD simulation of the full wind turbine rotating at constant rpm rate, step 2/5, was introduced in detail above. Aims of this simpler simulation were also discussed. The necessary file

structure for OpenFOAM is stored in Appendix G. It is the same as for the following two steps, since `pimpleDyMFoam` simply ignores components needed for mesh deformation. At this stage of the development of a fluid-structure-signal co-simulation step 2/5 was only run with different other meshes than the presented one. Reason was to simply validate the functionality with all linked features at an earlier point of time. Furthermore a nominal wind speed of 15 $\frac{\mathrm{m}}{\mathrm{s}}$ was used that time. So usable results have not been produced, because the used meshes simply were too coarse. In consequence the obtained low speed shaft torque differs enormous from NREL's measured values. This is due to a too large non-dimensional wall distance $y+$ in keeping with wrong modeling of the boundary layer effects and of stall effects occurring at this speed.

Next to the proof of functionality one interesting flow phenomenon can be shown applying the simulation results: At the tip of both rotating blades vortices are developing, which get carried downstream with the main flow. This phenomenon is visualizable by generating an isosurface for a velocity value slightly bigger than the mean free stream value. Figure 3.21 shows the resulting helix, an isosurface for a velocity of about 15.6 $\frac{\mathrm{m}}{\mathrm{s}}$. The full transient development can be seen in Appendix H as timeline or online in the Youtube video `http://www.youtube.com/watch?v=w-a0szPz5vY`. Video and pictures were generated with `paraFoam` using the "Calculator" filter to compute the velocity magnitude for every cell and the "Contour" filter to finally create the isosurface.

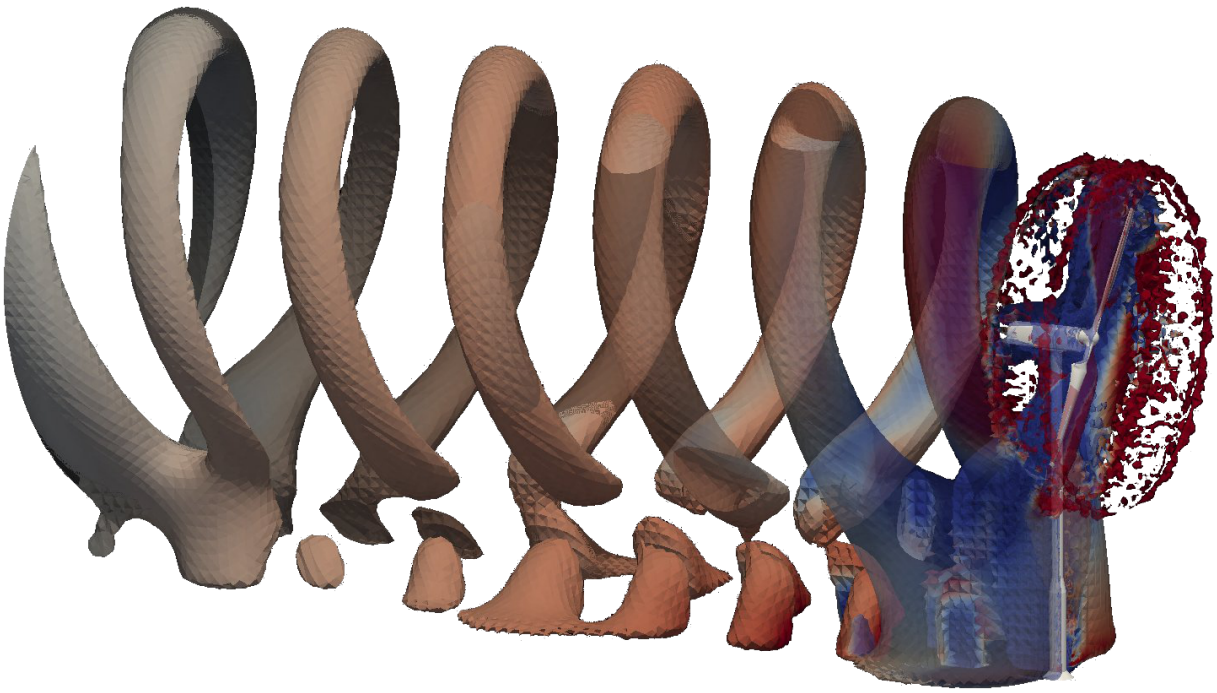


Figure 3.21: Vortices developing at the rotating blades' tip and getting carried downstream with the main flow are visualized using an isosurface of $|\mathbf{u}| \approx 15.6\ \frac{\mathrm{m}}{\mathrm{s}}$. Disturbance due to reflexions on the AMI can also be seen

3.3.3 Step 3/5 – One-Way Coupling with predefined Displacements

Step 3/5, one-way coupling with the substitutional "CSM" client meshClientTurbomachinery to realize predefined displacements independent of acting fluid forces, was also run on older meshes at this stage. They were generated with STAR-CCM+ or OpenFOAM's `snappyHexMesh` tool. Aim was to check different functionalities:

1. Coupling EMPIRE,
2. Mapping of surface forces and displacements between non-matching grids of CFD and "CSM" client,
3. Figuring out the most suitable diffusivity model for realizing mesh deformation,
4. Investigating the parameter range for the diffusivity models and
5. Observing the development of the mesh quality during deformation using the tool `checkMesh`.

Coupling was done after fixing different problems concerning the mapping. The results for point 3 and 4 can be summarized to the statement, that maximum deformation values whereby OpenFOAM crashes are achieved with the diffusivity model `directional (1 100 1)` and an iterative adjusted `exponential inverseDisatance` independent of the used mesh. Only the absolute reached values from few centimeters to over 20 cm are depending. Point 6 states, that fatal mesh errors occur few time steps and deformation values before OpenFOAM finally crashes. So mesh quality has to be monitored during the whole deformation procedure. Just as a footnote the full OpenFOAM file structure with all dictionaries is available in Appendix G.

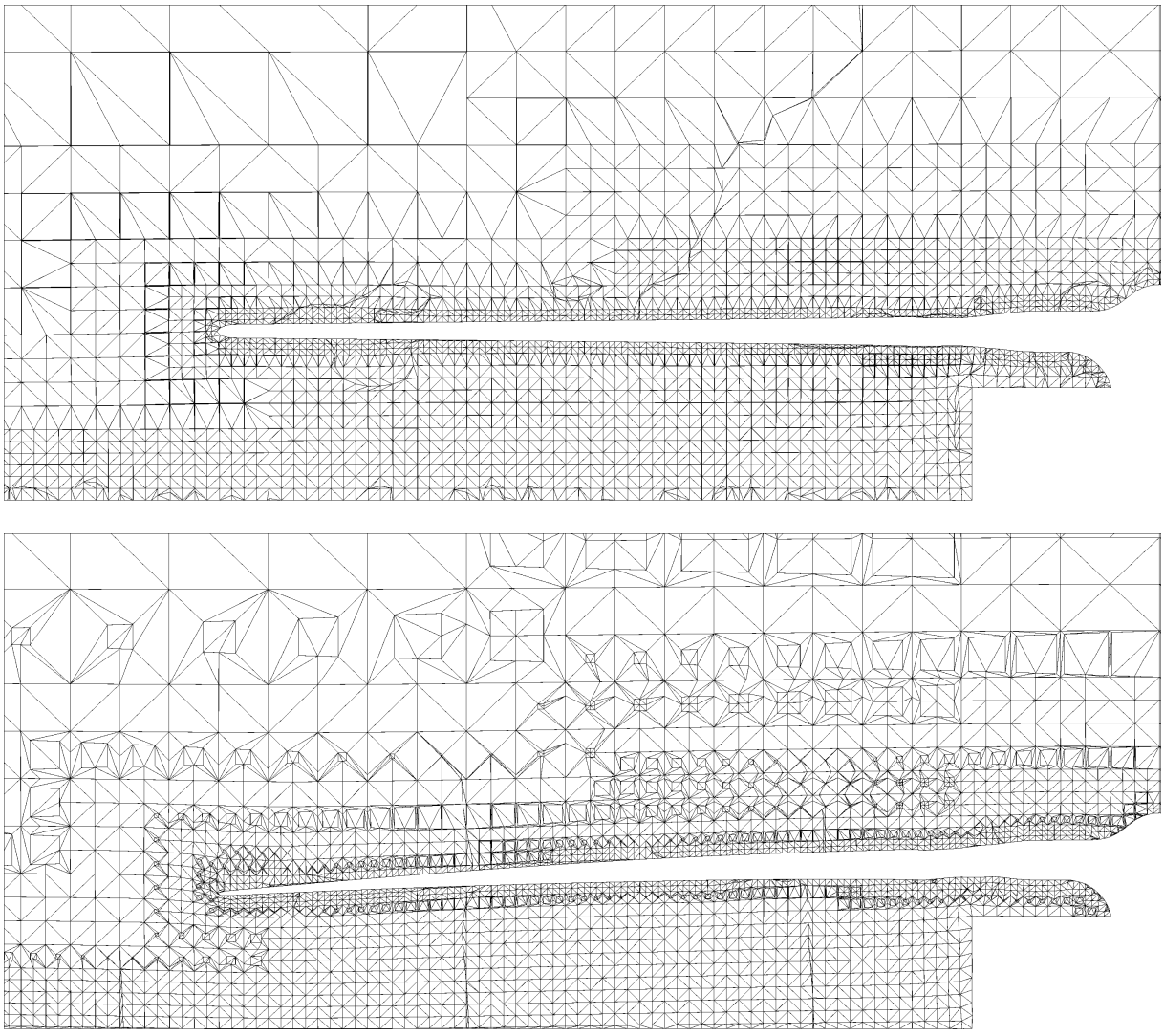


Figure 3.22: Example for parabolic dummy deformation of a blade with meshClientTurbomachinery until OpenFOAM crashes (Mesh from `snappyHexMesh`)

4

Conclusion and Outlook

All applicable features have been developed and assembled to until now a one-way coupling solution for simulating NREL's UAE Phase VI wind turbine at full scale, including a RANS k-ω-SST turbulence model. Critical points, like surface mesh resolution, $y+$ or the used diffusivity model etc., are now well-known and can so be handled with special attention in further developments. The geometry of the full wind turbine was realized using available and estimated data, as well as adaption and evaluation of measurement data available in an online database.

Based on all made achievements and experiences 2D cuts through the blade at selected positions and additional available wind tunnel test datum for the S809 airfoil will now as well as the open source tool Xfoil from the Massachusetts Institute of Technology (MIT) be used to converge surface mesh dimensions and wall functions. Afterwards step 1/5 will be repeated until all parameters and the 3D mesh are set trying to be as coarse as possible and at the same time keeping the pressure field the same and right one. Step 2/5 can be canceled, since it was only for testing the AMI environment, and proceeded next with step 3/5 and 4/5.

Until now all developments were focused on the CFD client, OpenFOAM. Challenges like implementing the CSM client Carat++, adding with MATLAB the third code for gearbox/generator and finally doing the full simulation will follow.

Appendix

A

MATLAB Script for Blade Geometry Generation

The MATLAB script for generating spatial points defining the blade surface is presented. In a next step they are imported in CATIA V5 via Microsoft Excel macro and a part is designed using splines. Necessary input files, like normalized S809 airfoil data, are also appended.

Listing A.1: bladeCrossSections.m

```
%% ------------------------------------------------------------------------ %
clc; clear all;

%% ------------------------------------------------------------------------ %
%  --- settings ------------------------------------------------------------ %
% scale factor
SCFAC = 1000; % here in mm/m
% pitch angle
PITCH = 3;    % in degree

%% ------------------------------------------------------------------------ %
%  --- provide basic data -------------------------------------------------- %
% S809 airfoil
temp = importdata('S809Airfoil.dat');
S809 = temp.data;
clear temp;
% Geometry definition
temp = importdata('geometryDefinition.dat');
GEODEF = temp.data;
clear temp;
% Cylinder "airfoil"
CYL(:,1) = S809(:,1);
CYL(1:31,2) = - sqrt(0.25 - (S809(1:31,1) - 0.5).^2);
CYL(32:62,2) = sqrt(0.25 - (S809(32:62,1) - 0.5).^2);

%% ------------------------------------------------------------------------ %
%  --- generate and export geometries -------------------------------------- %
figure;
for i = [1:length(GEODEF)]
    if GEODEF(i,2) == 0
        GEO = CYL;

        GEO(:,3) = SCFAC * GEODEF(i,4) * (GEO(:,1) - GEODEF(i,7));
        GEO(:,2) = SCFAC * GEODEF(i,4) * (-GEO(:,2));
        GEO(:,1) = SCFAC * GEODEF(i,3);

        temp = GEO(:,3);
```

```
        GEO(:,3) = GEO(:,3) * cosd(GEODEF(i,6) + PITCH) + GEO(:,2)...
            * sind(GEODEF(i,6) + PITCH);
        GEO(:,2) = - temp * sind(GEODEF(i,6) + PITCH) + GEO(:,2)...
            * cosd(GEODEF(i,6) + PITCH);
        clear temp;

    elseif GEODEF(i,2) == 1
        GEO = S809;

        GEO(:,3) = SCFAC * GEODEF(i,4) * (GEO(:,1) - GEODEF(i,7));
        GEO(:,2) = SCFAC * GEODEF(i,4) * (-GEO(:,2));
        GEO(:,1) = SCFAC * GEODEF(i,3);

        temp = GEO(:,3);
        GEO(:,3) = GEO(:,3) * cosd(GEODEF(i,6) + PITCH) + GEO(:,2)...
            * sind(GEODEF(i,6) + PITCH);
        GEO(:,2) = - temp * sind(GEODEF(i,6) + PITCH) + GEO(:,2)...
            * cosd(GEODEF(i,6) + PITCH);
        clear temp;

    elseif GEODEF(i,2) == 2
        GEO = S809(1:3:62,:);
        GEO(21,:) = S809(62,:);

        GEO(:,3) = SCFAC * GEODEF(i,4) * (GEO(:,1) - GEODEF(i,7));
        GEO(:,2) = SCFAC * GEODEF(i,5) * (-GEO(:,2));
        GEO(:,1) = SCFAC * GEODEF(i,3);

        temp = GEO(:,3);
        GEO(:,3) = GEO(:,3) * cosd(GEODEF(i,6) + PITCH) + GEO(:,2)...
            * sind(GEODEF(i,6) + PITCH);
        GEO(:,2) = - temp * sind(GEODEF(i,6) + PITCH) + GEO(:,2)...
            * cosd(GEODEF(i,6) + PITCH);
        clear temp;

    else
        disp('error');
    end

    xlswrite('bladeCrossSections.xls',GEO,['crossSection',num2str(i)]);
    %dlmwrite(['bladeCrossSection',num2str(i),'.txt'], GEO, 'delimiter',...
    %    '\t', 'precision', 9);
    hold on;
    plot(GEO(:,3), GEO(:,1), 'r',GEO(:,3), GEO(:,2), 'b');
end
```

Listing A.2: geometryDefinition.dat

```
Node   Airfoil     RNodes      Chord   Thickness    AeroTwst   AeroAxis
   1         0   0.508      0.218       0.218       0.0        0.50
   2         0   0.660      0.218       0.218       0.0        0.50
   3         0   0.883      0.183       0.183       0.0        0.50
   4         1   1.257      0.737       NaN        20.040      0.30
   5         1   1.343      0.728       NaN        18.074      0.30
   6         1   1.510      0.711       NaN        14.292      0.30
   7         1   1.648      0.697       NaN        11.909      0.30
   8         1   1.952      0.666       NaN         7.979      0.30
   9         1   2.257      0.636       NaN         5.308      0.30
  10         1   2.343      0.627       NaN         4.715      0.30
  11         1   2.562      0.605       NaN         3.425      0.30
  12         1   2.867      0.574       NaN         2.083      0.30
  13         1   3.185      0.542       NaN         1.115      0.30
  14         1   3.476      0.512       NaN         0.494      0.30
  15         1   3.781      0.482       NaN        -0.015      0.30
  16         1   4.023      0.457       NaN        -0.381      0.30
  17         1   4.086      0.451       NaN        -0.475      0.30
  18         1   4.391      0.420       NaN        -0.920      0.30
  19         1   4.696      0.389       NaN        -1.352      0.30
  20         1   4.780      0.381       NaN        -1.469      0.30
  21         2   4.938000   0.365050    0.365050   -1.686288   0.300000
  22         2   4.960750   0.360511    0.365000   -1.717574   0.295647
  23         2   4.983500   0.349601    0.360000   -1.748861   0.278263
  24         2   5.006250   0.329376    0.300000   -1.780148   0.238827
  25         2   5.017625   0.312436    0.230000   -1.795791   0.200131
  26         2   5.029000   0.267577    0.030000   -1.811434   0.069038
```

Listing A.3: S809Airfoil.dat

```
x/Chord     y/Chord
```

```
1.00000    0.00000
0.99612    0.00024
0.98446    0.00065
0.96509    0.00054
0.93852   -0.00075
0.90545   -0.00370
0.86677   -0.00859
0.82348   -0.01559
0.77668   -0.02466
0.72752   -0.03558
0.67710   -0.04792
0.62649   -0.06112
0.57663   -0.07442
0.52837   -0.08697
0.48234   -0.09756
0.43832   -0.10484
0.39541   -0.10842
0.35328   -0.10866
0.31188   -0.10589
0.27129   -0.10060
0.23175   -0.09326
0.19362   -0.08447
0.15752   -0.07467
0.12397   -0.06408
0.09325   -0.05301
0.06579   -0.04199
0.04223   -0.03144
0.02321   -0.02162
0.00933   -0.01272
0.00140   -0.00498
0.00000    0.00000
0.00037    0.00275
0.00575    0.01166
0.01626    0.02133
0.03158    0.03136
0.05147    0.04143
0.07568    0.05132
0.10390    0.06082
0.13580    0.06972
0.17103    0.07786
0.20920    0.08505
0.24987    0.09113
0.29259    0.09594
0.33689    0.09933
0.38223    0.10109
0.42809    0.10101
0.47384    0.09843
0.52005    0.09237
0.56801    0.08356
0.61747    0.07379
0.66718    0.06403
0.71606    0.05462
0.76314    0.04578
0.80756    0.03761
0.84854    0.03017
0.88537    0.02335
0.91763    0.01694
0.94523    0.01101
0.96799    0.00600
0.98528    0.00245
0.99623    0.00054
1.00000    0.00000
```

B

Used Measurement Data Part 1: Root Flap Bending Moments and Low Speed Shaft Torque

Part 1 of the used data taken from [26] is presented. It originates from measurements during NREL's UAE Phase VI sequence H test runs with a yaw angle of 0°. Seen are the root flap bending moments as well as the low speed shaft torques plotted for different wind tunnel velocities over time. Additional information is included.

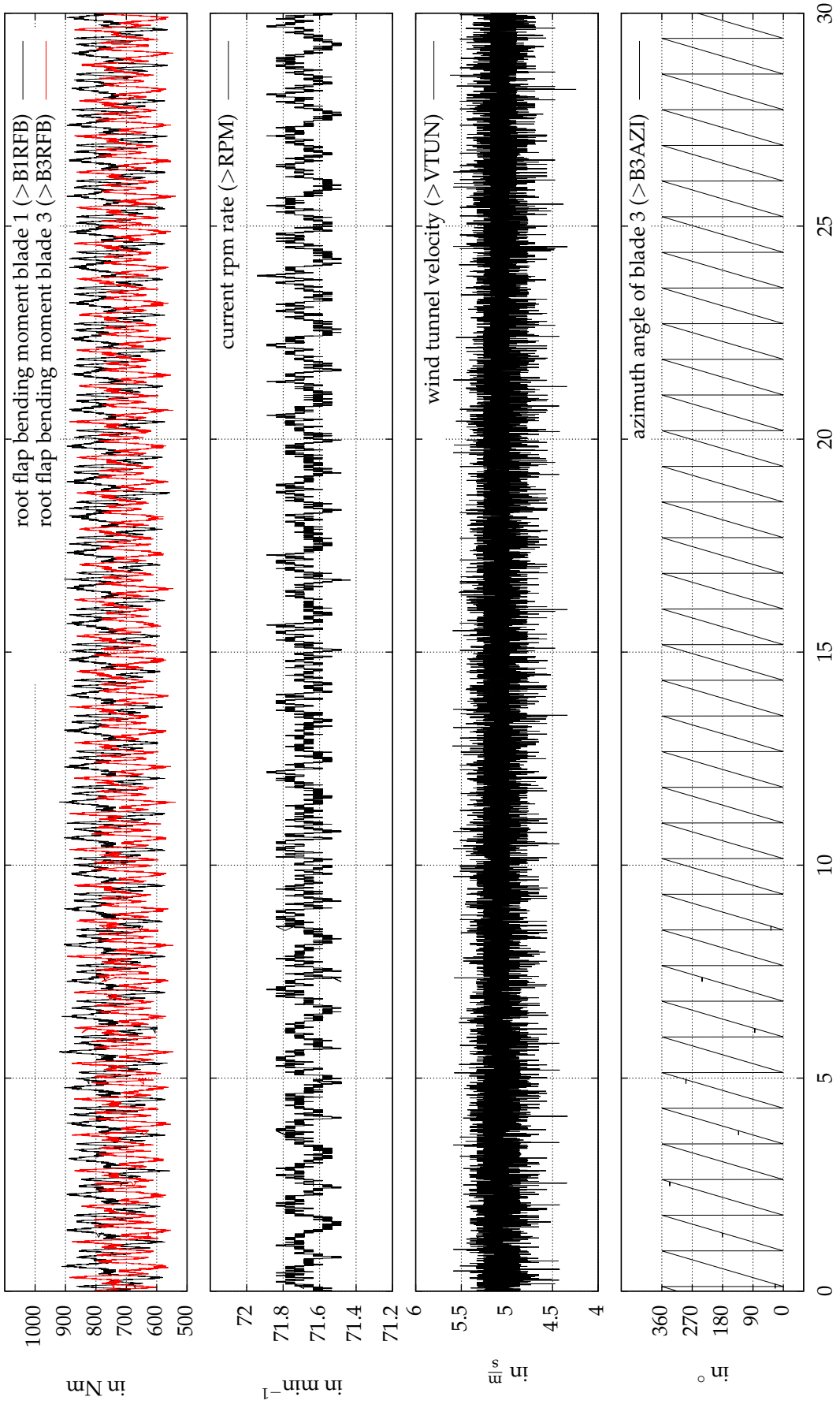


Figure B.1: Time history of the root flap bending moments for blades 1 and 3 with additional information measured during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 5 $\frac{\mathrm{m}}{\mathrm{s}}$, repetition 0 (data taken from [26]). (Average values: $\overline{M}_{A,1,y} = 759.4616$ Nm, $\overline{M}_{A,3,y} = 703.8124$ Nm, $\overline{\Phi}_{p1} = 2.9894°$, $\overline{\Phi}_{p3} = 2.9918°$, $\overline{\vartheta} = 13.4567°$, $\overline{\Phi}_y = 0.0299°$, $\overline{\Phi}_c = -0.0192°$, $\overline{n} = 71.6743\ \mathrm{min}^{-1}$, $\overline{T}_{lss} = 296.9291$ Nm, $\overline{P}_{lss} = 2.2291$ kW, $\overline{u} = 5.0766\ \frac{\mathrm{m}}{\mathrm{s}}$, $\overline{\rho} = 1.2244\ \frac{\mathrm{kg}}{\mathrm{m}^3}$, $\overline{p} = 101107.3879$ Pa)

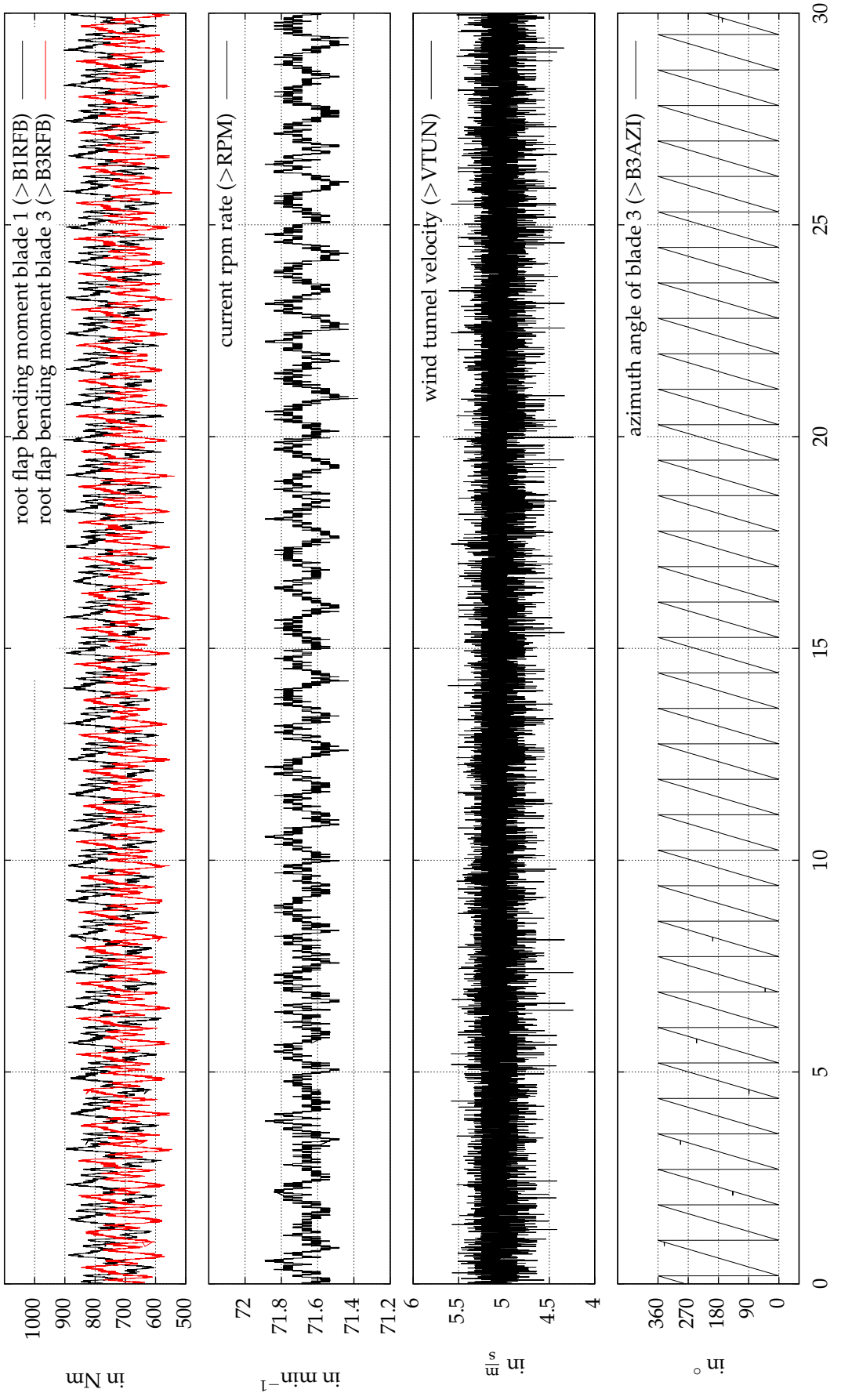


Figure B.2: Time history of the root flap bending moments for blades 1 and 3 with additional information measured during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 5 $\frac{\mathrm{m}}{\mathrm{s}}$, repetition 1 (data taken from [26]). (Average values: $\overline{M}_{A1,y} = 758.8238$ Nm, $\overline{M}_{A3,y} = 702.2171$ Nm, $\overline{\Phi}_{p1} = 2.9876°$, $\overline{\Phi}_{p3} = 2.9862°$, $\overline{\vartheta} = 13.0712°$, $\overline{\Phi}_y = 0.1215°$, $\overline{\Phi}_c = -0.0203°$, $\overline{n} = 71.6687\ \mathrm{min}^{-1}$, $\overline{T}_{iss} = 294.3565$ Nm, $\overline{P}_{iss} = 2.2097$ kW, $\overline{u} = 5.0514\ \frac{\mathrm{m}}{\mathrm{s}}$, $\overline{\rho} = 1.2261\ \frac{\mathrm{kg}}{\mathrm{m}^3}$, $\overline{p} = 101125.1719$ Pa)

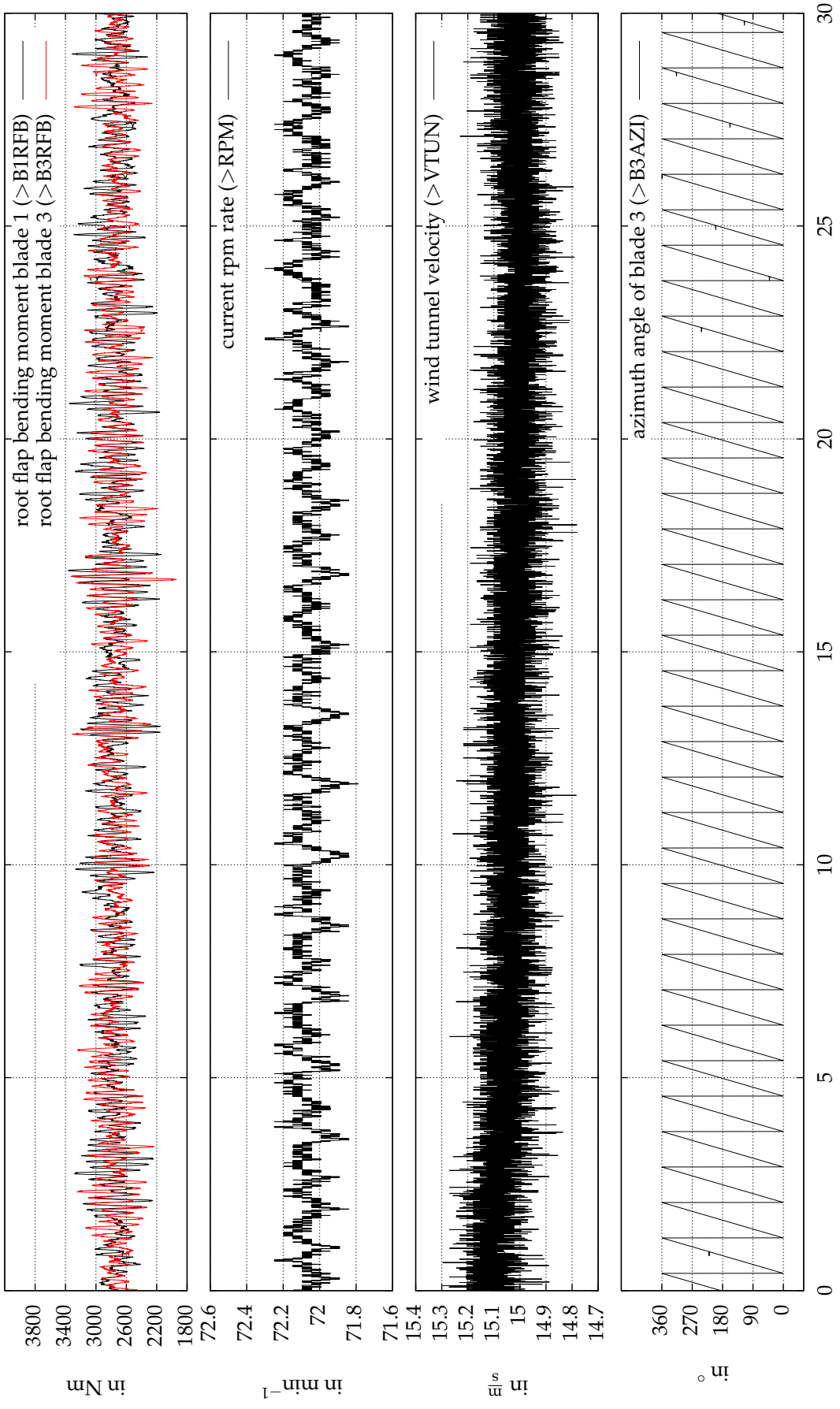


Figure B.3: Time history of the root flap bending moments for blades 1 and 3 with additional information measured during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 15 $\frac{\mathrm{m}}{\mathrm{s}}$, repetition 0 (data taken from [26]). (Average values: $\overline{M}_{A1,y} = 2762.3267$ Nm, $\overline{M}_{A3,y} = 2744.2399$ Nm, $\overline{\Phi}_{p1} = 2.9810°$, $\overline{\Phi}_{p3} = 2.9800°$, $\overline{\vartheta} = 11.9170°$, $\overline{\Phi}_y = -0.1290°$, $\overline{\Phi}_c = -0.0031°$, $\overline{n} = 72.0625\ \mathrm{min}^{-1}$, $\overline{T}_{lss} = 1269.7321$ Nm, $\overline{P}_{lss} = 9.5822$ kW, $\overline{u} = 15.0313\ \frac{\mathrm{m}}{\mathrm{s}}$, $\overline{\rho} = 1.2320\ \frac{\mathrm{kg}}{\mathrm{m}^3}$, $\overline{p} = 101253.6844$ Pa)

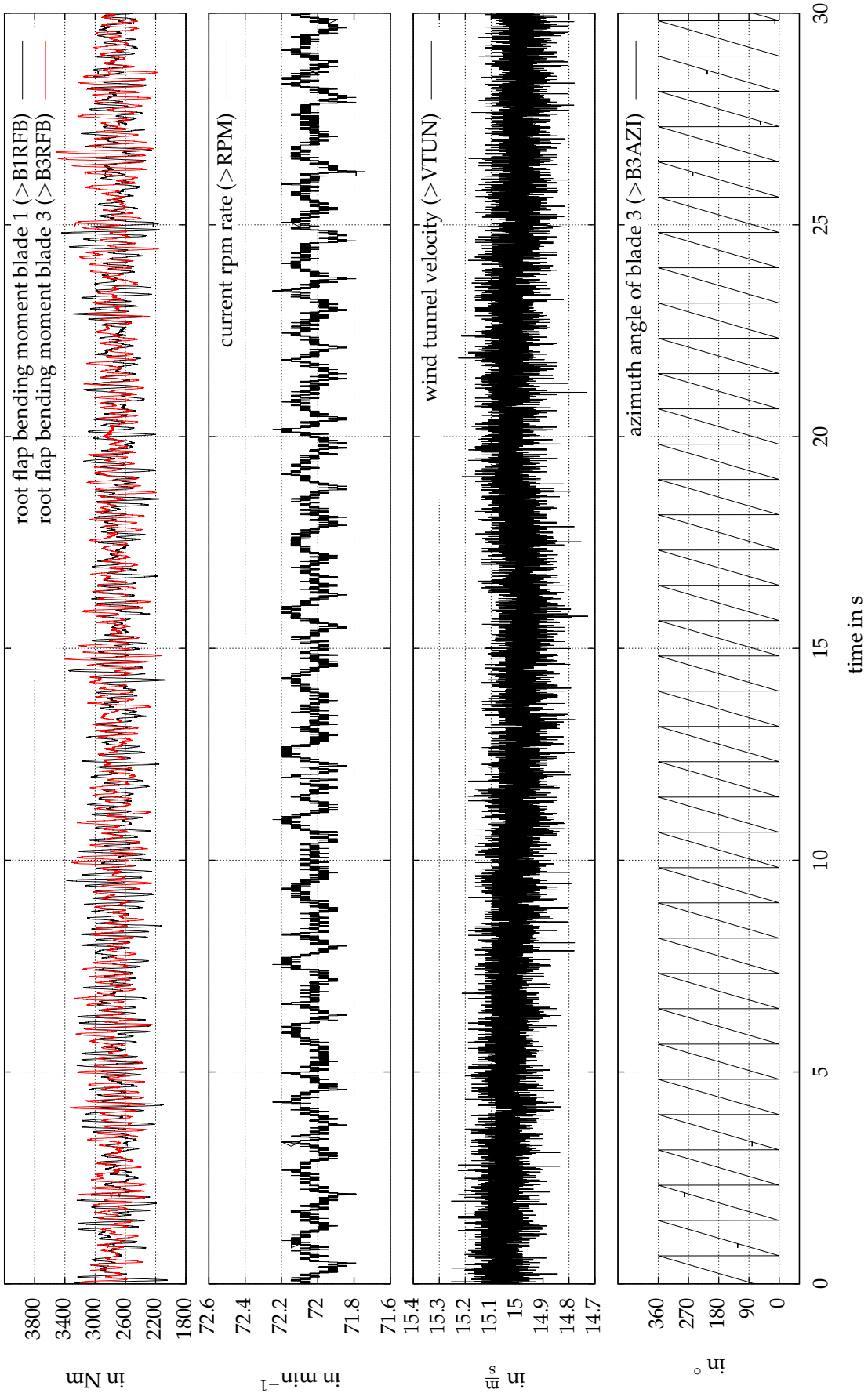


Figure B.4: Time history of the root flap bending moments for blades 1 and 3 with additional information measured during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 15 $\frac{\mathrm{m}}{\mathrm{s}}$, repetition 1 (data taken from [26]). (Average values: $\overline{M}_{A1,y} = 2715.1231$ Nm, $\overline{M}_{A3,y} = 2745.8696$ Nm, $\overline{\Phi}_{p1} = 2.9783°$, $\overline{\Phi}_{p3} = 2.9696°$, $\overline{\vartheta} = 11.8204°$, $\overline{\Phi}_y = 0.0515°$, $\overline{\Phi}_c = -0.0066°$, $\overline{n} = 72.0273\,\mathrm{min}^{-1}$, $\overline{T}_{lss} = 1152.6115$ Nm, $\overline{P}_{lss} = 8.6942$ kW, $\overline{u} = 15.0130\ \frac{\mathrm{m}}{\mathrm{s}}$, $\overline{\rho} = 1.2327\ \frac{\mathrm{kg}}{\mathrm{m}^3}$, $\overline{p} = 101271.0579$ Pa)

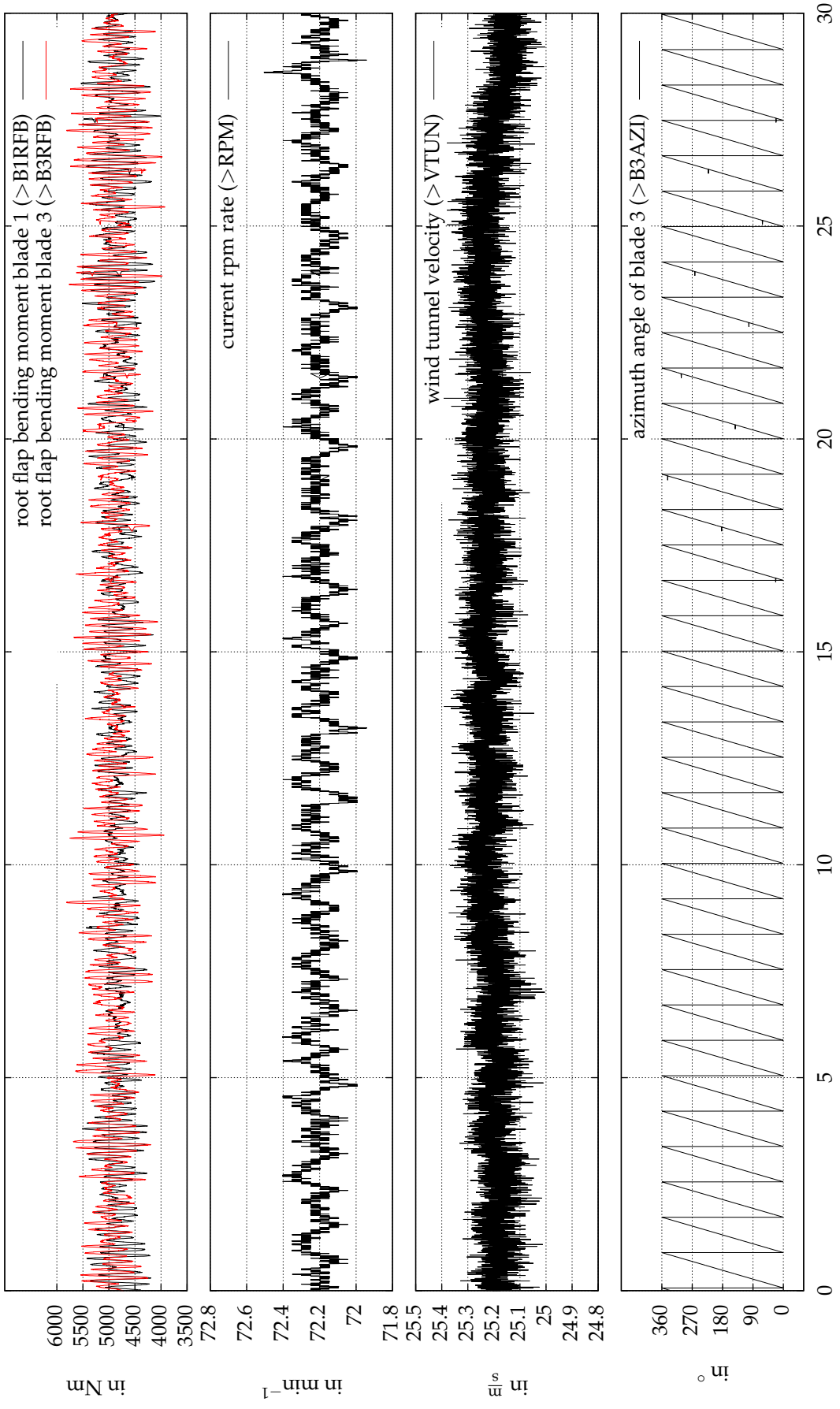


Figure B.5: Time history of the root flap bending moments for blades 1 and 3 with additional information measured during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 25 $\frac{\mathrm{m}}{\mathrm{s}}$, repetition 0 (data taken from [26]). (Average values: $\overline{M}_{A1,y} = 4833.5402$ Nm, $\overline{M}_{A3,y} = 4924.5888$ Nm, $\overline{\Phi}_{p1} = 2.9883°$, $\overline{\Phi}_{p3} = 2.9978°$, $\overline{\vartheta} = 15.7040°$, $\overline{\Phi}_y = -0.0387°$, $\overline{\Phi}_c = 0.0186°$, $\overline{n} = 72.2077\ \mathrm{min}^{-1}$, $\overline{T}_{lss} = 1580.4082$ Nm, $\overline{P}_{lss} = 11.9507$ kW, $\overline{u} = 25.2151\ \frac{\mathrm{m}}{\mathrm{s}}$, $\overline{\rho} = 1.2118\ \frac{\mathrm{kg}}{\mathrm{m}^3}$, $\overline{p} = 101075.0881$ Pa)

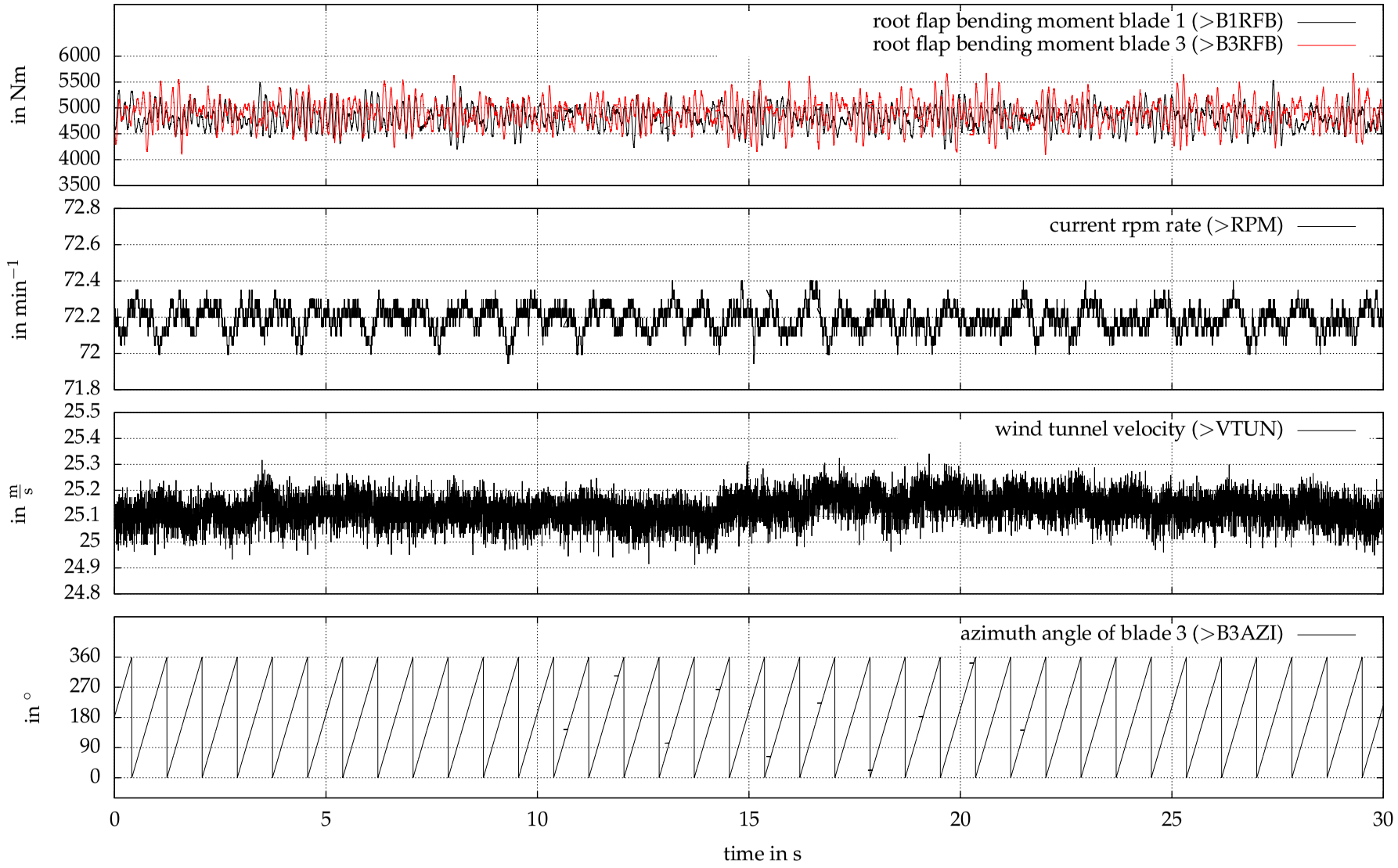


Figure B.6: Time history of the root flap bending moments for blades 1 and 3 with additional information measured during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 25 $\frac{\text{m}}{\text{s}}$, repetition 1 (data taken from [26]). (Average values: $\overline{M}_{A,1,y} = 4807.0106$ Nm, $\overline{M}_{A,3,y} = 4891.1880$ Nm, $\overline{\Phi}_{p1} = 2.9883°$, $\overline{\Phi}_{p3} = 3.0018°$, $\overline{\vartheta} = 15.4851°$, $\overline{\Phi}_y = -0.0449°$, $\overline{\Phi}_c = 0.0186°$, $\overline{n} = 72.1917\,\text{min}^{-1}$, $\overline{T}_{lss} = 1565.9987$ Nm, $\overline{P}_{lss} = 11.8391$ kW, $\overline{u} = 25.1296\ \frac{\text{m}}{\text{s}}$, $\overline{\rho} = 1.2126\ \frac{\text{kg}}{\text{m}^3}$, $\overline{p} = 101063.9259$ Pa)

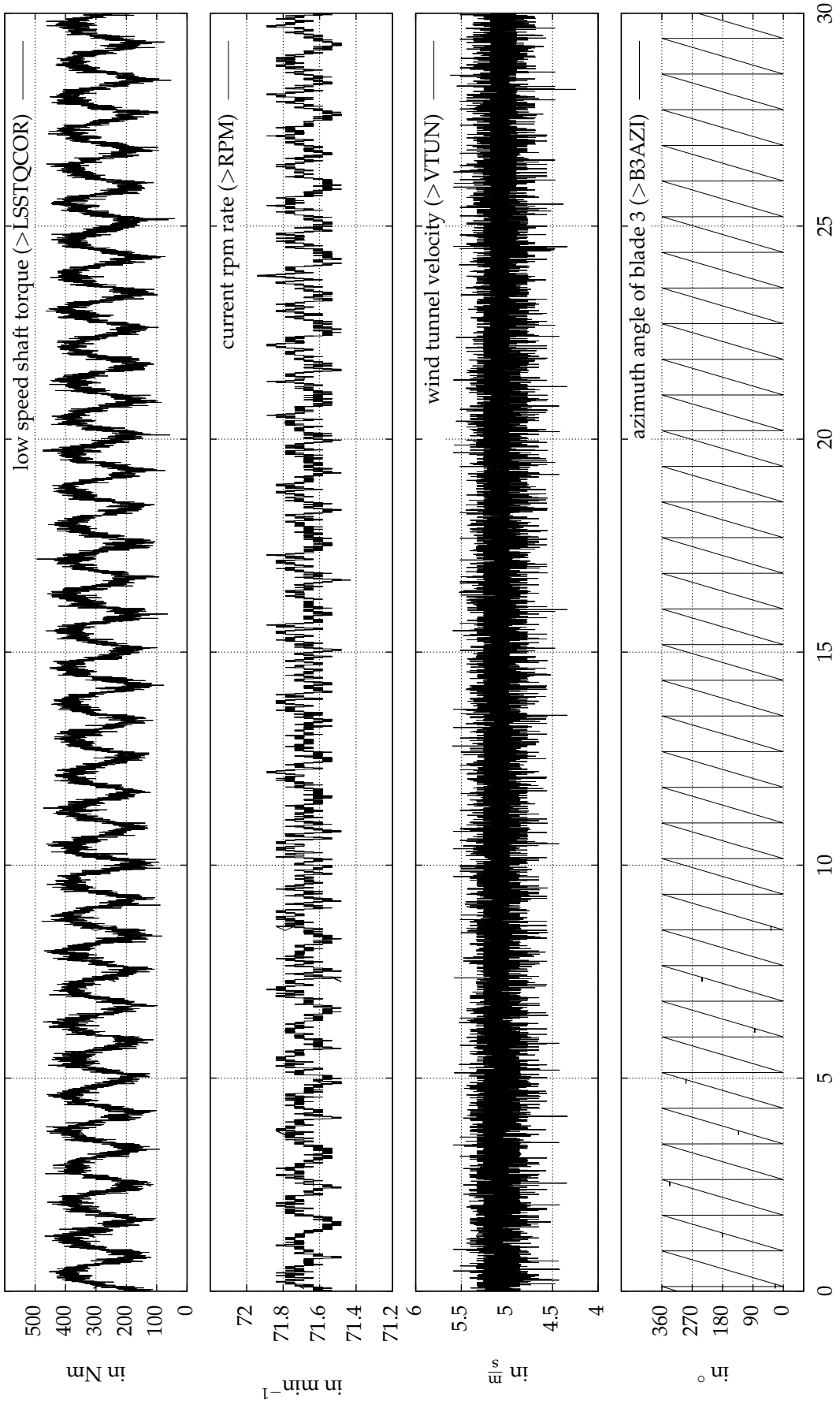


Figure B.7: Time history of the low speed shaft torque with additional information calculated from measurements during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 5 $\frac{\mathrm{m}}{\mathrm{s}}$, repetition 0 (data taken from [26]). (Average values: $\overline{M}_{A,1,y} = 759.4616$ Nm, $\overline{M}_{A,3,y} = 703.8124$ Nm, $\overline{\Phi}_{p1} = 2.9894°$, $\overline{\Phi}_{p3} = 2.9918°$, $\overline{\vartheta} = 13.4567°$, $\overline{\Phi}_y = 0.0299°$, $\overline{\Phi}_c = -0.0192°$, $\overline{n} = 71.6743\ \mathrm{min}^{-1}$, $\overline{T}_{lss} = 296.9291$ Nm, $\overline{P}_{lss} = 2.2291$ kW, $\overline{u} = 5.0766\ \frac{\mathrm{m}}{\mathrm{s}}$, $\overline{\rho} = 1.2244\ \frac{\mathrm{kg}}{\mathrm{m}^3}$, $\overline{p} = 101107.3879$ Pa)

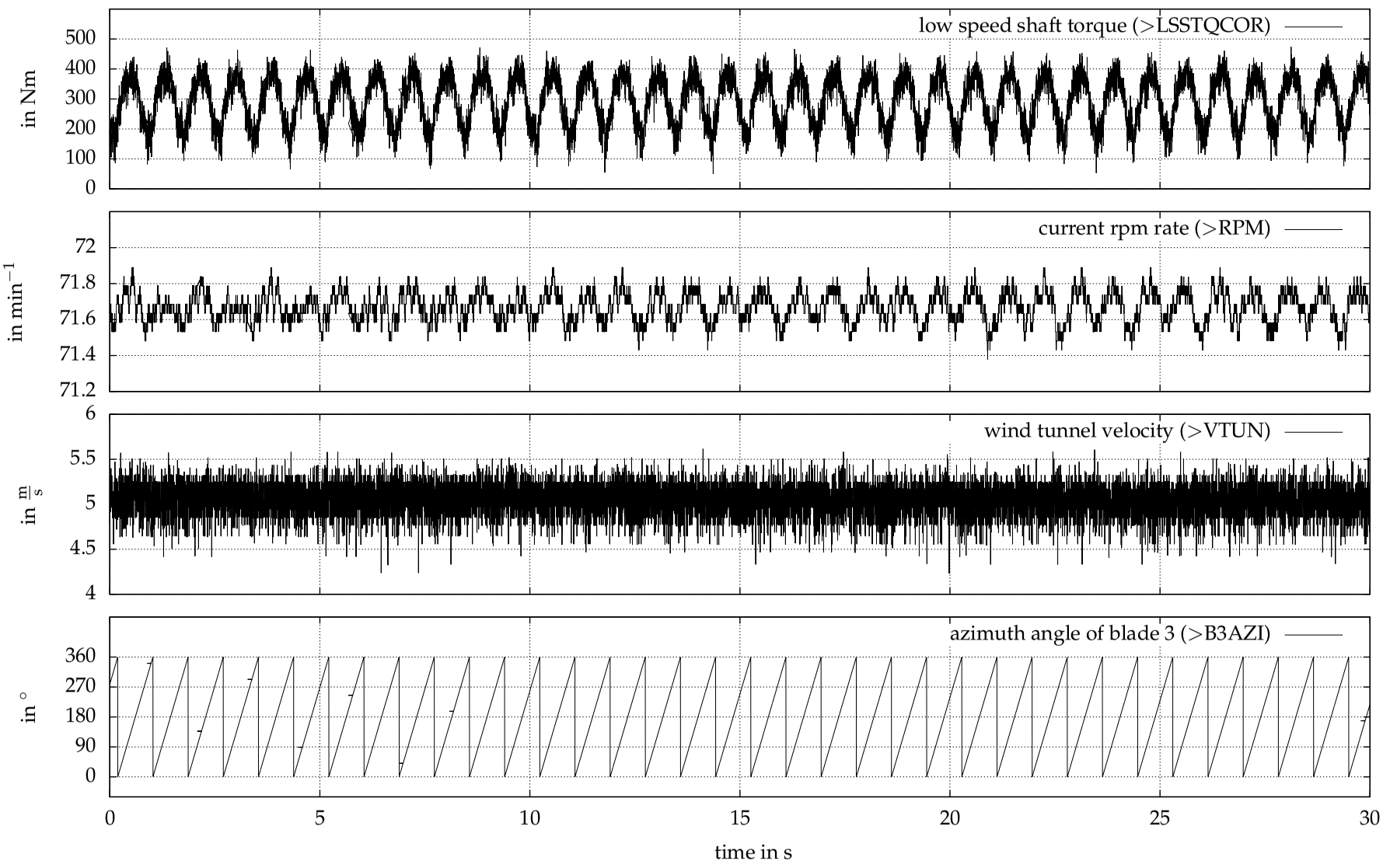


Figure B.8: Time history of the low speed shaft torque with additional information calculated from measurements during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 5 $\frac{m}{s}$, repetition 1 (data taken from [26]). (Average values: $\overline{M}_{A,1,y} = 758.8238$ Nm, $\overline{M}_{A,3,y} = 702.2171$ Nm, $\overline{\Phi}_{p1} = 2.9876°$, $\overline{\Phi}_{p3} = 2.9862°$, $\overline{\vartheta} = 13.0712°$, $\overline{\Phi}_y = 0.1215°$, $\overline{\Phi}_c = -0.0203°$, $\overline{n} = 71.6687\,\text{min}^{-1}$, $\overline{T}_{lss} = 294.3565$ Nm, $\overline{P}_{lss} = 2.2097$ kW, $\overline{u} = 5.0514\ \frac{m}{s}$, $\overline{\rho} = 1.2261\ \frac{kg}{m^3}$, $\overline{p} = 101125.1719$ Pa)

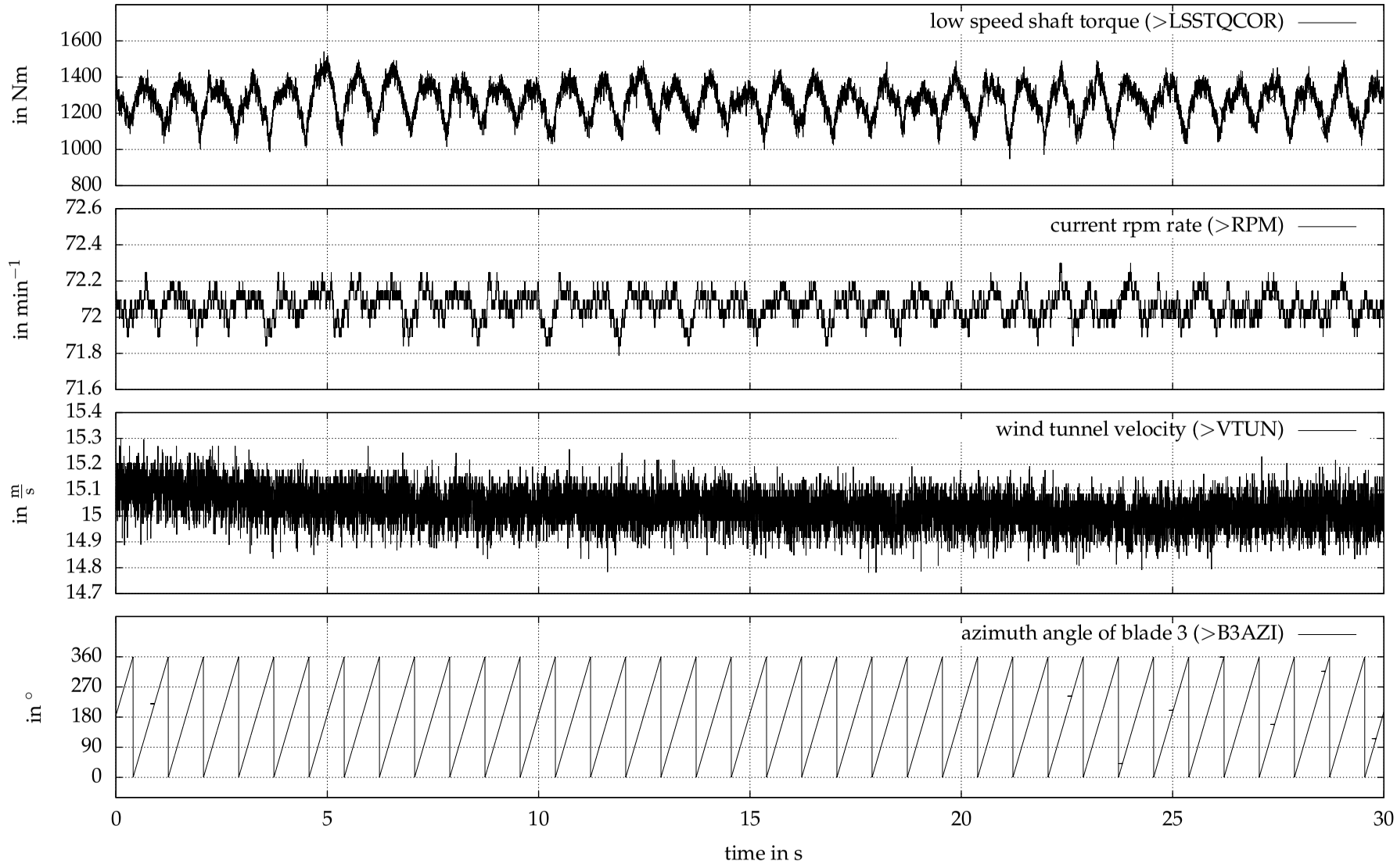


Figure B.9: Time history of the low speed shaft torque with additional information calculated from measurements during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 15 $\frac{m}{s}$, repetition 0 (data taken from [26]). (Average values: $\overline{M}_{A,1,y} = 2762.3267$ Nm, $\overline{M}_{A,3,y} = 2744.2399$ Nm, $\overline{\Phi}_{p1} = 2.9810°$, $\overline{\Phi}_{p3} = 2.9800°$, $\overline{\vartheta} = 11.9170°$, $\overline{\Phi}_y = -0.1290°$, $\overline{\Phi}_c = -0.0031°$, $\overline{n} = 72.0625\,\text{min}^{-1}$, $\overline{T}_{lss} = 1269.7321$ Nm, $\overline{P}_{lss} = 9.5822$ kW, $\overline{u} = 15.0313\ \frac{m}{s}$, $\overline{\rho} = 1.2320\ \frac{kg}{m^3}$, $\overline{p} = 101253.6844$ Pa)

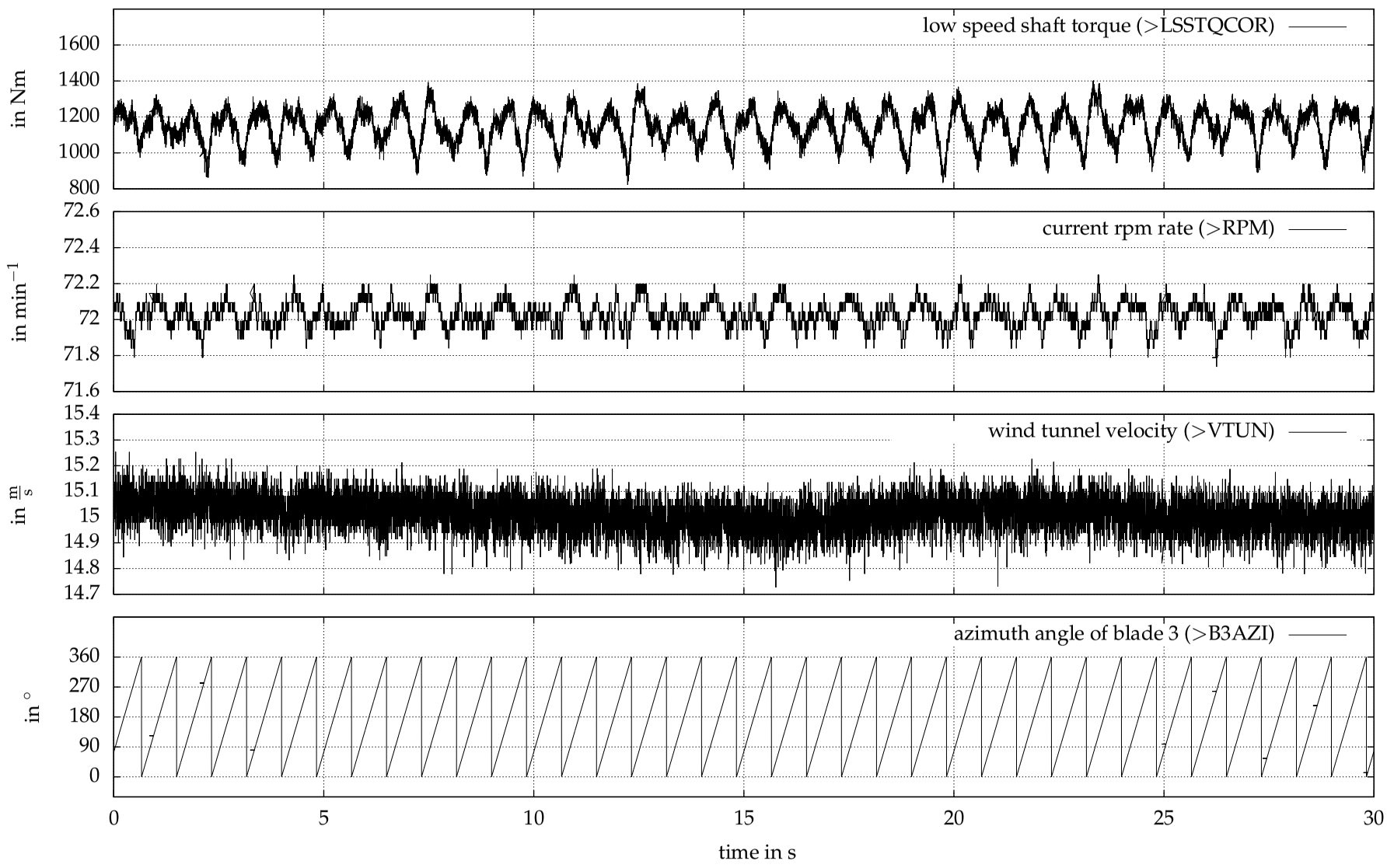


Figure B.10: Time history of the low speed shaft torque with additional information calculated from measurements during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 15 $\frac{m}{s}$, repetition 1 (data taken from [26]). (Average values: $\overline{M}_{A,1,y} = 2715.1231$ Nm, $\overline{M}_{A,3,y} = 2745.8696$ Nm, $\overline{\Phi}_{p1} = 2.9783°$, $\overline{\Phi}_{p3} = 2.9696°$, $\overline{\vartheta} = 11.8204°$, $\overline{\Phi}_y = 0.0515°$, $\overline{\Phi}_c = -0.0066°$, $\overline{n} = 72.0273\,\text{min}^{-1}$, $\overline{T}_{lss} = 1152.6115$ Nm, $\overline{P}_{lss} = 8.6942$ kW, $\overline{u} = 15.0130\ \frac{m}{s}$, $\overline{\rho} = 1.2327\ \frac{kg}{m^3}$, $\overline{p} = 101271.0579$ Pa)

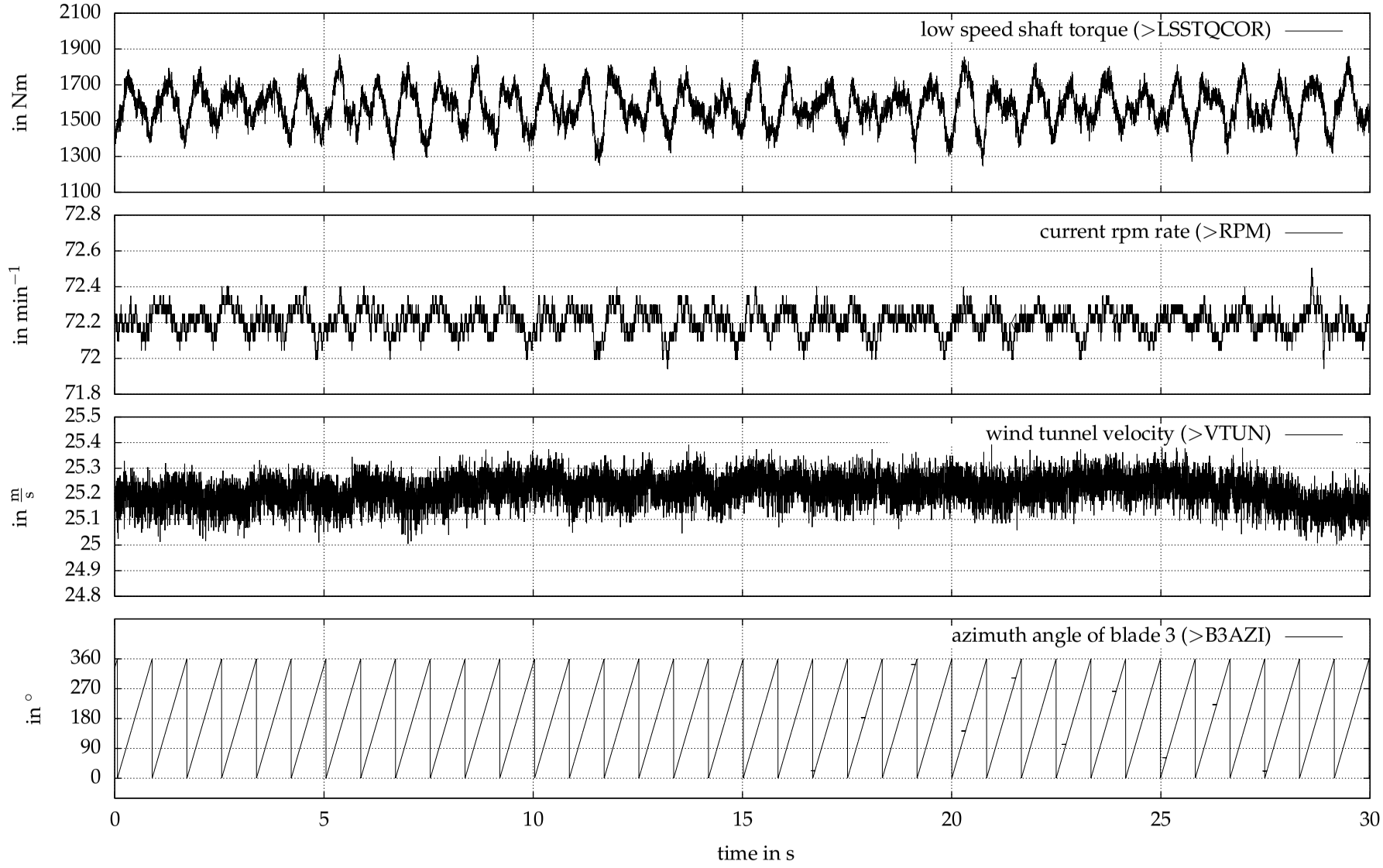


Figure B.11: Time history of the low speed shaft torque with additional information calculated from measurements during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 25 $\frac{m}{s}$, repetition 0 (data taken from [26]). (Average values: $\overline{M}_{A,1,y} = 4833.5402$ Nm, $\overline{M}_{A,3,y} = 4924.5888$ Nm, $\overline{\Phi}_{p1} = 2.9883°$, $\overline{\Phi}_{p3} = 2.9978°$, $\overline{\vartheta} = 15.7040°$, $\overline{\Phi}_y = -0.0387°$, $\overline{\Phi}_c = 0.0186°$, $\overline{n} = 72.2077\,\text{min}^{-1}$, $\overline{T}_{lss} = 1580.4082$ Nm, $\overline{P}_{lss} = 11.9507$ kW, $\overline{u} = 25.2151\ \frac{m}{s}$, $\overline{\rho} = 1.2118\ \frac{kg}{m^3}$, $\overline{p} = 101075.0881$ Pa)

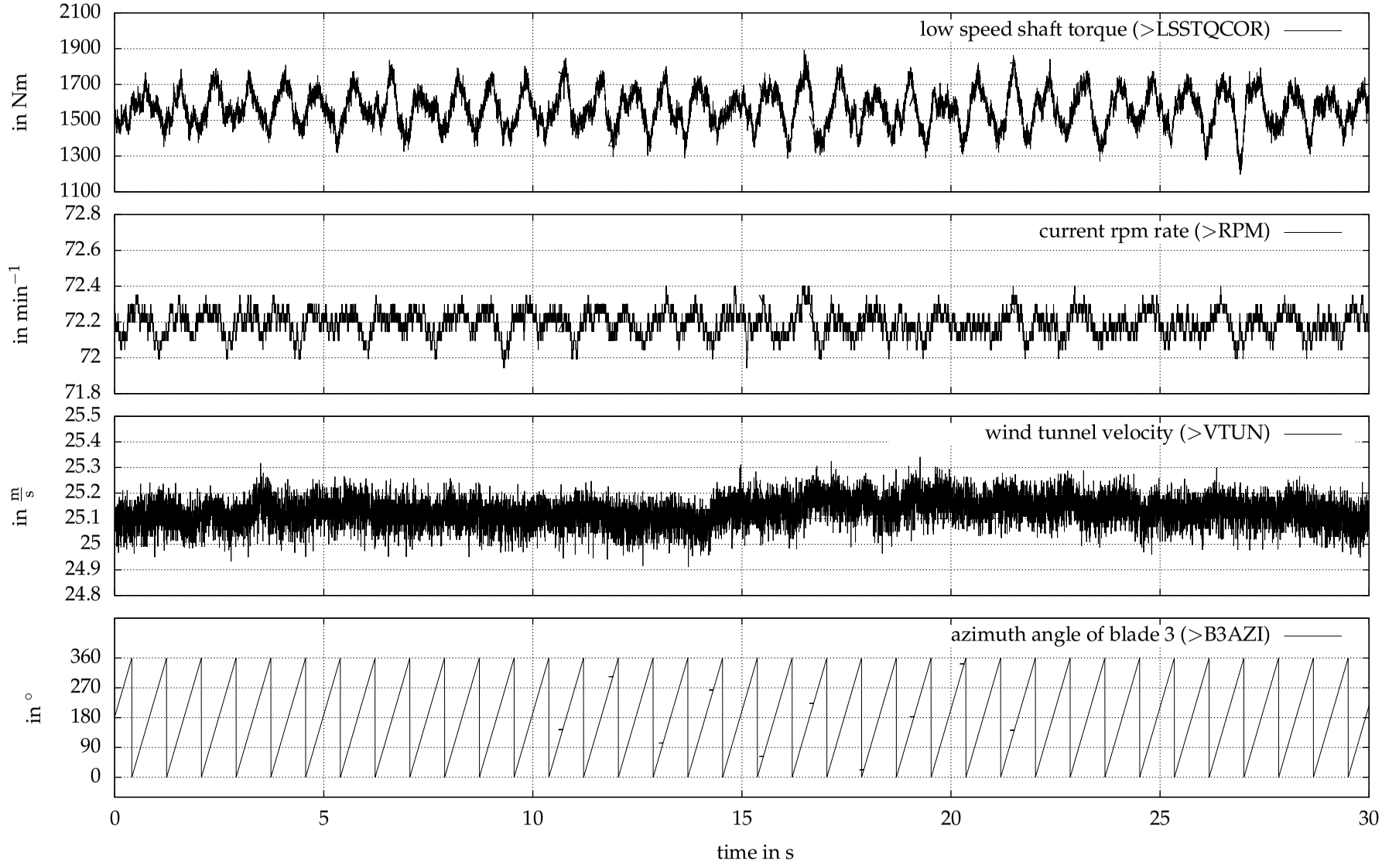


Figure B.12: Time history of the low speed shaft torque with additional information calculated from measurements during NREL's UAE Phase VI sequence H run: yaw angle 0°, wind tunnel velocity 25 $\frac{m}{s}$, repetition 1 (data taken from [26]). (Average values: $\overline{M}_{A,1,y} = 4807.0106$ Nm, $\overline{M}_{A,3,y} = 4891.1880$ Nm, $\overline{\Phi}_{p1} = 2.9883°$, $\overline{\Phi}_{p3} = 3.0018°$, $\overline{\vartheta} = 15.4851°$, $\overline{\Phi}_y = -0.0449°$, $\overline{\Phi}_c = 0.0186°$, $\overline{n} = 72.1917\,\text{min}^{-1}$, $\overline{T}_{lss} = 1565.9987$ Nm, $\overline{P}_{lss} = 11.8391$ kW, $\overline{u} = 25.1296\ \frac{m}{s}$, $\overline{\rho} = 1.2126\ \frac{kg}{m^3}$, $\overline{p} = 101063.9259$ Pa)

C

Point Displacement Definition for the Dummy CSM Client meshClientTurbomachinery

The dummy CSM client meshClientTurbomachinery is sending back independent point displacement data according to the definition in the AbstractDataCreator class. Parabolic (used during step 3/5) and constant displacement (used for the test case own3DPropellerCSMDummy) was implemented.

Listing C.1: AbstractDataCreator.h

```
/*****************************************************************************//**
 * \file AbstractDataCreator.h
 * This file holds the class of the AbstractDataCreator
 *****************************************************************************/

#ifndef ABSTRACTDATACREATOR_H_
#define ABSTRACTDATACREATOR_H_

/********//**
 * \brief This is the superclass of all solvers of the testAdapter
 ***********/

#include <string>
#include <iostream>
#include <math.h>
#include <stdlib.h>

using namespace std;

class AbstractDataCreator {
protected:
    int numNodes;
    int numElems;
    int nodesPerElem;
    double *nodeCoors;
    int *nodeNumbers;
    int *elemTable;

    double (*funcX)(double, double, double);
    double (*funcY)(double, double, double);
    double (*funcZ)(double, double, double);
    double (*funcT)(double, int);
    double (*funcXT)(double, double, double, double);
    double (*funcYT)(double, double, double, double);
```

```
35      double (*funcZT)(double, double, double, double);
36
37
38  [...]
39
40
41      static double meshClientTurbomachineryX(double x, double y, double z) {
42          return 0.0;
43      }
44      static double meshClientTurbomachineryY(double x, double y, double z) {
45          // NREL UAE Phase VI wind turbine (parabolic)
46          double uMax = -1;
47          double xMin = 0.25; // > 0 !!!
48          double xMax = 5.10; // > 0 !!!
49          double deltaY;
50
51          if ((x >= xMin) && (x <= xMax)){
52              deltaY = uMax * (1 / ((xMax - xMin) * (xMax - xMin)) * x * x -
53                (2 * xMin) / ((xMax - xMin) * (xMax - xMin)) * x +
54                (xMin * xMin) / ((xMax - xMin) * (xMax - xMin)));
55          }
56          else if ((x <= -xMin) && (x >= -xMax)){
57              x = -x;
58              deltaY = uMax * (1 / ((xMax - xMin) * (xMax - xMin)) * x * x -
59                (2 * xMin) / ((xMax - xMin) * (xMax - xMin)) * x +
60                (xMin * xMin) / ((xMax - xMin) * (xMax - xMin)));
61          }
62          else {
63              deltaY = 0;
64          }
65  /*
66          // own3DPropellerCSMDummy (constant)
67          double deltaY = -0.3; // = uMax
68  */
69          return deltaY;
70      }
71      static double meshClientTurbomachineryZ(double x, double y, double z) {
72          // NREL UAE Phase VI wind turbine (parabolic)
73          double deltaZ = 0.0;
74  /*
75          // own3DPropellerCSMDummy (constant)
76          double deltaZ = 0.0;
77  */
78          return deltaZ;
79      }
80
81      static double meshClientTurbomachineryT(double space, int timestep) {
82          double timestepfac = 0.02;
83          double deltaXYZofT;
84
85          if (timestep <= 0){
86            deltaXYZofT = 0;
87          }
88          else {
89            deltaXYZofT = space * (timestep - 0) * timestepfac;
90          }
91          return deltaXYZofT;
92      }
93
94
95  public:
96      /********//**
97       ****************************************************************************
98       * \brief Constructor
99       ***********/
100     AbstractDataCreator(int _numNodes, int _numElems, int _nodesPerElem, double *_nodeCoors,
101             int *_nodeNumbers, int *_elemTable, std::string function) :
102             numNodes(_numNodes), numElems(_numElems), nodesPerElem(_nodesPerElem), nodeCoors(
103                     _nodeCoors), nodeNumbers(_nodeNumbers), elemTable(_elemTable) {
104
105
106 [...]
107
108
109         else if (function == "meshClientTurbomachinery") {
110             funcX = meshClientTurbomachineryX;
111             funcY = meshClientTurbomachineryY;
112             funcZ = meshClientTurbomachineryZ;
113             funcT = meshClientTurbomachineryT;
114         }
115
116         else {
117             std::cerr << std::endl
118                     << "AbstractDataCreator::AbstractDataCreator: wrong function type!"
```

```
119                     << std::endl;
120             exit(EXIT_FAILURE);
121         }
122     }
123 
124     /********//**
125      ******************************************************************************
126      * \brief Destructor
127      ***********/
128     virtual ~AbstractDataCreator() {
129         //empty
130     }
131 
132     /********//**
133      ******************************************************************************
134      * \brief Create data on a mesh
135      * \param[out] data is the created (it is specified to be vector instead of scalar)
136      ***********/
137     virtual void create(double *data){};
138     /********//**
139      ******************************************************************************
140      * \brief Create data on a mesh
141      * \param[out] data is the created (it is specified to be vector instead of scalar)
142      * \param[in] time step
143      ***********/
144     virtual void create(double *data, int timestep){};
145     /********//**
146      ******************************************************************************
147      * \brief Create data on a mesh
148      * \param[out] data is the created (it is specified to be vector instead of scalar)
149      * \param[in] current Time
150      ***********/
151     virtual void create(double *data, double tau){};
152 
153 };
154 
155 #endif /* ABSTRACTDATACREATOR_H_ */
```

D

Detailed Settings for STAR-CCM+

Detailed settings for generating the fluid mesh with version 7.06 of CD-adapco's STAR-CCM+ are presented in form of screenshots. The resulting mesh is exported as file `XXX.ccm`, imported in OpenFOAM via `ccm26ToFoam XXX.ccm` and validated with the tool `checkMesh`, whose output concerning the presented mesh is also appended.

Figure D.1: Screenshots part 1 providing detailed STAR-CCM+ settings. To be read top-down in portrait mode

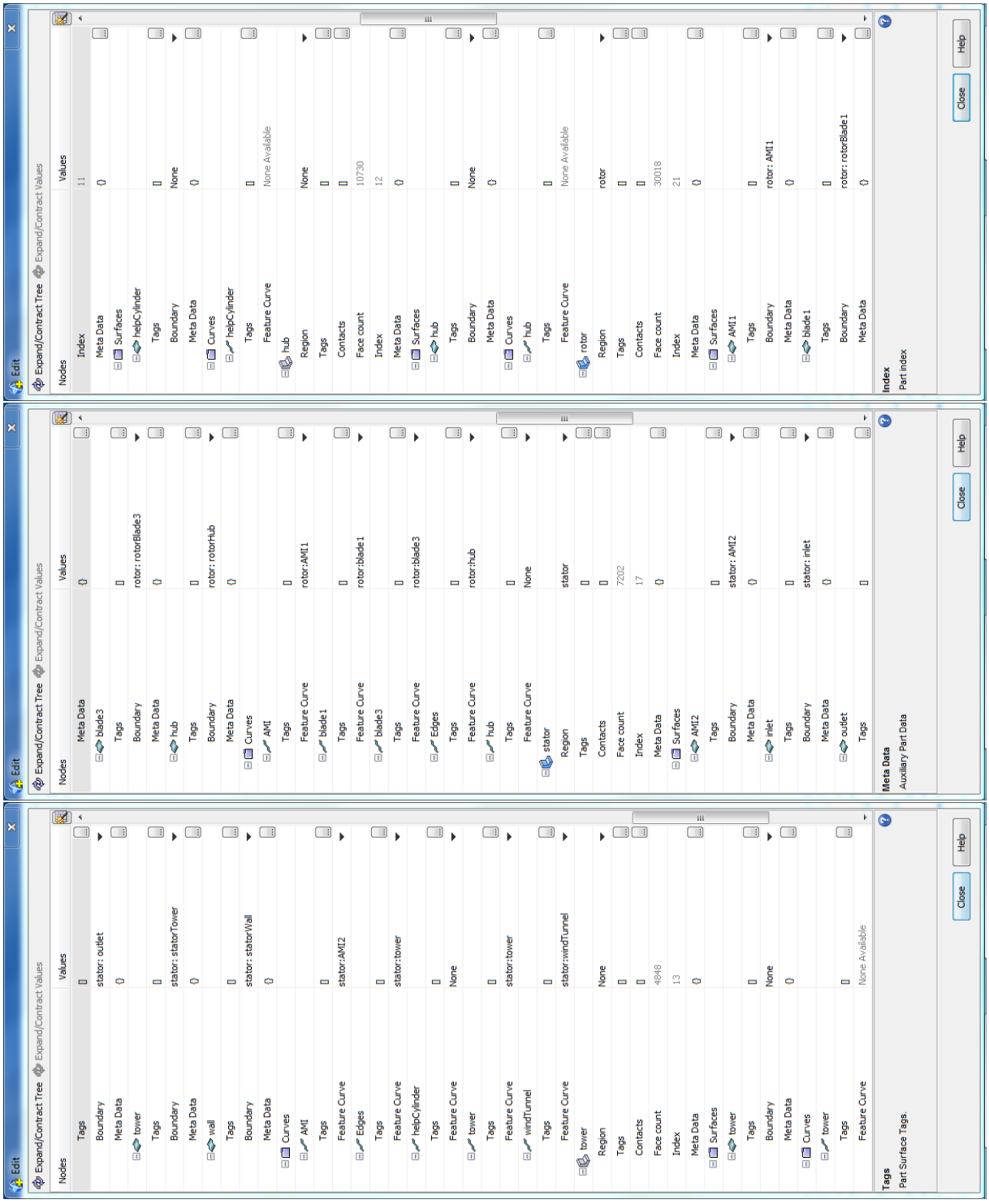


Figure D.2: Screenshots part 2 providing detailed STAR-CCM+ settings. To be read top-down in portrait mode

Figure D.3: Screenshots part 3 providing detailed STAR-CCM+ settings. To be read top-down in portrait mode

Figure D.4: Screenshots part 4 providing detailed STAR-CCM+ settings. To be read top-down in portrait mode

Figure D.5: Screenshots part 5 providing detailed STAR-CCM+ settings. To be read top-down in portrait mode

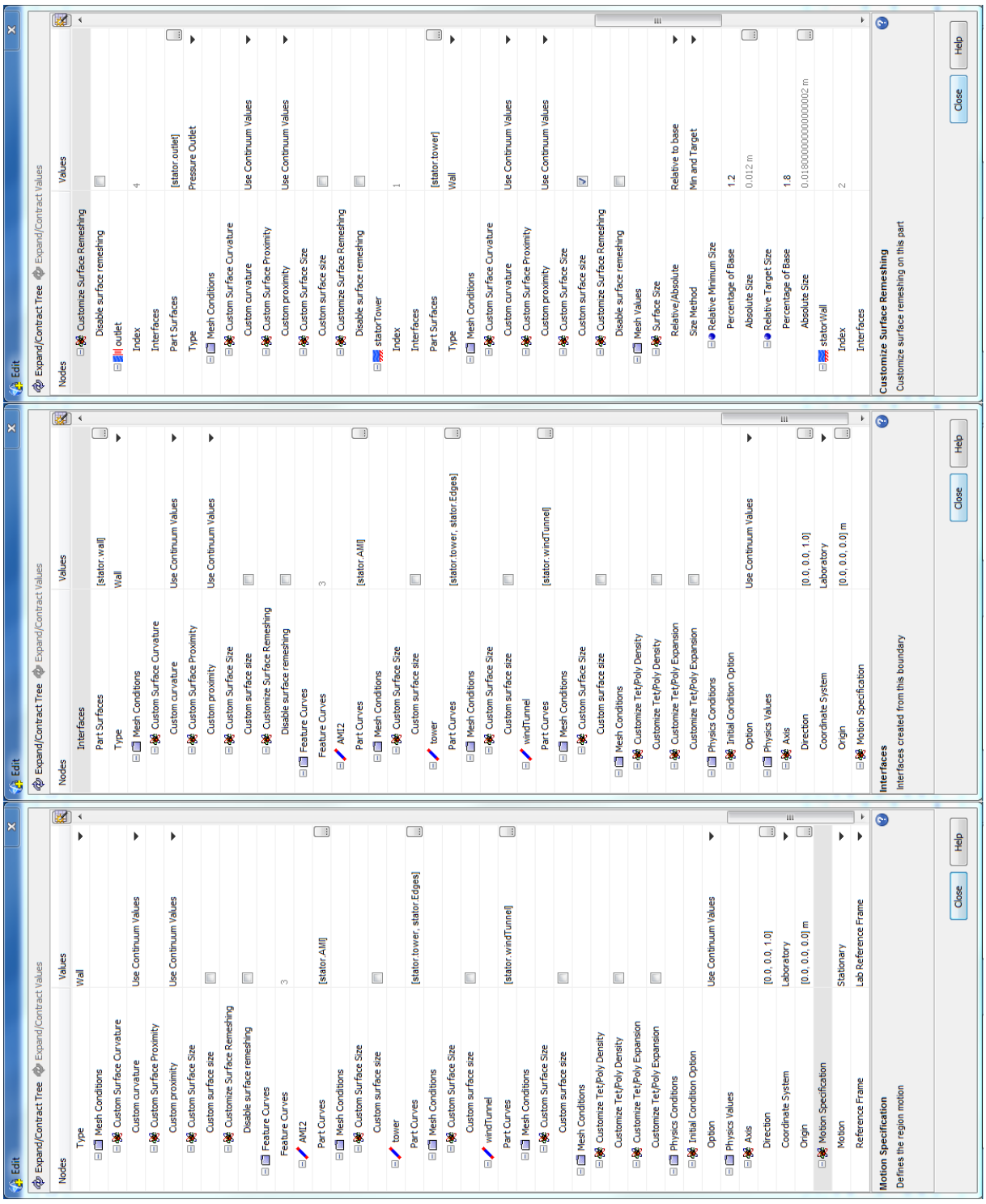


Figure D.6: Screenshots part 6 providing detailed STAR-CCM+ settings. To be read top-down in portrait mode

Listing D.1: checkMesh.log

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
Build  : 2.1.x-73ed96d3e552
Exec   : checkMesh
Date   : Mar 13 2013
Time   : 16:20:59
Host   : "head"
PID    : 29094
Case   : /home/christopher/starccm+/pimpleDyMFoam/OF
nProcs : 1
sigFpe : Floating point exception trapping - not supported on this platform
fileModificationChecking : Monitoring run-time modified files using timeStampMaster
allowSystemOperations : Disallowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Create polyMesh for time = 0

Time = 0

Mesh stats
    points:           11427259
    faces:            125121275
    internal faces:   121203137
    cells:            61581103
    boundary patches: 9
    point zones:      1
    face zones:       2
    cell zones:       4

Overall number of cells of each type:
    hexahedra:     0
    prisms:        0
    wedges:        0
    pyramids:      0
    tet wedges:    0
    tetrahedra:    61581103
    polyhedra:     0

Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
   *Number of regions: 2
    The mesh has multiple regions which are not connected by any face.
  <<Writing region information to "0/cellToRegion"

Checking patch topology for multiply connected surfaces ...
    Patch               Faces    Points   Surface topology
    rotorHub            23615    12247    ok (non-closed singly connected)
    AMI1                251282   125706   ok (non-closed singly connected)
    rotorBlade3         1455829  728102   ok (non-closed singly connected)
    rotorBlade1         1457100  728737   ok (non-closed singly connected)
    statorTower         165058   82693    ok (non-closed singly connected)
    statorWall          281828   141529   ok (non-closed singly connected)
    inlet               32628    16571    ok (non-closed singly connected)
    outlet              32628    16571    ok (non-closed singly connected)
    AMI2                218170   109146   ok (non-closed singly connected)

Checking geometry...
    Overall domain bounding box (-18.02892 -38.608 -12.164) (18.02892 19.304 11.9052)
    Mesh (non-empty, non-wedge) directions (1 1 1)
    Mesh (non-empty) directions (1 1 1)
    Boundary openness (-3.694238e-14 -3.537392e-15 -5.59639e-15) OK.
    Max cell openness = 3.279989e-16 OK.
    Max aspect ratio = 13.15011 OK.
    Minimum face area = 1.806004e-07. Maximum face area = 0.1441349.  Face area magnitudes OK.
    Min volume = 8.444738e-11. Max volume = 0.01196118.  Total volume = 50256.59.  Cell volumes OK.
    Mesh non-orthogonality Max: 79.92787 average: 15.72573
   *Number of severely non-orthogonal faces: 5.
    Non-orthogonality check OK.
  <<Writing 5 non-orthogonal faces to set nonOrthoFaces
    Face pyramids OK.
    Max skewness = 0.7190635 OK.
```

```
83     Coupled point location match (average 0) OK.
84
85 Mesh OK.
86
87 End
```

E

Full Case Structure with All OpenFOAM Dictionaries for Step 1/5

Figure E.1 on the next page gives an overview of the fully needed case structure to do a step 1/5 simulation. Included are also output files of most interest. The subordination in `$EMPIRE_CASE` holds only the general case structure of all five steps. All input files, as they were used with the presented `MRFSimpleFoam` results, are plotted here.

Listing E.1: $EMPIRE_CASE/OF/0/k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.0009375;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }

    outlet
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }

```

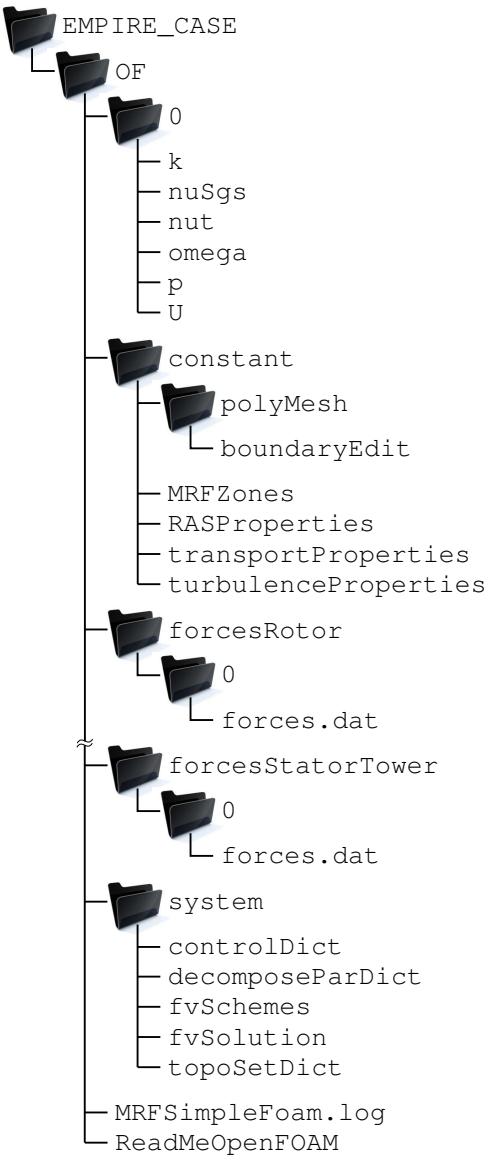


Figure E.1: Full case structure for step 1/5 using `MRFSimpleFoam` with all OpenFOAM dictionaries and output files of most interest

```
    "stator.*"
    {
        type            slip;
    }

    "rotor.*"
    {
        type            kqRWallFunction;
        value           $internalField;
    }

    "AMI.*"
    {
        type            cyclicAMI;
        value           $internalField;
    }
}

// ************************************************************************* //
```

Listing E.2: $EMPIRE_CASE/OF/0/nuSgs

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nutSgs;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    "stator.*"
    {
        type            calculated;
        value           uniform 0;
    }

    "rotor.*"
    {
        type            nutUSpaldingWallFunction;
        value           uniform 0;
    }

    "AMI.*"
    {
        type            cyclicAMI;
        value           uniform 0;
    }
}

// ************************************************************************* //
```

Listing E.3: $EMPIRE_CASE/OF/0/nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    "stator.*"
    {
        type            calculated;
        value           uniform 0;
    }

    "rotor.*"
    {
        type            nutkWallFunction;
        value           uniform 0;
    }

    "AMI.*"
    {
        type            cyclicAMI;
        value           uniform 0;
    }
}

// ************************************************************************* //
```

Listing E.4: $EMPIRE_CASE/OF/0/omega

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 -1 0 0 0 0];

internalField   uniform 0.00248945;

boundaryField
{
```

```
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }

    outlet
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }

    "stator.*"
    {
        type            slip;
    }

    "rotor.*"
    {
        type            omegaWallFunction;
        value           $internalField;

    }

    "AMI.*"
    {
        type            cyclicAMI;
        value           $internalField;
    }
}

// ************************************************************************* //
```

Listing E.5: $EMPIRE_CASE/OF/0/p

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    "stator.*"
    {
        type            slip;
    }

    "rotor.*"
    {
        type            zeroGradient;
    }

    "AMI.*"
```

```
46      {
47          type            cyclicAMI;
48          value           $internalField;
49      }
50  }
51
52  // ************************************************************************* //
```

Listing E.6: $EMPIRE_CASE/OF/0/U

```
1   /*--------------------------------*- C++ -*----------------------------------*\
2   | =========                 |                                                 |
3   | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4   |  \\    /   O peration     | Version:  2.1.x                                 |
5   |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6   |    \\/     M anipulation  |                                                 |
7   \*---------------------------------------------------------------------------*/
8   FoamFile
9   {
10      version     2.0;
11      format      ascii;
12      class       volVectorField;
13      location    "0";
14      object      U;
15  }
16  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18  dimensions      [0 1 -1 0 0 0 0];
19
20  internalField   uniform (0 -5 0);
21
22  boundaryField
23  {
24      inlet
25      {
26          type            fixedValue;
27          value           uniform (0 -5 0);
28      }
29
30      outlet
31      {
32          type            zeroGradient;
33      }
34
35      "stator.*"
36      {
37          type            slip;
38      }
39
40      "rotor.*"
41      {
42          type            fixedValue;
43          value           uniform (0 0 0);
44      }
45
46      "AMI.*"
47      {
48          type            cyclicAMI;
49          value           $internalField;
50      }
51  }
52
53  // ************************************************************************* //
```

Listing E.7: $EMPIRE_CASE/OF/constant/MRFZones

```
1   /*--------------------------------*- C++ -*----------------------------------*\
2   | =========                 |                                                 |
3   | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4   |  \\    /   O peration     | Version:  2.1.x                                 |
5   |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6   |    \\/     M anipulation  |                                                 |
7   \*---------------------------------------------------------------------------*/
8   FoamFile
9   {
10      version     2.0;
11      format      ascii;
12      class       dictionary;
13      location    "constant";
```

```
    object      MRFZones;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

1
(
    cellZone_1
    {
        // Fixed patches (by default they 'move' with the MRF zone)
        nonRotatingPatches (AMI1 AMI2);

        origin    origin [0 1 0 0 0 0 0]  (0 0 0);
        axis      axis   [0 0 0 0 0 0 0]  (0 1 0);
        omega     omega  [0 0 -1 0 0 0 0] 7.539822;
    }
)

// ************************************************************************* //
```

Listing E.8: $EMPIRE_CASE/OF/constant/RASProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        kOmegaSST;

turbulence      on;

printCoeffs     on;

// ************************************************************************* //
```

Listing E.9: $EMPIRE_CASE/OF/constant/transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1.46e-05;

// ************************************************************************* //
```

Listing E.10: $EMPIRE_CASE/OF/constant/turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

```
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RASModel;

// ************************************************************************* //
```

Listing E.11: $EMPIRE_CASE/OF/constant/polyMesh/boundary

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

9
(
    rotorHub
    {
        type            wall;
        nFaces          23615;
        startFace       99018528;
    }
    AMI1
    {
        type            cyclicAMI;
        nFaces          251282;
        startFace       99042143;
        matchTolerance  0.0001;
        neighbourPatch  AMI2;
        transform       noOrdering;
    }
    rotorBlade3
    {
        type            wall;
        nFaces          1455829;
        startFace       99293425;
    }
    rotorBlade1
    {
        type            wall;
        nFaces          1457100;
        startFace       100749254;
    }
    AMI2
    {
        type            cyclicAMI;
        nFaces          96040;
        startFace       102206354;
        matchTolerance  0.0001;
        neighbourPatch  AMI1;
        transform       noOrdering;
    }
    inlet
    {
        type            patch;
        nFaces          17868;
        startFace       102302394;
    }
```

```
    statorWall
    {
        type            wall;
        nFaces          130000;
        startFace       102320262;
    }
    outlet
    {
        type            patch;
        nFaces          17868;
        startFace       102450262;
    }
    statorTower
    {
        type            wall;
        nFaces          1788;
        startFace       102468130;
    }
)

// ************************************************************************* //
```

Listing E.12: $EMPIRE_CASE/OF/constant/polyMesh/boundaryEdit

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundaryEdit;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// * * * * * * * * * * For editing real file "boundary"  * * * * * * * * * * //
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    AMI1
    {
        type            cyclicAMI;
        nFaces          (number);
        startFace       (number);
        matchTolerance  0.0001;
        neighbourPatch  AMI2;
        transform       noOrdering;
    }

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// * * * * * * * * * * For editing real file "boundary"  * * * * * * * * * * //
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    AMI2
    {
        type            cyclicAMI;
        nFaces          (number);
        startFace       (number);
        matchTolerance  0.0001;
        neighbourPatch  AMI1;
        transform       noOrdering;
    }

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// * * * * * * * * * * For editing real file "boundary"  * * * * * * * * * * //
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// ************************************************************************* //
```

Listing E.13: $EMPIRE_CASE/OF/system/controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
```

```
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     MRFSimpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         100000;

deltaT          1;

writeControl    timeStep;

writeInterval   1000;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

functions
{
    forcesRotorBlade1
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches (rotorBlade1);
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 0 0);

            outputControl     timeStep;
            outputInterval    1;
    }

    forcesRotorBlade3
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches (rotorBlade3);
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 0 0);

            outputControl     timeStep;
            outputInterval    1;
    }

    forcesRotorHub
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches (rotorHub);
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 0 0);

```

```
87              outputControl     timeStep;
88              outputInterval    1;
89      }
90
91      forcesRotor
92      {
93          type    forces;
94          functionObjectLibs ("libforces.so");
95          patches ("rotor.*");
96          rhoName rhoInf;
97          rhoInf  1.23;
98          CofR    (0 0 0);
99
100             outputControl     timeStep;
101             outputInterval    1;
102     }
103
104     forcesStatorTower
105     {
106         type    forces;
107         functionObjectLibs ("libforces.so");
108         patches (statorTower);
109         rhoName rhoInf;
110         rhoInf  1.23;
111         CofR    (0 -1.401 -12.164); // with reference to root
112
113             outputControl     timeStep;
114             outputInterval    1;
115     }
116 }
117
118 // ************************************************************************* //
```

Listing E.14: $EMPIRE_CASE/OF/system/decomposeParDict

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.1.x                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "system";
14     object      decomposeParDict;
15 }
16
17 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
18
19 singleProcessorFaceSets ((AMI 1));
20 numberOfSubdomains    144;
21 method                scotch;
22 distributed           no;
23 //strategy              speed;
24 roots                 ( );
25
26 // ************************************************************************* //
```

Listing E.15: $EMPIRE_CASE/OF/system/fvSchemes

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.1.x                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "system";
14     object      fvSchemes;
15 }
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default steadyState;
}

gradSchemes
{
    default     Gauss linear;
}

divSchemes
{
    default     none;
    div(phi,U)  Gauss linearUpwindV grad(U);
    div(phi,k)  Gauss upwind;
    div(phi,omega) Gauss upwind;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default       Gauss linear limited 0.333;
    //default     Gauss linear corrected;
}

interpolationSchemes
{
    default     linear;
    interpolate(HbyA) linear;
}

snGradSchemes
{
    default       limited 0.333;
}

fluxRequired
{
    default     no;
    p           ;
}

// ************************************************************************* //
```

Listing E.16: $EMPIRE_CASE/OF/system/fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{

    p
    {
        solver          GAMG;
        smoother        DIC;
        preconditioner  FDIC;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration  off;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel  32;
        mergeLevels      1;
        tolerance        1e-6;
        relTol           0;
```

```
    }

    "(U|k|omega)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-6;
        relTol          0.0;
    }

}

potentialFlow
{
        nNonOrthogonalCorrectors  8;
}


SIMPLE
{
    nNonOrthogonalCorrectors  2;
    pRefCell            0;
    pRefValue           0;
    residualControl
    {
        p               1e-5;
        U               1e-5;
    }
}

relaxationFactors
{
    fields
    {
       p                0.52;
    }
    equations
    {
        "(U|k|omega).*"  0.63;
    }
}

// ************************************************************************* //
```

Listing E.17: $EMPIRE_CASE/OF/system/topoSetDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    AMI;
        type    faceSet;
        action  new;
        source  patchToFace;
        sourceInfo
        {
            name     "AMI.*";
        }
    }
    {
        name    frozen;
        type    pointSet;
        action  new;
        source  cellToPoint;
        sourceInfo
```

```
36          {
37              set      cellZone_2;
38              option   all;
39          }
40      }
41      {
42          name    frozen;
43          type    pointSet;
44          action  add;
45          source  faceToPoint;
46          sourceInfo
47          {
48              set      AMI;
49              option   all;
50          }
51      }
52  );
53
54  // ************************************************************************* //
```

Listing E.18: $EMPIRE_CASE/OF/readMeOpenFOAM

```
1  ===============================================================================
2  =                                   OF                                        =
3  ===============================================================================
4
5  1. ccm26ToFoam NRELUAEPhaseVIWindturbine.ccm
6
7     rm -f 0/cellId.gz 0/cellType.gz
8
9     # edit file "constant/polyMesh/boundary"
10      with file "constant/polyMesh/boundaryEdit"
11
12 ===============================================================================
13
14 2. topoSet
15
16    setsToZones -noFlipMap
17
18 ===============================================================================
```

F

Used Measurement Data Part 2: Normalized Pressure Coeficients

Simulation results for normalized pressure coefficients have been compared to measured ones. The simulation data has been achieved with `MRFSimpleFoam` (step 1/5) after 5840 iterations. The measurement data originates from NREL's UAE Phase VI sequence H test run with a yaw angle of 0° and a wind tunnel velocity of 5 $\frac{\mathrm{m}}{\mathrm{s}}$ (repetition 0). The results for all available radial ranges are presented.

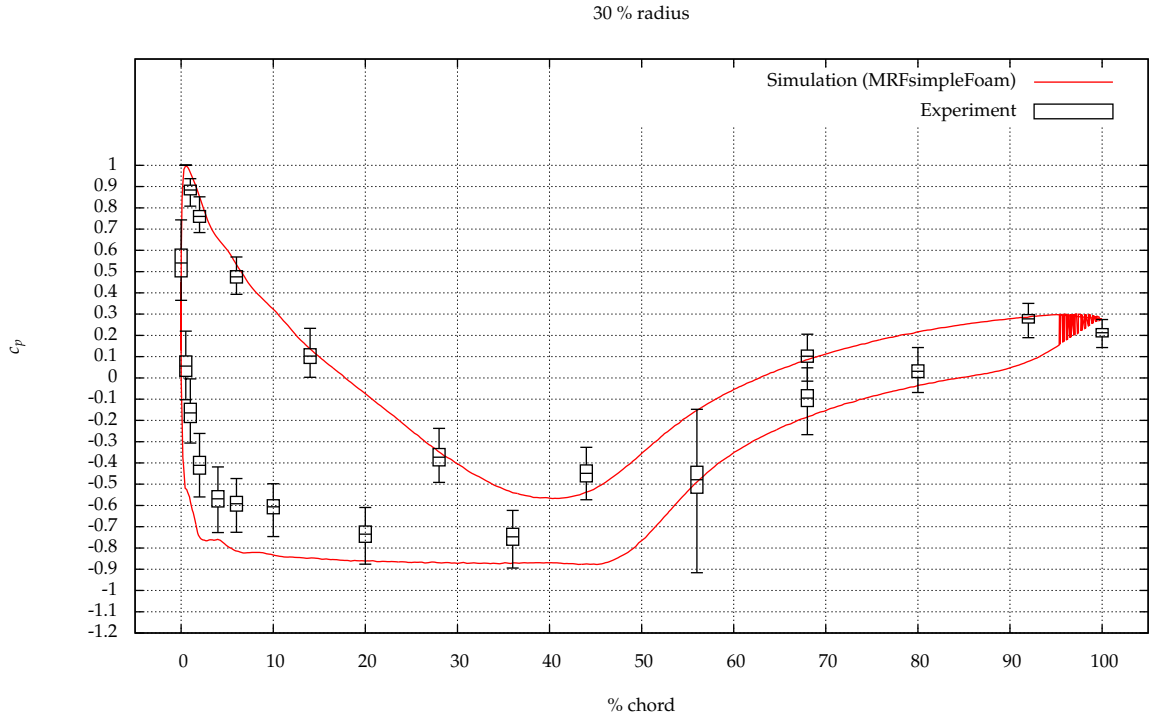


Figure F.1: Normalized pressure coefficients at 30 % radius = 1.5087 m: Simulation results (step 1/5, `MRFSimpleFoam`, 5840 iterations) vs. NREL UAE Phase VI (sequence H, yaw angle 0°, wind tunnel velocity 5 $\frac{m}{s}$, repetition 0) measurements. Measurement data is represented by minimum, maximum and mean value with its standard deviation

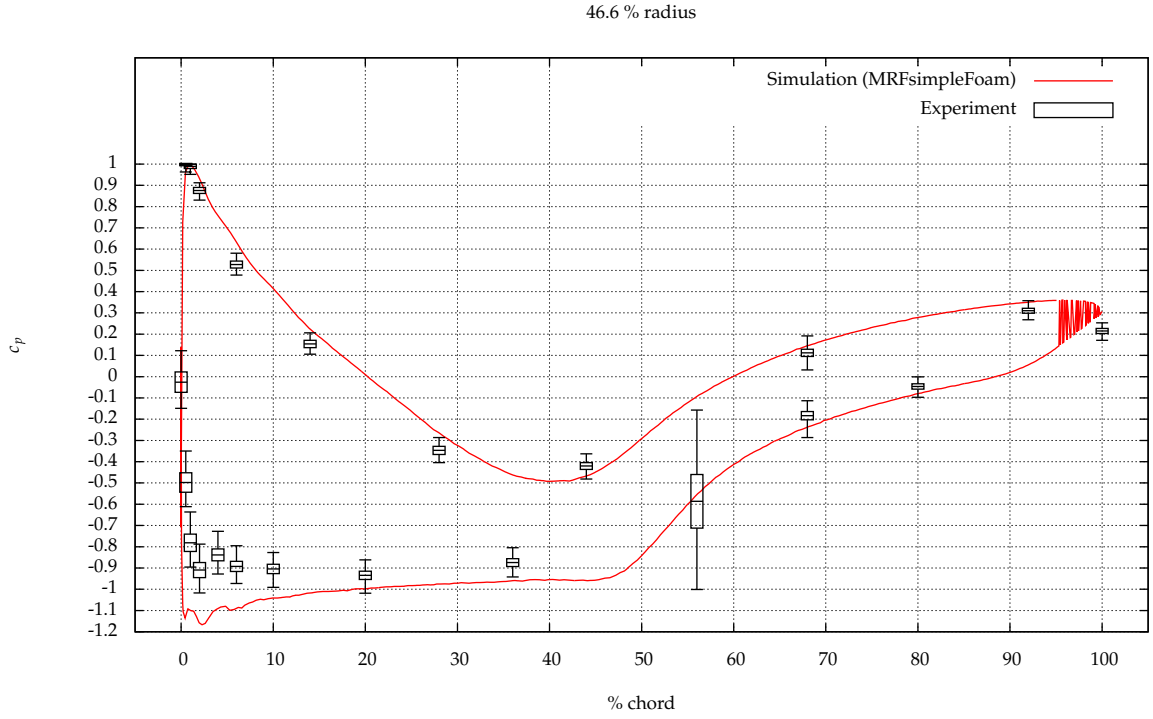


Figure F.2: Normalized pressure coefficients at 46.6 % radius = 2.3435 m: Simulation results (step 1/5, `MRFSimpleFoam`, 5840 iterations) vs. NREL UAE Phase VI (sequence H, yaw angle 0°, wind tunnel velocity 5 $\frac{m}{s}$, repetition 0) measurements. Measurement data is represented by minimum, maximum and mean value with its standard deviation

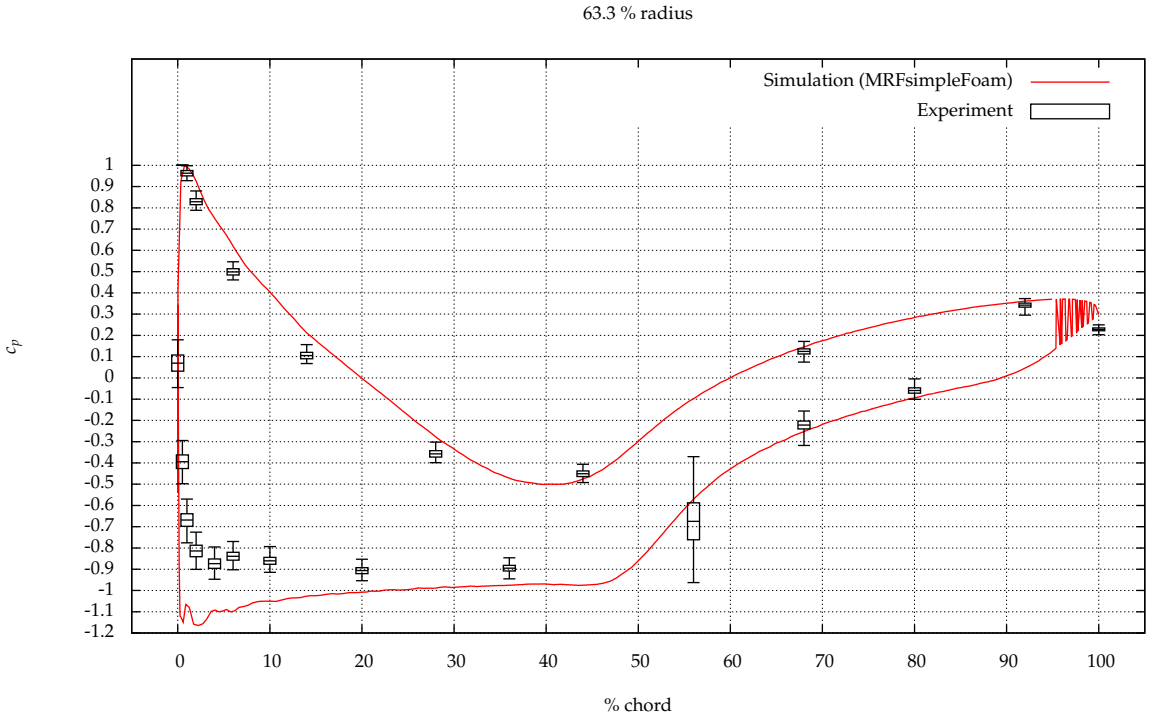


Figure F.3: Normalized pressure coefficients at 63.3 % radius = 3.1834 m: Simulation results (step 1/5, `MRFSimpleFoam`, 5840 iterations) vs. NREL UAE Phase VI (sequence H, yaw angle 0°, wind tunnel velocity 5 $\frac{m}{s}$, repetition 0) measurements. Measurement data is represented by minimum, maximum and mean value with its standard deviation

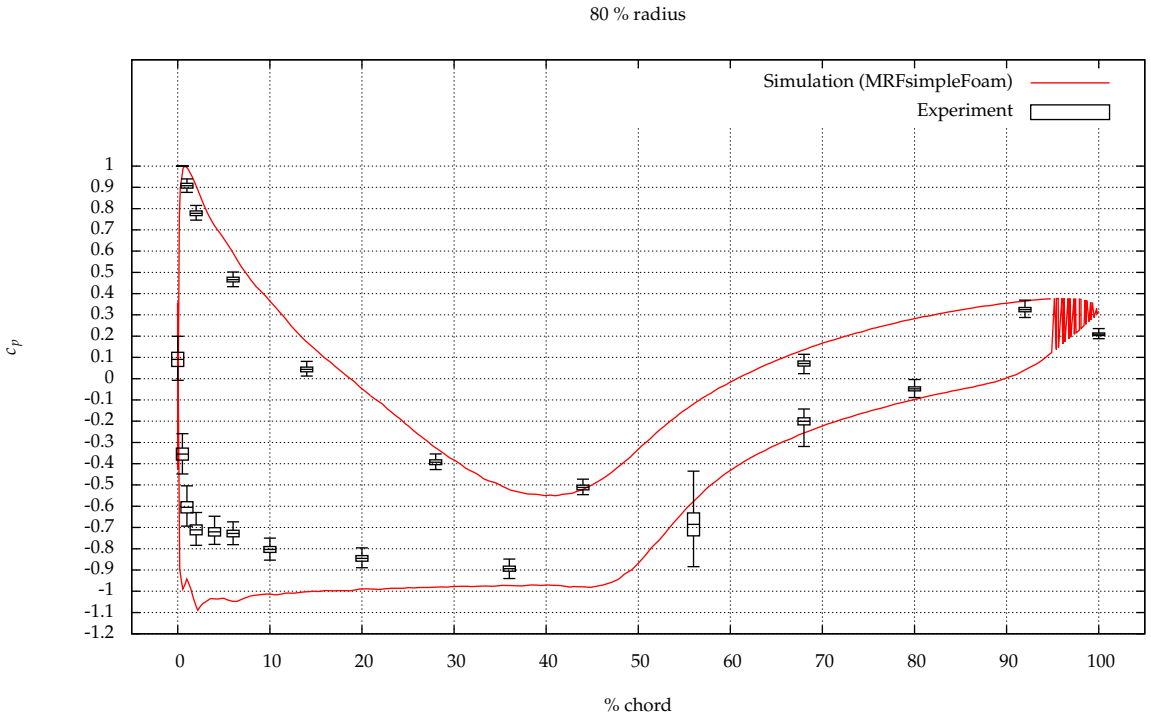


Figure F.4: Normalized pressure coefficients at 80 % radius = 4.0232 m: Simulation results (step 1/5, `MRFSimpleFoam`, 5840 iterations) vs. NREL UAE Phase VI (sequence H, yaw angle 0°, wind tunnel velocity 5 $\frac{m}{s}$, repetition 0) measurements. Measurement data is represented by minimum, maximum and mean value with its standard deviation

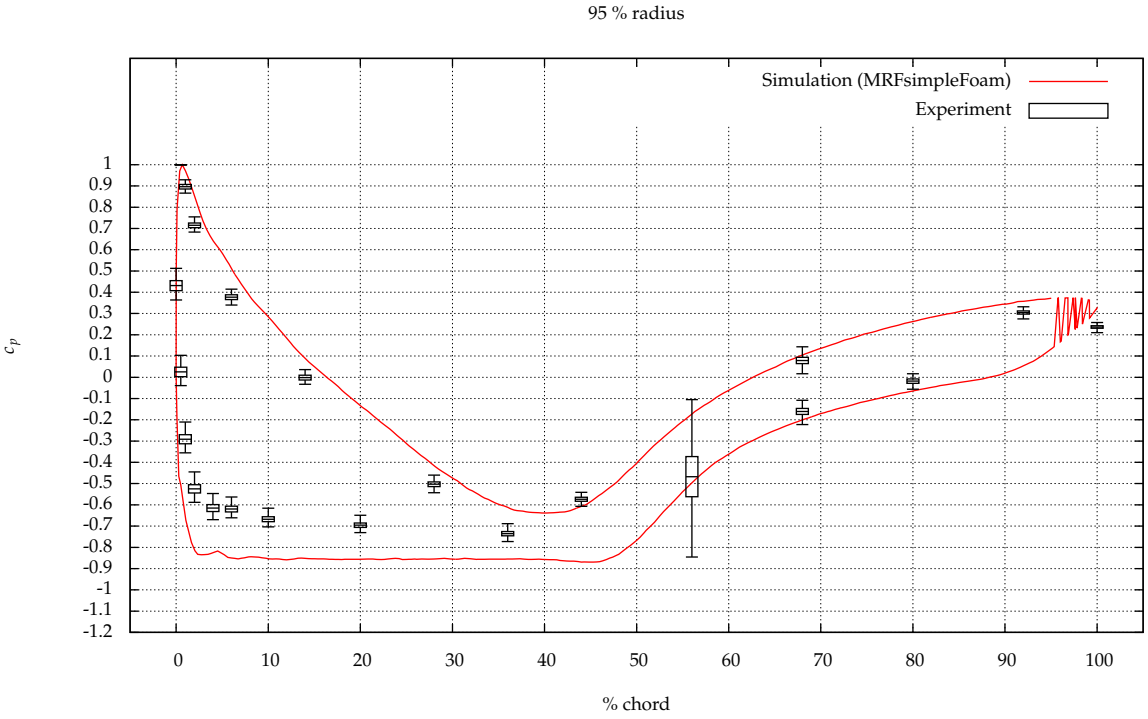


Figure F.5: Normalized pressure coefficients at 95 % radius = 4.7776 m: Simulation results (step 1/5, `MRFSimpleFoam`, 5840 iterations) vs. NREL UAE Phase VI (sequence H, yaw angle 0°, wind tunnel velocity 5 $\frac{\mathrm{m}}{\mathrm{s}}$, repetition 0) measurements. Measurement data is represented by minimum, maximum and mean value with its standard deviation

G

Full Case Structure with All OpenFOAM Dictionaries etc. for Steps 2/5, 3/5 or 4/5

An overview of the fully needed case structure to do simulations of type step 2/5, 3/5 and 4/5 is given in Figure G.1 on the next page. Included are also output files of most interest. Only for input files different or additional to those of a step 1/5 simulation examples are presented here. It must be noticed, that the final settings still depend on further developments.

Listing G.1: $EMPIRE_CASE/OF/0/pointDisplacement

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       pointVectorField;
    location    "0";
    object      pointDisplacement;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 0 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            slip;
    }

    outlet
    {
        type            slip;
    }
```

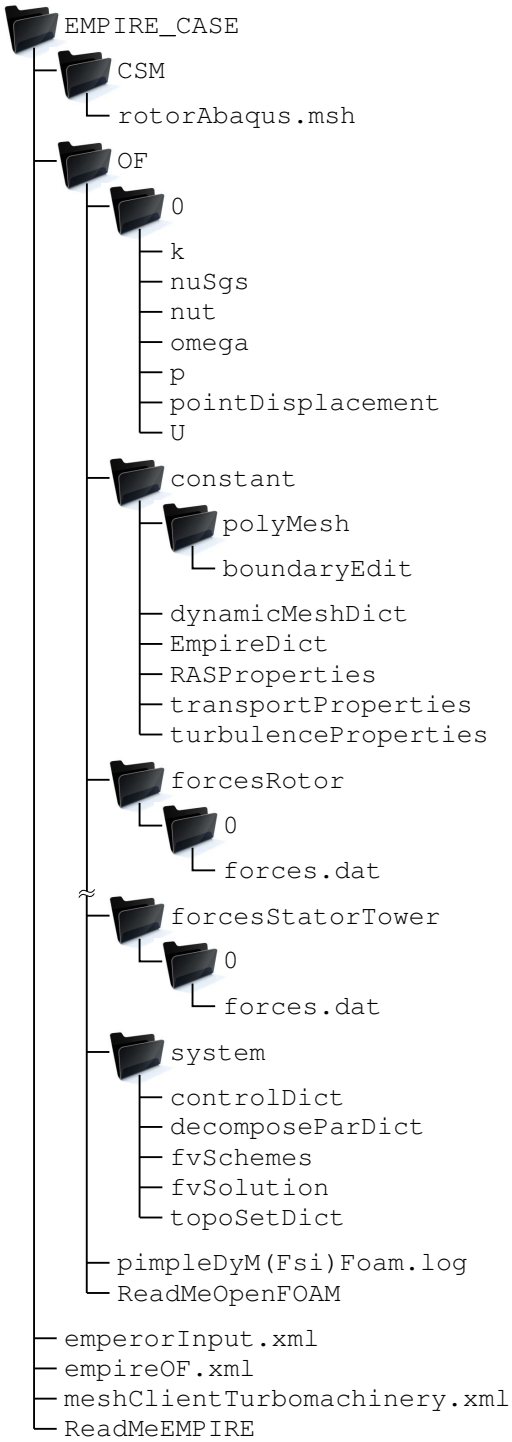


Figure G.1: Full case structure for step 2/5, 3/5 or 4/5 using `pimpleDyMFoam` or `pimpleDyMFsiFoam` with all OpenFOAM dictionaries etc. and output files of most interest

```
33
34      "stator.*"
35      {
36          type            slip;
37      }
38
39      "rotor.*"
40      {
41          type            fixedValue;
42          value           uniform (0 0 0);
43      }
44
45      "AMI.*"
46      {
47          type            slip;
48      }
49  }
50
51  // ************************************************************************* //
```

Listing G.2: $EMPIRE_CASE/OF/0/U

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.1.x                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       volVectorField;
13     location    "0";
14     object      U;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 dimensions      [0 1 -1 0 0 0 0];
19
20 internalField   uniform (0 -5 0);
21
22 boundaryField
23 {
24     inlet
25     {
26         type            fixedValue;
27         value           uniform (0 -5 0);
28     }
29
30     outlet
31     {
32         type            zeroGradient;
33     }
34
35     "stator.*"
36     {
37         type            fixedValue;
38         value           uniform (0 0 0);
39     }
40
41     "rotor.*"
42     {
43         type            movingWallVelocity;
44         value           uniform (0 0 0);
45     }
46
47     "AMI.*"
48     {
49         type            cyclicAMI;
50         value           $internalField;
51     }
52 }
53
54 // ************************************************************************* //
```

Listing G.3: $EMPIRE_CASE/OF/constant/dynamicMeshDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dynamicFvMesh                   CoSimulationMotionSolverFvMesh;

motionSolverLibs ("libfvMotionSolvers.so");
CoSimulationMotionSolverFvMeshCoeffs
{
    cellZone                    cellZone_1; // rotor
    solidBodyMotionFunction     rotatingMotion;
    rotatingMotionCoeffs
    {
        CofG                    (0 0 0);
        radialVelocity          (0 432 0); // in deg/s
    }
}

solver                          displacementLaplacian;
diffusivity                     directional (0.1 10 0.1);
frozenDiffusion                 off;
frozenPointsZone                frozen;

// ************************************************************************* //
```

Listing G.4: $EMPIRE_CASE/OF/constant/EmpireDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

coupledPatchNames   ("rotor.*");
densityFluid        densityFluid [ 0 2 -1 0 0 0 0 ] 1.23;

// ************************************************************************* //
```

Listing G.5: $EMPIRE_CASE/OF/system/controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pimpleDyMFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         60;

deltaT          1e-4;

writeControl    adjustableRunTime;

writeInterval   2e-2;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  yes;

maxCo           2;

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

functions
{
    forcesRotorBlade1
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches (rotorBlade1);
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 0 0);

            outputControl     timeStep;
            outputInterval    1;
    }

    forcesRotorBlade3
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches (rotorBlade3);
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 0 0);

            outputControl     timeStep;
            outputInterval    1;
    }

    forcesRotorHub
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches (rotorHub);
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 0 0);

            outputControl     timeStep;
            outputInterval    1;
    }

    forcesRotor
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches ("rotor.*");
```

```
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 0 0);

            outputControl     timeStep;
            outputInterval    1;
    }

    forcesStatorTower
    {
        type    forces;
        functionObjectLibs ("libforces.so");
        patches (statorTower);
        rhoName rhoInf;
        rhoInf  1.23;
        CofR    (0 -1.401 -12.164); // with reference to root

            outputControl     timeStep;
            outputInterval    1;
    }
}

// ************************************************************************* //
```

Listing G.6: $EMPIRE_CASE/OF/system/fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default     Euler;
}

gradSchemes
{
    default     Gauss linear;
    grad(p)     Gauss linear;
    grad(U)     Gauss linear 1;
}

divSchemes
{
    default     Gauss linear;
    div(phi,U)  Gauss linearUpwind grad(U);
    div(phi,k)  Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div(phi,omega)  Gauss upwind;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default     Gauss linear corrected;
}

interpolationSchemes
{
    default     linear;
    interpolate(HbyA) linear;
}

snGradSchemes
{
    default     corrected;
}

```

```
fluxRequired
{
    default     no;
    pcorr       ;
    p           ;
}

// ************************************************************************* //
```

Listing G.7: $EMPIRE_CASE/OF/system/fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    pcorr
    {
        solver          GAMG;
        smoother        DIC; // GaussSeidel;
        preconditioner  FDIC;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration off;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels      1;

        tolerance        0.02;
        relTol           0;
    }

    p
    {
        $pcorr;
        tolerance       1e-06;
        relTol          0;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|nuSgs|omega)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-3;
        relTol          0.1;//1e-6;
    }

    "(U|k|epsilon|nuSgs|omega)Final"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-5;
        relTol          0;
    }

    cellDisplacement
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-03;
```

```
        relTol          0;
    }


    cellMotionUx
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-03;
        relTol          0;
    }
}

PIMPLE
{
    correctPhi          yes;
    momentumPredictor   yes;
    nOuterCorrectors    1;
    nCorrectors         1;
    nNonOrthogonalCorrectors 2;
    pRefCell            0;
    pRefValue           0;

    residualControl
    {
        p
        {
            tolerance 6e-05;
            relTol 0;
            absTol 0;
        }
         // p               1e-5;
         // U               1e-5;
         // nuTilda         1e-10;
    }
}

relaxationFactors
{
    fields
    {
    }
    equations
    {
        "(U|k|epsilon|nuSgs).*" 1;
    }
}

// ************************************************************************* //
```

Listing G.8: $EMPIRE_CASE/emperorInput.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- EMPIRE input file -->
<EMPEROR xmlns="http://www.example.org/emperorInput" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.org/emperorInput emperorInput.xsd">
  <clientCode name="meshClientTurbomachinery">
    <mesh name="myMesh1">
      <dataField name="displacements" location="atNode"
        dimension="vector" typeOfQuantity="field" />
      <dataField name="forces" location="atNode" dimension="vector"
        typeOfQuantity="fieldIntegral" />
    </mesh>
  </clientCode>
  <clientCode name="OpenFOAM">
    <mesh name="myMesh1">
      <dataField name="displacements" location="atNode"
        dimension="vector" typeOfQuantity="field" />
      <dataField name="tractionsElem" location="atElemCentroid"
        dimension="vector" typeOfQuantity="field" />
      <dataField name="tractionsNode" location="atNode"
        dimension="vector" typeOfQuantity="field" />
    </mesh>
  </clientCode>

  <dataOutput name="output1" interval="1">
    <dataFieldRef clientCodeName="meshClientTurbomachinery"
      meshName="myMesh1" dataFieldName="displacements" />
    <dataFieldRef clientCodeName="meshClientTurbomachinery"
      meshName="myMesh1" dataFieldName="forces" />
    <dataFieldRef clientCodeName="OpenFOAM" meshName="myMesh1"
```

```
      dataFieldName="displacements" />
    <dataFieldRef clientCodeName="OpenFOAM" meshName="myMesh1"
      dataFieldName="tractionsElem" />
    <dataFieldRef clientCodeName="OpenFOAM" meshName="myMesh1"
      dataFieldName="tractionsNode" />
  </dataOutput>

  <mapper name="mortar1" type="mortarMapper">
    <meshA>
      <meshRef clientCodeName="meshClientTurbomachinery" meshName="myMesh1" />
    </meshA>
    <meshB>
      <meshRef clientCodeName="OpenFOAM" meshName="myMesh1" />
    </meshB>
    <mortarMapper oppositeSurfaceNormal="true" dual="false" enforceConsistency="true" />
  </mapper>

  <connection name="transfer displacements">
    <input>
      <dataFieldRef clientCodeName="meshClientTurbomachinery"
        meshName="myMesh1" dataFieldName="displacements" />
    </input>
    <output>
      <dataFieldRef clientCodeName="OpenFOAM" meshName="myMesh1"
        dataFieldName="displacements" />
    </output>
    <sequence>
      <filter type="mappingFilter">
        <input>
          <dataFieldRef clientCodeName="meshClientTurbomachinery"
            meshName="myMesh1" dataFieldName="displacements" />
        </input>
        <output>
          <dataFieldRef clientCodeName="OpenFOAM" meshName="myMesh1"
            dataFieldName="displacements" />
        </output>
        <mappingFilter>
          <mapperRef mapperName="mortar1" />
        </mappingFilter>
      </filter>
    </sequence>
  </connection>
  <connection name="transfer forces">
    <input>
      <dataFieldRef clientCodeName="OpenFOAM" meshName="myMesh1"
        dataFieldName="tractionsElem" />
    </input>
    <output>
      <dataFieldRef clientCodeName="meshClientTurbomachinery"
        meshName="myMesh1" dataFieldName="forces" />
    </output>
    <sequence>
      <filter type="locationFilter">
        <input>
          <dataFieldRef clientCodeName="OpenFOAM" meshName="myMesh1"
            dataFieldName="tractionsElem" />
        </input>
        <output>
          <dataFieldRef clientCodeName="OpenFOAM" meshName="myMesh1"
            dataFieldName="tractionsNode" />
        </output>
      </filter>
      <filter type="mappingFilter">
        <input>
          <dataFieldRef clientCodeName="OpenFOAM" meshName="myMesh1"
            dataFieldName="tractionsNode" />
        </input>
        <output>
          <dataFieldRef clientCodeName="meshClientTurbomachinery"
            meshName="myMesh1" dataFieldName="forces" />
        </output>
        <mappingFilter>
          <mapperRef mapperName="mortar1" />
        </mappingFilter>
      </filter>
    </sequence>
  </connection>

  <coSimulation>
    <sequence>
      <couplingLogic type="timeStepLoop">
        <timeStepLoop numTimeSteps="100000">
          <dataOutputRef dataOutputName="output1" />
        </timeStepLoop>
```

```
        <sequence>
          <couplingLogic type="connection">
            <connectionRef connectionName="transfer displacements" />
          </couplingLogic>
          <couplingLogic type="connection">
            <connectionRef connectionName="transfer forces" />
          </couplingLogic>
        </sequence>
      </couplingLogic>
    </sequence>
  </coSimulation>
  <general>
    <portFile>server.port</portFile>
    <verbosity>INFO</verbosity>
  </general>
</EMPEROR>
```

Listing G.9: $EMPIRE_CASE/empireOF.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- EMPIRE input file -->
<EMPIRE>
  <code name="OpenFOAM">
    <type>field</type>
  </code>
  <general>
    <portFile>server.port</portFile>
    <verbosity>DEBUG</verbosity>
  </general>
  <userDefined>
    <couplingType>looseCoupling</couplingType>
    <sendDataField>traction</sendDataField>
  </userDefined>
</EMPIRE>
```

Listing G.10: $EMPIRE_CASE/meshClientTurbomachinery.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- EMPIRE input file -->
<EMPIRE>
  <code name="meshClientTurbomachinery">
    <type>field</type>
  </code>
  <general>
    <portFile>server.port</portFile>
    <verbosity>DEBUG</verbosity>
  </general>
  <userDefined>
    <GiDMeshFile>CSM/rotorAbaqus.msh</GiDMeshFile>
    <numTimeSteps>1000000</numTimeSteps>
    <couplingType>looseCoupling</couplingType>
  </userDefined>
</EMPIRE>
```

H

Development of the Velocity Isosurface in Step 2/5

Vortices, which are developing at the tip of both blades during rotation, get carried downstream with the main flow. This phenomenon is visualizable in simulations by generating an isosurface for a velocity value slightly bigger than the mean free stream value, which is here about 15.6 $\frac{\mathrm{m}}{\mathrm{s}}$. The transient development of the resulting helix is illustrated in form of a timeline.

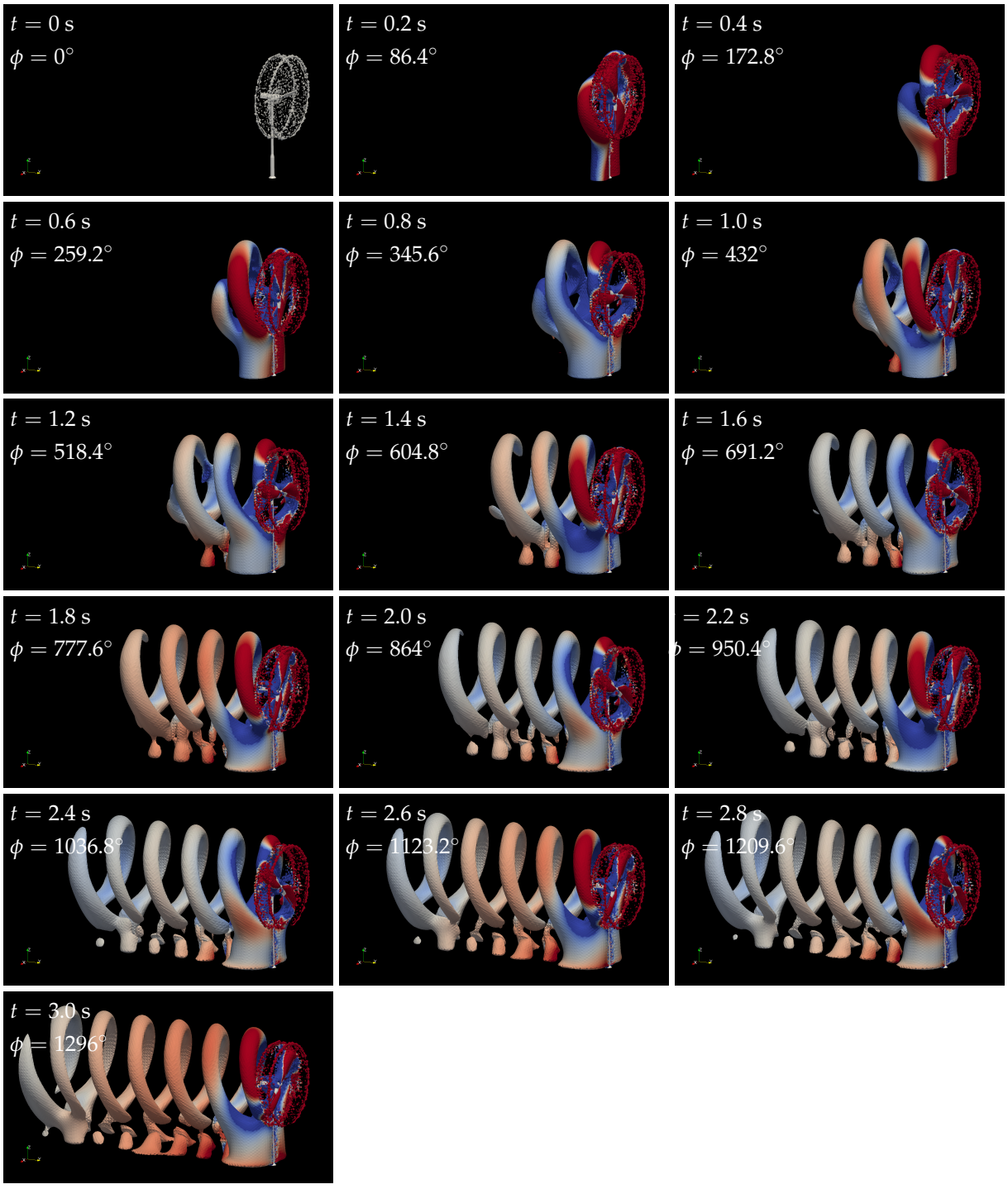


Figure H.1: Vortices developing at the rotating blades' tip and getting carried downstream with the main flow are visualized from $t = 0$ to 3.0 s using an isosurface of $|\mathbf{u}| \approx 15.6 \frac{\text{m}}{\text{s}}$

List of Figures

List of Tables

Listings

Bibliography

[1] Saxena A.
Guidelines for Specification of Turbulence at Inflow Boundaries.
Accessed 11/03/2013.
ESI CFD Customer Support.
URL: http://support.esi-cfd.com/esi-users/turb_parameters/.

[2] National Aeronautics and Space Administration (NASA).
NASA Ames Research Center.
Accessed 01/03/2013.
URL: http://www.nasa.gov/centers/ames/.

[3] Kassiotis C.
Which strategy to move the mesh in the Computational Fluid Dynamic code OpenFOAM.
Cachan Cedex, France: École Normale Supérieure de Cachan, 2008,
P. 14.
URL: http://perso.crans.org/kassiotis/openfoam/movingmesh.pdf.

[4] CD-adapco.
CD-adapco.
Accessed 13/03/2013.
URL: http://www.cd-adapco.com/.

[5] CD-adapco.
STAR-CCM+. User Guide. Version 7.06.
2012,
P. 12179.

[6] Simms D. et al.
NREL Unsteady Aerodynamics Experiment in the NASA-Ames Wind Tunnel: A Comparison of Predictions to Measurements.
Technical Report NREL/TP-500-29494.
Golden, Colorado: National Renewable Energy Laboraty (NREL), 2001,
P. 51.
URL: http://www.nrel.gov/docs/fy01osti/29494.pdf.

[7] Simms D. et al.
Plans for Testing the NREL Unsteady Aerodynamics Experiment 10-m Diameter HAWT in the NASA Ames Wind Tunnel.
Technical Report NREL/TP-500-27599.
Golden, Colorado: National Renewable Energy Laboraty (NREL), 1999,
P. 315.
URL: http://www.nrel.gov/docs/fy00osti/27599.pdf.

[8] Simms D.A. et al.
Unsteady Aerodynamics Experiment Phases II–IV Test Configurations and Available Data Campaigns.
Technical Report NREL/TP-500-25950.
Golden, Colorado: National Renewable Energy Laboraty (NREL), 1999,
P. 177.
URL: http://wind.nrel.gov/amestest/PhaseII-IVReport.pdf.

[9] Chao D.D. and van Dam C.P.
CFD Analysis of Rotating Two-Bladed Flatback Wind Turbine Rotor.
Sandia Report SAND2008-1688.
Albuquerque, New Mexico: Sandia National Laboratories (2), 2008,
P. 85.
URL: http://prod.sandia.gov/techlib/access-control.cgi/2008/081688.pdf.

[10] J.H. Ferziger and M. Perić.
Computational Methods for Fluid Dynamics.
Vol. 3.
Springer-Verlag Berlin etc., 2012.

[11] OpenCFD Ltd. (ESI Group).
OpenFOAM. The open source CFD toolbox. ESI.
Accessed 13/03/2013.
URL: http://www.openfoam.com.

[12] OpenCFD Ltd. (ESI Group).
OpenFoam. The Open Source CFD Toolbox. Programmer's Guide. Version 2.1.1.
2012,
P. 93.
URL: http://foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf.

[13] OpenCFD Ltd. (ESI Group).
OpenFoam. The Open Source CFD Toolbox. User Guide. Version 2.1.1.
2012,
P. 209.
URL: http://foam.sourceforge.net/docs/Guides-a4/UserGuide.pdf.

[14] OpenCFD Ltd. (ESI Group).
OpenFOAM. The OpenFOAM Foundation. OpenFOAM v2.1.0: Arbitrary Mesh Interface (AMI).
Accessed 13/03/2013.
URL: http://www.openfoam.org/version2.1.0/ami.php.

[15] Jonkman J.M. et al.
Definition of a 5-MW Reference Wind Turbine for Offshore System Development.
Technical Report NREL/TP-500-38060.
Golden, Colorado: National Renewable Energy Laboraty (NREL), 2009,
P. 75.
URL: http://www.nrel.gov/docs/fy09osti/38060.pdf.

[16] Cotrell J.R. and Pierce K.G.
Analysis of the Unsteady Aerodynamics Experiment for National Full-Scale Aerodynamics Complex Testing.
Loads Document Volume 1 of 2.
Golden, Colorado: National Renewable Energy Laboraty (NREL), 1999,
P. 34.
URL: http://wind.nrel.gov/amestest/LoadsAnalysisDocument.pdf.

[17] Hsu M.-C., Akkerman I., and Bazilevs Y.
"Wind turbine aerodynamics using ALE–VMS: validation and the role of weakly enforced boundary conditions".
In: *Computational Mechanics* (2012), pp. 499–511.

[18] Hsu M.-C. and Bazilevs Y.
"Fluid–structure interaction modeling of wind turbines: simulating the full machine".
In: *Computational Mechanics* (2012), pp. 821–833.

[19] Hsu M.C.
"Fluid—Structure Interaction Analysis of Wind Turbines".
PhD thesis. University of California, San Diego, 2012,
P. 195.

[20] Hand M.M. et al.
Unsteady Aerodynamics Experiment Phase V: Test Configuration and Available Data Campaigns.
Technical Report NREL/ TP-500-29491.
Golden, Colorado: National Renewable Energy Laboraty (NREL), 2001,
P. 162.
URL: http://www.nrel.gov/docs/fy01osti/29491.pdf.

[21] Hand M.M. et al.
Unsteady Aerodynamics Experiment Phase VI: Wind Tunnel Test Configurations and Available Data Campaigns.
Technical Report NREL/TP-500-29955.
Golden, Colorado: National Renewable Energy Laboraty (NREL), 2001,
P. 310.
URL: http://www.nrel.gov/docs/fy02osti/29955.pdf.

[22] Sezer-Uzol N. and Long L. N.
"3-D time-accurate CFD simulations of wind turbine rotor flow fields".
In: *AIAA paper* 394 (2006), p. 23.

[23] Sørensen N.N., Michelsen J.A., and Schreck S.
"Navier—Stokes Predictions of the NREL Phase VI Rotor in the NASA Ames 80 ft $\times$ 120 ft Wind Tunnel".
In: *Wind Energy* 5.2-3 (2002), pp. 151–169.

[24] National Renewable Energy Laboratory (NREL).
NREL 10-m Wind Turbine Testing in NASA Ames 80'x120' Wind Tunnel.
Accessed 01/03/2013.
2000.
URL: http://wind.nrel.gov/amestest/.

[25] National Renewable Energy Laboratory (NREL).
NREL, National Renewable Enery Laboratory, Leading clean energy innovation, U.S. Department of Energy.
Accessed 01/03/2013.
URL: http://www.nrel.gov/.

[26] National Renewable Energy Laboratory (NREL).
NREL's National Wind Technology Center Unsteady Aerodynamics Experiment Database.
Accessed 01/03/2013.
2000.
URL: https://www.nrel.gov/extranet/uaewtdata/.

[27] Giguère P. and Selig M.S.
Design of a Tapered and Twisted Blade for the NREL Combined Experiment Rotor. March 1998 — March 1999.
Subcontractor Report NREL/SR-500-26173.
Golden, Colorado: National Renewable Energy Laboraty (NREL), 1999,
P. 32.
URL: http://wind.nrel.gov/amestest/BladeDesign.pdf.

[28] Moradnia P.
Project work for the PhD course in OpenFOAM. A tutorial on how to use Dynamic Mesh solver IcoDyMFOAM.
Göteborg, Sweden: Lunds universitet, Spring 2008,
P. 17.
URL: http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2007/PiroozMoradnia/OpenFOAM-rapport.pdf.

[29] van Rooij R.
Terminology, Reference Systems and Conventions.
Duwind 2001.004.
Delft: Delft University of Technology (TU Delft), 2001,
P. 48.
URL: http://ocw.tudelft.nl//fileadmin/ocw/courses/OffshoreWindFarmEnergy/res00062/Terminology.pdf.

[30] Schreck S.
IEA Wind Annex XX. HAWT Aerodynamics and Models from Wind Tunnel Measurements.
Final Report NREL/TP-500-43508.
Golden, Colorado: National Renewable Energy Laboraty (NREL), 2008,
P. 91.
URL: http://www.ieawind.org/task_FinalReports/Annex%2020%20final%20report%20as%20posted%20OSTI.pdf.

[31] Sicklinger S., Wang T., and Andre M.
"Co-Simulation for Multiple (N) Codes with EMPIRE. Enhanced MultiPhysics Interface Research Engine".
Presentation EADS.
München: Lehrstuhl für Statik, Technische Universität München, 2012.

[32] Sicklinger S. et al.
EMPIRE: A N-Code Coupling Tool for Multiphysic Co-Simulations with OpenFOAM. EMPIRE (Enhanced Multiphysics Interface Research Engine).
Poster 7th OpenFOAM Workshop.
Lehrstuhl für Statik, Technische Universität München, 2012.

[33] Unknown.
Unknown.
Accessed unknown date.
URL: Unknown.

[34] H.K. Versteeg and W. Malalasekera.
An Introduction to Computational Fluid Dynamics. The Finite Volume Method.
Vol. 2.
Pearson Education Limited, 2007.

[35] Bazilevs Y. et al.
"3D simulation of wind turbine rotors at full scale. Part I: geometry modeling and aerodynamics".
In: *International Journal for Numerical Methods in Fluids* 65.1-3 (2011), pp. 207–235.

[36] Bazilevs Y. et al.
"3D simulation of wind turbine rotors at full scale. Part II: fluid–structure interaction modeling with composite blades".
In: *International Journal for Numerical Methods in Fluids* 65.1-3 (2011), pp. 236–253.