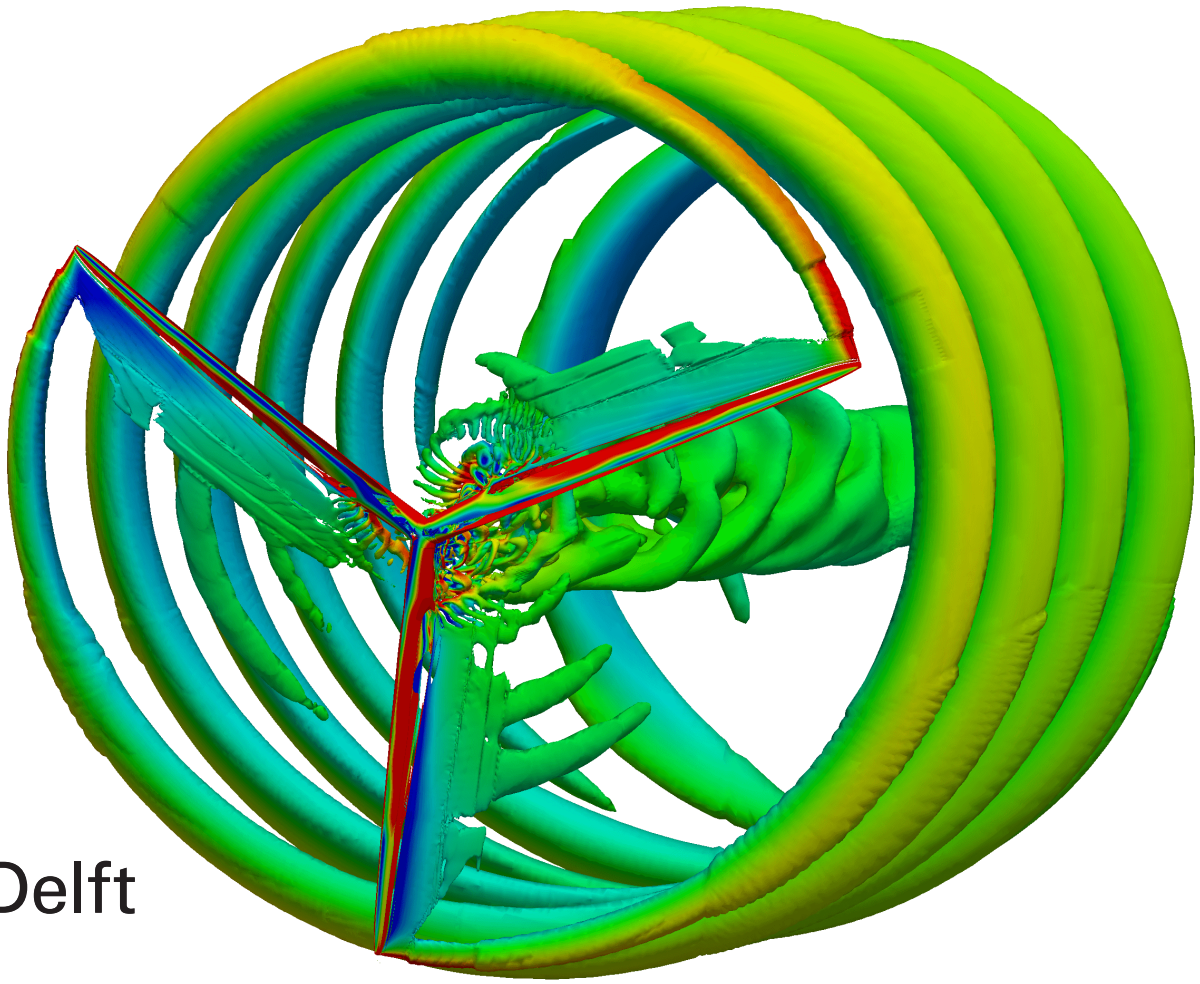
fastFoam

An aero-servo-elastic wind turbine simulation method based on CFD

Max Becker



TU Delft

fastFoam

An aero-servo-elastic wind turbine simulation method based on CFD

by

Max Becker

for obtaining the degree of Master of Science in Aerospace Engineering
at the Delft University of Technology,
to be defended publicly on Tuesday December 5, 2017 at 2:00 PM.

Student number:	4152042	
Date:	November 17, 2017	
Master Project duration:	July 1, 2016 – December 5, 2017	
Supervisor:	Dr. ir. C. J. Simão Ferreira,	TU Delft
	Ir. C. F. Baptista,	TU Delft
	Dr. E. Daniele,	Fraunhofer IWES

Wind Energy Research Group, Faculty of Aerospace Engineering, Delft University of Technology

The Master of Science Thesis was carried out externally at the CFD department of Fraunhofer IWES.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Fraunhofer IWES

Summary

Aero-servo-elastic wind turbine simulation methods which are based on the highest fidelity aerodynamic modelling of computational fluid dynamics (CFD) are currently still relatively rare. Although in the last years some methods were developed, only within one of the most famous wind turbine tools, namely HAWC2, this approach was utilised. In theory, the replacement of lower fidelity methods such as the commonly used engineering blade element momentum (BEM) approach by methods with increased complexity such as CFD could offer advantages. In particular, BEM introduces errors due to inaccurate modelling of its engineering add-ons for the calculation of aerodynamic loads. This occurs especially in complex situations such as the operation in extreme conditions, for instance severe yawing, or modelling of futuristic turbines including very flexible blades probably with flaps or slats.

To investigate and possibly overcome this issues the BEM in the popular aero-servo-elastic tool FAST from NREL was substituted in this project by a more sophisticated method based on CFD. For the CFD method the freely available OpenFOAM package was utilized. It was coupled to FAST via a partitioned coupling approach, where both codes are kept as an own, utilizing the code coupling environment MpCCI developed by Fraunhofer SCAI.

Then, simulation cases were set-up in FAST and OpenFOAM for two turbines, the smaller NREL phase VI and the more modern NREL 5MW turbine. In OpenFOAM this included the time consuming task of the generation of meshes. The simulation cases were carefully chosen in order to support the answering of the research question related to the possible improvement and justification of the implemented CFD based coupled aero-servo-elastic method, hereafter called fastFoam. The main work of this project addresses the development of this method, which includes the implementation of a suitable CFD solver. The task of this solver is to calculate the aerodynamic loads similar to BEM in accordance with the wind turbine state from FAST given by the positions of the blades. In addition, also the coupling of this solver to FAST is addressed, where changes in the FAST were implemented.

After the implementation of the coupled method via MpCCI the preselected cases were simulated based on three different methods, the standalone FAST using BEM, a CFD method with OpenFOAM and finally the developed coupled fastFoam approach also based on CFD. Results for the smaller NREL phase VI turbine, which could be compared to experimental data, showed excellent results for all three methods in normal operating conditions. However, in extreme conditions such as large yawing or pitching the CFD methods showed a better agreement to the experimental data. Especially, in yawing the utilized Pitt/Peters skewed-wake correction model in FAST was unable to predict the load variations over the rotor rotation accurately. However, it could also be concluded that for such a turbine with very stiff blades and no explicit controller a pure CFD method such as OpenFOAM may be sufficient. In addition, it was found that in the coupled method periodic fluctuations in the power, torque and thrust were present only within FAST, these were unresolved in this early version.

Finally, results for the NREL 5MW turbine indicated that for these modern turbines the torque and pitch control is increasingly important and may greatly influence the results. For this turbine with increased blade flexibility the deviations between both CFD methods were increased thus emphasising the importance of aero-servo-elastic modelling. Finally, in yaw a similar picture such as shown for the phase VI was present for the load fluctuations, thus indicating that the yaw models in BEM can still be improved. However, for this turbine no experimental data is available, therefore it cannot be stated which results show the highest validity.

In conclusion, the developed aero-servo-elastic method based on CFD can be justified especially in situations where other methods fail to accurately predict the wind turbine behaviour, which was found to be the case in the extreme yawing or pitching cases. Moreover, there may be more applications which would need to require further investigation such as the modelling of flaps or slats on 10MW+ turbines with even more flexible blades, which was not done due to time constrains in this project. However, it was found that in normal operating conditions methods based on BEM still show excellent results. Concerning the extremely large computational effort of such a CFD method, BEM may still be the first choice for these cases. However, for more special cases and in the detailed design stages a method similar to the implemented one may be wise to conduct due to improved accuracy.

Acknowledgements

This thesis would not have been possible without the support and help of others, who I would like to express my gratitude.

First of all, I would like to thank my daily supervisor at Fraunhofer IWES Dr. Elia Daniele for the support through the entire project from the first generated mesh to the final page of this thesis. In addition, I very much appreciated the support of Bastian Dose, who really pushed me forward with the fastFoam solver in OpenFOAM. Moreover, I also would like to thank Hamid Rahimi and Cherif Mihoubi for the help on the meshing, the solvers and schemes in OpenFOAM as well as on all kind of Ubuntu related problems.

Also I would like to express my gratefulness for the project opportunity to Dr. Bernhard Stoevesandt from Fraunhofer IWES and Klaus Wolf from collaborating institute Fraunhofer SCAI. A special thanks goes out to Dr. Jan Kleinert, formerly from SCAI, for the support with the MpCCI coupling environment. Although we were separated by distance there was a good collaboration and I was especially impressed that really all of my hundreds of emails with multiple questions were well answered.

Furthermore, I am very thankful for the valuable feedback from my supervisors Dr. ir. Carlos Simão Ferreira and Ir. Carlos Baptista at TU Delft. Then, I also would like to thank Dr. Scott Schreck from NREL for providing me the measurement data for the NREL phase VI turbine.

Finally, I wanted to thank all others for the good discussions, especially all the students and employees at Fraunhofer, who I cannot name here because this would get way to long. Last but for me very important, I would like to thank my family for the great support during the months of my thesis.

Max Becker
Oldenburg, November 2017

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AEP	annual energy production
ALE	arbitrary Lagrangian-Eulerian
AMI	arbitrary mesh interface
API	application programming interface
AVATAR	AdVanced Aerodynamic Tools of lArge Rotors
BEM	blade element momentum
BTC	bend-twist coupling
CAD	computer-aided design
CAE	computer-aided engineering
CFD	computational fluid dynamics
CoE	cost of energy
CSD	computational structural dynamics
CV	control volume
CVT	constant volume tetrahedron
DDES	delayed detached eddy simulation
DES	detached eddy simulation
DLL	dynamic link library
DNS	direct numerical simulations
DOF	degree of freedom
DTU	Technical University of Denmark
FAST	Fatigue Aerodynamics Structures and Turbulence
FEM	finite element method
FSI	fluid structure interaction
GEBT	geometrically exact beam theory
GGI	generalized grid interface
GUI	graphical user interface
HAWC2	Horizontal Axis Wind turbine simulation Code 2nd generation
HMB2	Helicopter Multi-Block
HPC	high performance computers
IEC	International Electrotechnical Commission
IWES	Institute for Wind Energy and Energy System Technology
LES	large eddy simulation
MBD	multibody dynamics
MIC	multi-iterative coupling
MpCCI	Mesh-based parallel Code Coupling Interface
MPI	message passing interface
MRF	multiple reference frame
NREL	National Renewable Energy Laboratory
OBJ	Wavefront Object
OpenFOAM	Open Source Field Operation and Manipulation
P-FSI	practical fluid-structure interaction

PISO	pressure-implicit split-operator
RANS	Reynolds-averaged Navier–Stokes
SAM	spring analogy method
SCAI	Scientific Computing and Algorithms Institute
SIMPLE	semi-implicit method for pressure-linked equations
SOWFA	Simulator fOr Wind Farm Applications
SST	shear stress transport
STL	Stereolithography
TFI	transfinite interpolation
VLES	very large eddy simulation

Latin Symbols

c	Speed of Sound	m/s
C	Sutherland's constant	K
C_p	Pressure coefficient	-
C_M	Pitching moment coefficient	-
C_{Th}	Thrust force coefficient	-
C_{Tq}	Torque force coefficient	-
dt	Time step	s
$d\psi$	Azimuth step	deg
F_{Th}	Thrust force	N/m
F_{Tq}	Torque force	N/m
g_i	Gravitational acceleration	m/s^2
I	Turbulence intensity	-
k	Turbulence kinetic energy	m^2/s^2
l	Turbulent length scale	m
M	Pitching moment	Nm/m
p	Pressure	Pa
q	Quaternion	-
R	Rotor radius	m
R	Gas Constant	$J\ kg^{-1}K^{-1}$
R	Rotation matrix	-
t	Time	s
T	Temperature	K
T_{ref}	Reference temperature	K
u_i	Velocity in Cartesian coordinate direction	m/s
$\bar{u}_i$	Mean part of velocity	m/s
u_i'	Fluctuating part of velocity	m/s
U	Velocity	m/s
x_i	Cartesian coordinates	m

Greek symbols

α	Angle of attack	deg
γ	Adiabatic index	-
μ	Dynamic viscosity	Pa s
μ_{ref}	Reference viscosity	Pa s
ν	Turbulent eddy viscosity	m^2/s
$\tilde{\nu}$	Spalart-Allmaras variable	m^2/s
ρ	Air density	kg/m^3
τ_{ij}	Viscous surface stress tensor	Pa
ω	Specific dissipation rate	1/s
Ω	Rotational speed	rpm

1 Introduction

The currently available state-of-the-art wind turbine simulation tools are mainly based on blade element momentum (BEM) aerodynamic modelling, see for instance the tools Horizontal Axis Wind turbine simulation Code 2nd generation (HAWC2), BLADED or Fatigue Aerodynamics Structures and Turbulence (FAST). This low-fidelity method is in principle only valid for steady two dimensional flow [37]. However, by using engineering correction methods it is generally applied throughout the entire wind turbine operational conditions [37]. These engineering correction methods or engineering add-ons account for root and tip corrections as well as for yawed, respectively sheared inflow or tower shadow [37]. Due to the nature of these correction methods errors are inevitable introduced. While the magnitude of these errors may still be acceptable for most cases, for extreme operational conditions they might exceed a certain threshold.

In addition, the BEM aerodynamic modelling of wind turbines is not yet validated for the future generation of wind turbines [36]. This is due to the reason that these turbines would have very large blades of at least 100 meters compared to the current designs of about 65 meters [36]. These future turbines would be operating at high tip speeds higher than 110 m/s [36]. At these conditions Reynolds and Mach number effects act, which BEM may not account for [36]. Besides this, these modern turbines may use flaps or slats to increase the power output while actively mitigating loads. However, modelling of flaps and slats is not included into BEM and as such in current wind turbine simulations.

Within the last years a more sophisticated aerodynamic modelling approach for wind turbines has emerged based on computational fluid dynamics (CFD). In CFD the Navier-Stokes equations, which are describing the physics, are solved. Due to the high complexity required for solving these equations it requires high performance computerss (HPCs). Within recent work by Heinz et al., see [14], CFD was utilised within a very popular wind turbine simulation tool namely HAWC2. By replacement of BEM through CFD the code HAWC2CFD was established in [14]. This resulted in improvements concerning the aerodynamic modelling, for instance within root and tip region, or in extreme conditions and added new capabilities such as resolving vortex induced vibrations [14].

Replacing BEM by CFD for wind turbine simulations is not an easy task as it requires coupling between possible high-complexity structural and fluid solvers, including different mesh mapping and communication of variables. Although, such a coupling has been achieved by several researchers before, for the most popular wind turbine design tools this has not been achieved yet, except for HAWC2. An update within the commonly used and freely available tool FAST resulted in improved modularization. This is assumed to simplify the coupling to a CFD code. As CFD code the open source Open Source Field Operation and Manipulation (OpenFOAM) is a highly feasible choice, which has also been heavily used for wind turbine simulations in the last years. In addition, access to the code coupling software Mesh-based parallel Code Coupling Interface (MpCCI), developed by Fraunhofer Scientific Computing and Algorithms Institute (SCAI), is provided, thus further reducing the coupling effort.

Although, it is not possible to fully replace BEM by CFD due to the large computational times, it may be a viable choice for specific applications. In industry a large amount of load calculations, such as given by International Electrotechnical Commission (IEC) requirements, has to be executed. In addition, in the preliminary design phase a lot of iterations and thus computations are present. Therefore, such a method may specifically be useful for the detailed design phase, where a high accuracy is desirable.

Thesis Motivation and Goals The research gap, which is addressed within this project, relates to the utility of such a coupled CFD method especially for modelling extreme operational conditions or future turbines with highly flexible blades possibly equipped with exotic configurations such as flaps or slats.

The motivation for implementing the high fidelity aerodynamic model based on CFD (OpenFOAM) within a state-of-the-art wind turbine computer-aided engineering (CAE) tool such as FAST can be justified by the following benefits:

- Increase the accuracy in aero-servo-elastic modelling of wind turbines in general with a specific aim on the modelling of severe conditions for large futuristic wind turbines.
- For instance due to the higher fidelity aerodynamic model (CFD) improvements in highly unsteady inflow such as large yaw or dynamic stall are possible.
- The modelling of futuristic turbines equipped with flaps or slats may be easier to implement and more accurate with CFD versus BEM, which requires additional engineering add-ons.
- There are new modelling capabilities due to CFD for example allowing to resolve vortex induced vibrations.
- An other advantage is that the current modelling with BEM requires airfoil polar data often obtained through experiments, which with the CFD method is no longer needed.
- In general higher accuracy in the simulations may result in better and more optimized blades thereby reducing the overall cost of energy (CoE) for wind energy.
- Especially improved estimation of the loads and the power (annual energy production (AEP)) may in turn lead to lighter and cheaper blades.
- Results from high fidelity aero-servo-elastic modelling may lead to improvements in low fidelity methods (e.g. engineering add-ons) based on BEM due to new insights.
- The utilized codes such as FAST and OpenFOAM are open source and freely available, therefore such a coupled method and its implementations may have an increased interest.

Thus, the goal of the project is to create an equivalent aerodynamic CFD module within OpenFOAM and couple it to FAST, thereby replacing its standard BEM module. The corresponding research objective can be formulated as: Development of a new aero-servo-elastic wind turbine simulation method by coupling FAST v8 to an OpenFOAM CFD code. The project approach is therefore more practice oriented by dealing with the coupling of the two codes OpenFOAM and FAST. The related research question has been formulated as: To what extend is the use of CFD within FAST for modelling the aerodynamics in extreme conditions justified when comparing its computational complexity relative to BEM?

Thesis Outline The first part of this thesis deals with an in-depth literature review of relevant literature, see Chapter 2. In this literature review the governing aerodynamic methods such as BEM and CFD, as well as structural modelling and finally combined fluid structure interaction (FSI) methods were reviewed. In addition, the different tools to be used which are OpenFOAM, FAST and MpCCI were presented. In the next Chapter (3) the wind turbine simulation cases are discussed. This short Chapter gives a quick overview which wind turbines will be simulated and addresses which methods will be used.

In Chapter 4, the available methods such as the CFD method based on OpenFOAM and the BEM-based FAST approach are outlined. The Section on OpenFOAM deals with all required aspects such as mesh generation, mesh movements and the solving approach. For the aero-servo-hdyro-elastic tool FAST the entailed modelling approaches as well as the solver settings were discussed. Both tools were utilized and coupled via MpCCI to result in the resulting method named fastFoam. Its methodology including the MpCCI coupling process is presented in Chapter 5. Next, the results for the simulations based on the three different methods namely OpenFOAM, FAST and fastFoam are elaborated on in Chapter 6. Finally in the last Chapter, conclusions and recommendations are drawn also in view of the research questions to be answered, see Chapter 7.

2

Literature Review

This chapter deals with an in depth review of literature, which will be relevant for the execution of the project. At first, a general overview of rotor aerodynamics as well as wind turbine elasticity is given. This is related to the modelling techniques of both aspects of which some are used within this research. Afterwards, FSI coupling methods for simulating wind turbines are discussed, combining the interaction of fluid and structure, thus aerodynamics and elasticity. Therefore, this review is focused on the industrial standards for simulating wind turbine aero- and structural dynamics with a detailed part on CFD methods. Finally, an overview of the modelling tools, which are based on the reviewed methods and which will be used within the current project are given.

2.1. Rotor Aerodynamics

This section is about rotor aerodynamics and the currently available modelling methods ranging from low-fidelity, low-complexity methods such as BEM to high-fidelity, high-complexity CFD.

2.1.1. Blade Element Momentum Theory

At first, the state-of-the-art method BEM is discussed, being implemented in several design codes such as FAST, HAWC2 and Bladed for instance. Its theory is explained here based on the book of Hansen, see [12].

2.1.1.1. Actuator Disc Momentum Theory

The blade element momentum theory is based on the one dimensional actuator disc momentum theory, considering an ideal rotor [12]. Therefore, the rotor is modelled as a permeable disc, without exerting friction or vertical velocity on the flow. Due to the pressure drop over the rotor related to the acting drag, a thrust force is obtained. Using the assumptions for this 1D problem that no external force is acting on the rotor, as well as stationary incompressible and frictionless flow, thus no change in internal energy, several simplified equations can be applied to analyze the problem [12].

First of all, the Bernoulli equation is applicable from upstream to closely in front of the rotor (from 1 to 2 in Figure 2.1) as well as from close behind the rotor to downstream (from 3 to 4 in Figure 2.1). Combining both expressions of the Bernoulli equation, one finds a relation for the pressure drop in terms of the upstream and downstream velocities. Furthermore, the continuity equation yielding conservation of mass is used, to obtain a relationship between the cross sectional areas at the rotor and downstream for instance.

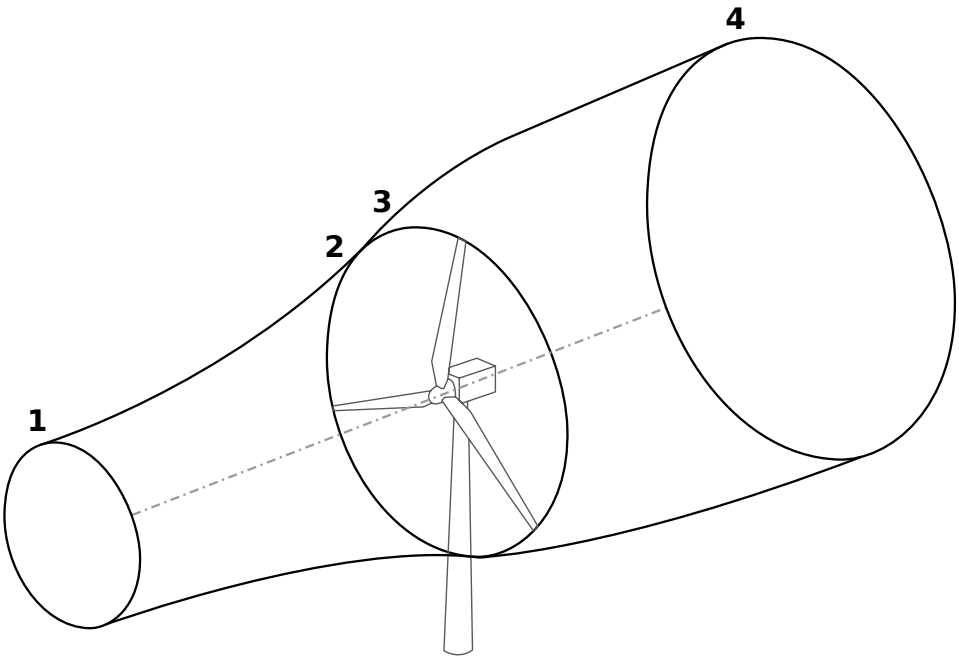


Figure 2.1: Stream-tube for a horizontal axis wind turbine.

Finally, the axial momentum equation can be applied in the direction perpendicular to the rotor on a circular control volume ranging from upstream to downstream (from 2 to 3 in Figure 2.1). By doing so one obtains an expression for the thrust using the previously found relations [12]. From the energy equation the corresponding power can be derived. Using the definition of the axial induction factor, one then obtains the famous expressions for the thrust and power coefficients as a function of axial induction factors [12]. As a next step one can easily find the Betz limit at an induction factor of 1/3. For the exact equations see the book of Hansen [12].

2.1.1.2. Momentum Theory

However, the previously described procedure is only valid for an ideal rotor, the actuator disc. Therefore, the BEM theory considers also the actual geometry of the rotor as well as its number of blades, the twist and chord distribution and the airfoils used [12]. The classical BEM theory originating from Glauert in the year 1935 will be discussed. It is able to calculate the loads as well as the power of the rotor in steady conditions at different operating conditions of wind speed and pitch angle for instance.

At first to go from actuator disc to an actual rotor, the disc is discretized into several annuli. These are assumed to be radially independent and as an initial assumption the relative thrust force at every annuli is assumed to be constant corresponding to an infinite number of blades. This is later corrected by using Prandtl's tip and root loss correction accounting for tip and root vortices, which break up the momentum balance [12]. The momentum equation is then applied to each annuli yielding the thrust, torque and power, due to each element as function of the axial and tangential velocity and induction factor [12].

2.1.1.3. Blade Element Theory

In addition, the thrust, torque and power can also be obtained from the blade elements directly. This is obtained by using the general lift and drag expressions using the known angle of attack [12]. Therefore, the values for the lift and drag coefficient have to be known, which are given from airfoil data at each annuli. The corresponding angle of attack is known and it is related to the inflow angle, which is dependent on pitch and twist angle. Then the obtained lift and drag expressions can be related to normal and tangential force through the inflow angle [12]. As these forces are in units per length, the thrust and moment is obtained by multiplying the force by the number of blades and the radius of the annuli, where these forces have been evaluated [12].

2.1.1.4. Combined BEM Theory

Equalizing the expressions for thrust and torque from both momentum and blade element calculations, one obtains expressions for the axial and azimuthal induction factor as function of rotor solidity, lift and drag coefficient and inflow angle [12]. Then the final BEM solving scheme can be summarized by the following approach. At first, a value for the axial and azimuthal induction factor is assumed, commonly a value of zero is chosen. Then the lift and drag coefficient is calculated from the look up tables of airfoil polar data [12]. Next one can calculate the new induction factors from the previously determined expression. This procedure is repeated until the induction factors stay constant relative to a certain tolerance. The convergence does indicate that the obtained thrust and torque from both momentum and blade element calculations are equal. Thus, a converged solution is obtained. Finally, as one has only obtained the thrust and torque contributions of each annuli, a simple summation over all annuli will result in the thrust and torque of the complete rotor [12]..

2.1.1.5. Engineering Add-ons

The BEM engineering method is due to its simplicity including assumptions, theoretically valid only for stationary two dimensional flow and non-yawed conditions [37]. Therefore, several engineering add-ons are commonly used to be able to analyze three dimensional unsteady flow in extreme conditions like yaw. These add-ons are for instance based on measurements or more advanced methods. An overview of these add-ons is given based on the doctoral thesis of Schepers, see [37], where these models haven been thoroughly investigated.

As mentioned previously, Prandtl's tip and root loss corrections need to be applied. This is due to the reason that there are tip and root vortices for a finite number of blades, which is conflicting with the assumption of an infinite number of blades. Therefore, an additional variable is introduced in the computations of both induction factors mentioned before. This variable is dependent on the location of the annuli on the rotor, as the influences of tip and root vortices are only corrected at inbord and outbord locations [12].

In addition, for axial induction factors larger than approximately 0.4 Glauert's correction is used [12]. This is due to the reason that above induction factors of 0.5 flow reversal occurs and thus momentum theory is invalid. Therefore, the empirical Glauert correction will adjust the relation between thrust coefficient and axial induction factor.

As in reality the presence of turbulence will violate the assumption of steady flow, an unsteady BEM correction is applied [12]. This correction accounts for yawed inflow, thus the case when the mean wind speed is not perpendicular to the rotor area . Due to turbulence this is often the case and therefore its consequences of power loss and load fluctuations have to be included [37].

Besides that, corrections for sheared flow are included, which model the wind shear either logarithmically or according to the power law [37]. Thus, the flow velocity is changing over height.

Next to that, one also has to consider corrections due to applied cone and tilt angle of the turbine, which effectively reduce the rotor area. Also unconventional blade shapes such as pre-swept blades may be modelled through corrections.

Additionally, there is the effect of the tower on the flow leading to the so called effect of tower shadow. For simplicity the tower is modelled cylindrically and the flow around it is modelled as a dipole. Then the induced velocities due to the tower are added to the undisturbed velocities.

Finally, the effect of dynamic stall needs to be included. Due to the unsteady effects mentioned previously the angle of attack undergoes dynamic fluctuations [37]. Especially in the case of trailing edge separation dynamic stall needs to be accounted for. This is corrected within the Beddoes–Leishman model [12].

2.1.2. Computational Fluid Dynamics

Besides low-fidelity methods such as BEM, more sophisticated methods such as CFD exist, which will be discussed within this section.

Computational fluid dynamics, commonly abbreviated as CFD, deals with the discretized form of the Navier-Stokes equations on a computational grid. These discretized equations are solved numerically for each time step on every grid point related to a cell [12]. As the solution is obtained iteratively on a large number of cells the computational time is relatively large [1]. Therefore, HPC with hundreds of cores are used in order to limit the computational time [13].

For the purpose of determining the loads at the rotor, a grid needs to be constructed, for instance in terms of finite volumes around the turbine. Therefore, a finer mesh is required at the blades, where the boundary layer needs to be resolved, while far from the rotor a coarser mesh is sufficient [1]. Moreover, near to the blade tip a highly refined mesh is required in order to accurately reproduce the tip vortices [13].

An additional difficulty appears due to the reason that the rotor is rotating and blades are pitching, thus the mesh should rotate as well [1]. However, a certain benefit is that no airfoil data is needed as the velocity and pressure are directly calculated around the blade surface [1].

There is a quite a difference in whether incompressible or compressible Navier-Stokes equations are solved. Modern wind turbines operate at tip speeds of up to 100 m/s, with a speed of sound of about 340 m/s at sea level conditions. At these operational conditions one is just at the Mach barrier of about 0.3, until which the assumption of incompressible flow is still valid [13]. However, future wind turbines with rated power above 10MW and rotor diameters larger than 200 m/s will show larger tip speeds where compressible flow will be acting, see for instance [36]. For wind turbines both compressible and incompressible Navier-Stokes solvers are used, depending on the actual case to be considered.

In addition, the Navier-Stokes equations are solved not only directly at the rotor, but also in the wake. Therefore, a mesh of the entire outer domain is required. Also turbulence is modelled within methods Reynolds-averaged Navier–Stokes (RANS)-based or large eddy simulation (LES)-based turbulence models.

RANS describes a Reynolds-averaging procedure where flow variables, such as velocity, are divided into a mean term and a fluctuating term [35]. This process is called Reynolds decomposition [35]. This ensemble average is then substituted into the Navier-Stokes equations, which leads to the Reynolds averaged Navier-Stokes equations [35]. However, to be able to solve the system of equations the resulting Reynolds stresses in the equations have to be related to mean flow variables [35]. Therefore, often a so called Boussinesq hypothesis is used relating both variables via the turbulent eddy viscosity [35].

Submodels of the RANS turbulence approach are the famous two equation model $k-\omega$ shear stress transport (SST) and one equation model Spalart-Allmaras.

An alternative to RANS is the LES turbulence model. It is becoming more popular in recent years as it has advantages over RANS in some aspects [35]. In particular, it is well suited for modelling large eddies with turbulent mixing in unsteady and anisotropic flow [35]. However, the disadvantage is that this requires a much larger computational time [35]. The assumption in the model is that smaller eddies can be described universally by a subgrid-scale model, while for the large eddies time consuming calculations have to be set up [35]. LES is especially time consuming as it requires a fine mesh gradient in three dimensions near the wall, while RANS only requires a fine mesh in the wall normal direction [35].

Moreover, a combined approach is also possible called detached eddy simulation (DES). It uses the less time consuming RANS approach near the wall to obtain the boundary layer and the LES approach for the wake [35]. Currently a combination of it with an empirically based transition model is the state of the art in turbulence modelling of wind turbines using CFD [13].

CFD for wind turbine applications is still a relatively new method. CFD results have been compared to data from the National Renewable Energy Laboratory (NREL) Unsteady Aerodynamics Experiment in the year 2000 [13] . The measured data of the two bladed NREL phase VI rotor with 10 m diameter yielded good agreement with the CFD results [13] . It came out that the Navier-Stokes based CFD was actually the best computational method [13]. This was a major step in the development of CFD methods as there validity for wind turbines was confirmed the first time by measurement [13].

In the next Section a simplified overview of the Navier-Stokes equations is given.

2.1.2.1. Navier-Stokes Equations

The Navier-Stokes equations entain the conservation equations. For incompressible flows such as for wind turbines, where the Mach number is below 0.3, mainly the conservation of mass and momentum are addressed. For compressible flow also the energy conservation equation needs to be included. The Navier-Stokes equations accurately describe the flow of Newtonian fluids. However, only in simplified cases such as channel flows analytical solutions exist [11]. Therefore, numerical methods are utilized to solve the Navier-Stokes equations in there discretized form.

Conservation of Mass The conservation of mass is addressed in the continuity equation stating that mass can neither be created or destroyed. The continuity equation is given on a control volume (CV), which refers to a specific spatial region. It is given in Cartesian coordinates according to Equation 2.1 [11].

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_i)}{\partial x_i} = \frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_x)}{\partial x} + \frac{\partial \left(\rho u_y\right)}{\partial y} + \frac{\partial (\rho u_z)}{\partial z} = 0 \tag{2.1}$$

Where x_i (with i from 1 to 3) respectively (x, y, z) are Cartesian coordinates and u_i or u_x, u_y, u_z are the corresponding velocities. For incompressible flows the change in the density ρ over time t can be neglected.

Conservation of Momentum In addition to the conservation of mass, there is conservation of momentum. Following the CV approach the forces acting on the fluid are aconsidered. This includes surfaces forces such as pressure, normal and shear stresses as well as surface tension and body forces for instance due to gravitation, centrifugal or Coriolis effects [11]. Making the assumption of Newtonian fluids the change in momentum according to Newton's second law equals the sum of forces on the CV.

In its general form the momentum equation is given according to Equation 2.2 if gravity is considered as the only body force [11].

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial \left(\rho u_i u_j\right)}{\partial x_j} = \frac{\partial \tau_{ij}}{\partial x_j} - \frac{\partial p}{\partial x_i} + \rho g_i \tag{2.2}$$

Where τ_{ij} is the viscous surface stress tensor specified on each face of the CV in every direction (j from 1 to 3 respectively x, y, z). The pressure is indicated by p. The gravity term g_i corresponds to the component of gravitational acceleration in the direction of the cartesian coordinate x_i, normally refering to the z-direction.

The momentum equation (Equation 2.2) needs to be written in the three cartesian directions (x, y, z) for each CV. This is due to the different contributions of the surface forces on the CV, leading to a total of three equations.

Together with the continuity equations and the three momentum equations a set of four equations needs to be solved for the incompressible Navier-Stokes equations. As there are more unknowns than equations additional assumptions and relations such as the Boussinesq approximation, ideal gas law or Stokes equations are used for solving the Navier-Stokes equations [11]. For a more comprehensive overview on these relations and there derivation the reader is refered to the book of Ferziger and Peric, see [11].

2.1.2.2. Reynolds-averaged Navier–Stokes Equations

The Navier-Stokes equations can be solved numerically by means of direct numerical simulations (DNS). The flows investigated in engineering applications for instance for wind turbines are often not laminar, but turbulent. Turbulent flows can be described by highly unsteady behaviour in three spatial dimensions. In contradiction to laminar flow, turbulent flow layers interact with each other and mix. The fluid particles form so called turbulent eddies, which break up over time into smaller eddies and influence others. However, the fluctuation in length and time scales of these eddies is highly unpredictable. This is the main reason why DNS simulations of turbulent flows are a difficult and especially computationally time consuming task [11].

Therefore, other simulation strategies in CFD have emerged, which include simplifications but allow for a compromise between accuracy and computational time. One of the most common approaches relates to the so called RANS simulations, where the Reynolds-averaged Navier–Stokes Equations are solved. Within Reynolds-averaging, based on Osborne Reynolds, the unsteadiness is averaged by decomposing the flow properties in a mean part and a fluctuating part over an ensemble [11]. This ensemble averaging procedure allows for the inclusion of unsteady effects compared to a time averaging process. Then for the solution it is solved for the mean part, whereas the turbulent part is modelled by engineering models, the so called turbulence models such as for instance k - ω SST or Spalart-Allmaras. For an overview of these models see the book of Ferziger and Peric [11].

The introduction of turbulence models is required as no closed set of equations is formed by the RANS equations (closure problem). The Reynolds decomposition for the velocity components is shown in Equation 2.3.

$$u_i = \bar{u}_i + u_i' \tag{2.3}$$

Where the $\bar{u}_i$ term addresses the mean part and the u_i' term relates to the fluctuations.

Applying the Reynolds decomposition for other flow parameters as well one obtains the RANS Equations (for incompressible flow) for continuity, see Equation 2.4 [11].

$$\frac{\partial(\rho\bar{u}_i)}{\partial x_i} = 0 \tag{2.4}$$

Similarly, the momentum equation can be obtained such as shown in Equation 2.5 if no body forces are considered [11].

$$\frac{\partial(\rho\bar{u}_i)}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\bar{u}_i\bar{u}_j + \rho u_i' u_j'\right) = \frac{\partial\bar{\tau}_{ij}}{\partial x_j} - \frac{\partial\bar{p}}{\partial x_i} \tag{2.5}$$

In addition to RANS models, other approaches exist such as LES models or even hybrid methods for instance DES. In LES only the largest eddies are fully resolved, whereas the small scale motions are approximated. As mostly RANS will be used in this project for resolving the wind turbine aerodynamics, the interested reader is referred to Ferziger and Peric for further information on other models, see [11].

2.1.2.3. Solution Algorithms

To apply CFD simulations appropriate solvers are required which contain a solution algorithm to solve the Navier-Stokes equations in their discretized format. It can be distinguished between compressible and incrompressible solver and the related algorithms, see for instance the implemented solvers within OpenFOAM [41]. A main difference between the solvers for compressible and incompressible cases is that within incompressible solvers each flow variable is solved separately. In this paragraph is is focused on the solution of incompressible cases where such a highly iterative procedure is applied.

Two of the most common solution algorithms for incompressible flows are the so called semi-implicit method for pressure-linked equations (SIMPLE) and the pressure-implicit split-operator (PISO) algorithm. In addition, the PIMPLE algorithm exists combining both of them [41].

These algorithms can be explained by the following procedure according to Ferziger and Peric [11]:

1. Use solution for u_i^n and p^n as initial estimation to calculate fields u_i^{n+1} and p^{n+1} at t_{n+1}.
2. Solve momentum equations (Eq. 2.2) for velocity components u_i^{m*}.
3. Solve the pressure-correction equation for p' and get u_i' from there relation.
4. Apply the velocity and pressure corrections and solve the continuity equation (Eq. 2.1) yielding the velocity field u_i^m and the updated pressure p^m.
5. Start again at step 2 with u_i^m and p^m as estimates for u_i^{n+1} and p^{n+1}. Repeat until the convergence tolerances are met.
6. Move on to the next time step.

In the aforementioned procedure the superscript m indicates the iteration number for the inner iterations. The asterisk for the obtained velocity components u_i^{m*} indicates that it is not the final value for iteration m, which is u_i^m. The vertical dash ($'$) as a superscript indicates the velocity respectively pressure correctors, which are yielding the updated values for u_i^m and p^m according to Equation 2.6 [11].

$$u_i^m = u_i^{m*} + u_i' \qquad \text{and} \qquad p^m = p^{m-1} + p' \tag{2.6}$$

While the SIMPLE and PISO algorithms essentially solve the same equations (Navier-Stokes), there are differences in the looping (iterative) procedure. The SIMPLE algorithm commonly includes under-relaxation terms α_p and α_u for the pressure and velocity correctors in Equation 2.6 to improve the

convergence behaviour [11]. Under-relaxation is not required for the PISO and PIMPLE algorithms. In addition, for the PISO algorithm a second pressure-correction equation is solved after step 4 where the pressure and velocity are corrected again [11].

2.1.3. Other Methods

Besides the low-complexity BEM and high-complexity CFD, there are other methods of which some of them combine aspects of both BEM or CFD. These are for instance so called actuator disc or actuator line models as well as (free) vortex wake models, which will be discussed in this section.

2.1.3.1. Actuator Disc and Line Models

Acuator disc and actuator line models are a combination of BEM and CFD methods [13]. The rotor is not modelled using a mesh, instead the loads are calculated from airfoil data using the same approach such as in BEM, see paragraph about blade element theory in Section 2.1.1. These loads are then represented on the computational grid. The flows reaction to the forces can then be calculated to obtain the flow variables, such as pressure and velocity, at all grid points (cells) as a result of the aerodynamic loads and the related induced velocities at the rotor. The procedure can be summarized as follows: Navier-Stokes equations are solved and replace the momentum theory part used in BEM, while at the rotor loads are still calculated from airfoil data such as in blade element theory of BEM [13].

An advantage is that no engineering add-ons, for instance for dynamic wake or yaw, are needed as their physical basis is already included within the Navier-Stokes equations [13]. Actuator disc models are different to actuator line models, as they distribute the forces over the entire rotor. For actuator line models this distribution is limited to the blade positions only. The actuator disc methods are commonly used for simulating entire wind farms, while actuator lines models represent the actual physical situation of one wind turbine in more detail [13].

2.1.3.2. Vortex Wake Models

Vortex wake models use the Biot-Savart law to calculate the velocity at every point in space induced by the vorticity field [13]. The resulting induced velocity is added to the free stream velocity in order to obtain the velocity of the flow at a certain point [13]. This procedure is repeated at every point in space to resolve the entire flow field. Generally the Biot-Savart law satisfies the Navier-Stokes equations [13]. However, in practice, to reduce computations, discretized vortex filaments with a certain circulation being constant over a predetermined area are considered. The circulation thereby equals the average vorticity multiplied by this area [13].

By using the Kutta-Joukowski theorem and known airfoil lift data, the bound circulation along the wind turbine blade is calculated and shed into the wake [13]. Thereby Kelvin's theorem, stating that the total circulation of the flow is conserved and Helmholtz's theorem, describing that a vortex filament cannot start or end within the flow, have to be satisfied [13]. As mentioned before, the method still relies on airfoil data and due to its medium complexity it has larger computational times than BEM [13].

2.2. Wind Turbine Elasticity

Similar to modelling the aerodynamics concerning the flow around the wind turbine, it is required to model the structural deformations of the turbine. The structure is not rigid and as a result of its elasticity it will deform. These deformations include deflections and rotations, which typically, for wind turbines, can be quite large at the blade tips.

Therefore, the elasticity of the turbine will be modelled by using computational structural dynamics (CSD) methods. For wind turbines the most common methods are finite element method (FEM), lumped parameter method, modal analysis and multibody dynamics (MBD) modelling. These methods will be shortly discussed within this section to give an overview of their approach and the differences and which of them will be used.

A common method which is generally applicable to all kinds of structures is FEM [31]. Within FEM the structure is divided into several small elements. The elements are represented through nodes, which may lie on the interior or on the boundary of the structure [31]. Interaction only takes place at the boundary as the structural properties at the interior of the element are considered constant [31]. Properties such as thickness, density, stiffness or shear modulus are allocated to the element nodes. Furthermore, the displacements and rotations of the nodes are given.

FEM is a commonly used approach for studying single structures such as blades. This is often done using simplified approaches such as beam models, for instance using Euler–Bernoulli or Timoshenko beam theory. Notice that FAST, which will be used within this research project, uses geometrically exact beam theory (GEBT) within its BeamDyn blade structural solver, see [42]. Thus, a FEM method is applied to obtain the highly elastic response of the blade.

Besides FEM there is an approach called lumped parameter method. It discretizes non-uniform moving bodies, such as the drive train, into known bodies such as the rotating rotor, shafts, gears and generator rotor [31]. This is done by allocating their known inertia and stiffnesses. As this method is commonly used for drive trains it will not be further discussed within this section.

Next to that, there is the modal analysis method. Within the modal analysis approach the degree of freedom (DOF) of the system are reduced by division into smaller sections [31]. These sections are then analyzed and natural frequencies and mode shapes are determined related to eigenmodes. This follows a procedure using generalized coordinates in order to solve the uncoupled equations of motion of the vibrating system [31]. The solution of the modal equations corresponding to the eigenmodes are then added by superposition in order to resemble the entire system [31]. Typically a low order of eigenmodes is sufficient to physically represent the system, thus reducing computational effort making the method highly efficient.

Finally, there is the MBD approach which is used for modelling of motions of mechanical systems with more components, here called bodies [31]. Bodies will move with respect to others, for instance think about the nacelle moving on top of the tower. Bodies are interconnected by links and can be modelled rigid or flexible. The characteristics of the bodies are modelled much more accurate compared to lumped parameter method [31]. Loads and deflections of the bodies are communicated and influence body motions. Constituting from Newton's second law the dynamic equations of motions of the bodies are solved, but for complex structures such as wind turbines special numerical techniques are required [31]. Current wind turbine design tools such as FAST, HAWC2 or Bladed are based on this approach.

To conclude, there are four major approaches in wind turbine structural modelling. Current state-of-the-art wind turbine design software, such as FAST or HAWC2, mainly use the MBD approach. This is probably due to its decent physical representation and computational efficiency. In addition, for highly elastic components such as blades more refined methods such as FEM have emerged accounting for the large deformations. These are then implemented in the design codes by using beam theories for the blades, while using multiple bodies for the rest of the turbine.

2.3. FSI Coupling Methods for Wind Turbine Simulations

As has been presented previously there are several ways to analyse rotor aerodynamics. Each of the methods have certain advantages and drawbacks. In addition to these aerodynamic methods, there are structural solvers accounting for the elasticity of the bodies. To account for the interaction of the fluid and the structure, commonly referred to as FSI, coupling methods between fluid and structural solvers exist. In wind turbine simulation software, such as for instance FAST or HAWC2, most commonly used is the simplified coupling of BEM with MBD. However, due to some limitations of BEM and the increase in computational power, some high-fidelity FSI methods emerged within the last years for wind turbines as well. Although such methods have already been used for instance for helicopters, the application to wind turbines was relatively new, due to difficulty of modelling and the large number of load cases to be examined.

Moreover, there exist coupling methods with a medium complexity for the aerodynamic modelling between BEM and CFD. For example Hsu and Bazilevs coupled a lower order FEM method for aerodynamics to a isogeometric analysis for structural mechanics to resolve wind turbine FSI [16].

Within this project it was chosen to focus on high-fidelity CFD, due to this reason the literature study on FSI coupling methods is focused on coupling of CFD to structural solvers. Therefore, coupling methods using high-fidelity CFD and structural modelling for wind turbine simulations will be presented in the following section.

FSI Computations for Geometrically Resolved Rotor Simulations Using CFD [14] Heinz et al. coupled the Technical University of Denmark (DTU) developed codes HAWC2 and EllipSys3D to obtain the so called HAWC2CFD code, which is used for full rotor simulations [14]. HAWC2 is an aero-hydro-servo-elastic simulation code for horizontal axis wind turbines. By coupling it to the finite volume CFD

solver EllipSys3D, the BEM based HAWC2 is developed into the CFD based HAWC2CFD. The code EllipSys3D solves the RANS equations combined with a k-ω SST turbulence model.

The chosen coupling scheme is a loose coupling scheme, which is not energy conservative. Within the loose coupling scheme the fluid and structural solver communicate their solutions once per time step. A strong tightly coupled scheme was not necessary as it did not show a huge difference according to investigations. The source codes of both HAWC2 and EllipSys3D were untouched, instead a generic coupling framework was established using python and its extension OpenMDAO. By following this procedure partitioned coupling is achieved.

The resulting framework transfers the deflections from HAWC2 to EllipSys3D and the aerodynamic forces from EllipSys3D to HAWC2. As the structural solver within HAWC2 uses one dimensional beam elements, whereas EllipSys3D exploits a three dimensional mesh, there needs to be a transformation. This is achieved by the fact that only the information at the blade sections corresponding to beam elements is exchanged. Thus, the CFD solvers output needs to be transformed onto these beam elements by integration and interpolation. Similarly, the calculated structural deflections are projected onto the mesh by assuming rigid blade cross sections, followed by interpolation of translations and rotations onto the mesh vertices. The obtained deflections at the mesh vertices are thereby used to transform the mesh, according to the determined rotational and translational movement.

Coupled CFD/MBD Method with Application to Wind Turbine Simulations [26] Within his Ph.D. dissertation at University of Iowa Li coupled the incompressible CFD code CFDShip-Iowa v4.5 with the MBD code Virtual.Lab Motion [26]. CFDShip-Iowa v4.5 is calculating the aerodynamic response either by solving unsteady RANS or DES equations. The motion of the turbine is solved by using the MBD code Virtual.Lab Motion. Virtual.Lab Motion uses generalised coordinates and Euler parameters to express the equations of motion in Cartesian coordinate system. For modelling turbulence a delayed DES approach based on a k-ω SST turbulence model was implemented within the CFD code.

The implemented coupling method communicates the obtained forces and moments from the CFD code to the MBD code for all bodies via in- and output files. Similarly, the positions and rotations as calculated by the MBD code are sent to the CFD code. These are then used by the CFD code to move the mesh. The tower and blades are represented by one dimensional bodies. As the one dimensional bodies and the finite volume three dimensional CFD grid do not match, a special procedure was applied. This procedure accounts for the CFD cell area, related to each body, to calculate the forces and moments, for instance applied on a certain body. Similarly, the motions in terms of positions and rotations from the bodies are projected onto the CFD cells to move the CFD grid accordingly.

As the CFD and MBD time steps are not necessarily the same, interpolation will be applied to the forces and moments from the CFD solvers actual and previous solution. This is done as the MBD time step may be smaller and thus an updated value of forces and moments needs to be known at each of these small time steps. It is determined by interpolation to avoid increase of computational time due to CFD. At a certain communication time the results of CFD and MBD code are then communicated once, thus following a loose coupling strategy.

Time-Accurate Aeroelastic Simulations of a Wind Turbine in Yaw and Shear Using a Coupled CFD-CSD Method [44] Yu and Kwon coupled a CFD and CSD solver to obtain a method for aeroelastic simulations of wind turbines within their studies at Korea Advanced Institute of Science and Technology [44]. Thereby, aerodynamic loads along the blade were determined by the CFD solver, for which an in-house one was used based on RANS. Therefore, the incompressible Navier-Stokes equations were solved including k-ω SST turbulence modelling. In order to account for the blade motion and deformation a combined overset and deforming mesh technique was applied. For the mesh a background mesh was used for the wind field, including tower and nacelle. For each blade a moving sub-mesh is used.

Next to the CFD solver also an in-house CSD solver has been used to account for elasticity of the blades. It was based on nonlinear Euler-Bernoulli beam theory applied by discretizing the blade in finite elements. It solves the nonlinear equations of the blade elastic motions. This system of equations includes the aerodynamic loads such as obtained by the CFD solver. However, for initializing of the coupled analysis a BEM module is included.

The method of coupling between CFD and CSD follows the loose-coupling delta-airload method. The aerodynamic loads obtained through CFD and the blade deformations, such as obtained by CSD,

are exchanged after each rotor revolution periodically. First the blade deformations are obtained through CSD based on the aerodynamic loads from BEM module. If periodic blade response has converged the deformations are communicated to the CFD solver as well as the pitch angle. Then the CFD solver calculates the aerodynamic loads along the blade for the deformed blade. The obtained aerodynamic loads are used by the CSD solver again. The procedure is repeated several times until convergence is met, thus when the loads from BEM and CFD remain unchanged.

Investigating Aeroelastic Performance of Multi-MegaWatt Wind Turbine Rotors Using CFD [6] Corson et al. used the commercial CFD solver AcuSolve coupled with a modal structural analysis to simulate a 13.2MW rotor with 100-meter blade length, see [6]. The resulting coupled tool was compared with FAST and WT_Perf, a BEM tool from NREL.

AcuSolve is used for several engineering problems by utilizing the Galerkin/Least-Squares FEM. For the analysis of wind turbines the incompressible RANS equations were solved including the implementation of a Spalart-Allmaras turbulence model. For transient simulations a delayed-DES approach has been used. For mesh motion, for instance due to rotation of the rotor, the transformations are applied to update the position of each node at each time step. To include three dimensional motions, for example due to elasticity, a new technique was applied.

Instead of a directly coupled FSI approach between a fluid and structural code, a practical fluid-structure interaction (P-FSI) approach is used within AcuSolve. Within this P-FSI approach the structural deformation is modelled by superposition of vibrational modes. By limiting the analysis to a certain number of eigenmodes, the linear finite element system of equations simplifies a lot and can be solved for the accelerations, velocities and displacements of each node. Moreover, multi-iterative coupling (MIC) method is used to converge the fluid and the structure iteratively at concurrent time steps, thereby also increasing stability.

The advantage of this P-FSI approach is that it does not require mesh interpolation between different fluid and structural meshes, as it is not based on direct coupling of fluid and structural solvers. Using this approach a modal representation of the blade obtained from FEM will be forwarded to the fluid solver.

To include the mesh motion of the blade the surrounding mesh nodes were projected onto the blade surface. In addition, all nodes which are close to 30 m of the blades were forced to move with the blade, while all other nodes in the entire domain were stationary.

For steady state simulations the flow solver was iterated until convergence was met. In case of transient simulations two iterations were done at every time step in order to ensure a certain convergence. Then the calculations were repeated at the next time step .

Aeroelastic Analysis of Wind Turbines Using a Tightly Coupled CFD–CSD Method [2] Carrion et al. developed a tightly coupled CFD-CSD method by using two aeroelastic approaches within the compressible Helicopter Multi-Block (HMB2) solver developed by Liverpool University [2]. The validated HMB2 code solves the RANS equations using the arbitrary Lagrangian-Eulerian (ALE) formulation in time domain with moving boundaries. For steady simulations a source term is added and no grid rotation is applied. Different turbulence models such as k - ω, k - ω SST including Scale-Adaptive Simulation modelling are integrated into HMB2.

For structural modelling two aeroelastic methods, namely a decoupled one for steady-state and a dynamically coupled method for unsteady problems, are implemented into HMB2. Within the decoupled method aerodynamic loads from CFD are forwarded to the CSD solver, which is based on FEM using NASTRAN. For the unsteady method both aerodynamic loads and structural deformations are calculated in parallel dynamically coupled. In this case a modal approach is used based on blade eigenmode shapes and frequencies calculated via NASTRAN using a blade structural mode.

Within the analysis two blades were considered, which are the famous NREL phase VI turbine blade and MEXICO project turbine blade. The blades were modelled as a beam in NASTRAN by placement of non-linear elements with known structural properties. Additionally, rigid elements were placed to account for interpolating beam deflections to blade surface, which is used for the dynamic movement of the mesh.

In the steady approach the obtained aerodynamic loads from the converged CFD solution are used within NASTRAN as static loads on the structural elements. The FEM solution then results in the blade deformations.

In the unsteady case modal approach the blade shape is approximated as a sum of eigenvectors consisting of displacements multiplied by their amplitude. A differential equation yields the solution for the modal amplitudes at each unsteady time step similar to the NS equation CFD solver. A tightly coupled approach is used with a dual time-step method including pseudo-time stepping yielding the blade shape from the calculated converged modal amplitudes.

The mesh motion in the code HMB2 is achieved using a three stages approach. First, each fluid node along the blade surface is moved linearly with the nearest structural element (constant volume tetrahedron (CVT) method). Then dilatation of blocks and skewness is reduced via spring analogy method (SAM). Afterwards a transfinite interpolation (TFI) is used to interpolate the faces and blocks from boundary vertices and surfaces based on a certain weight.

A New, High-Fidelity Offshore Wind Turbines Aeroelasticity Prediction Method with Significant CPU Time Reduction [15] Horcas et al. used the commercial CFD solver FINE/Turbo coupled with a linear structural solver [15]. The structural solver utilises structural properties of the blade, such as natural frequencies and mode shapes. These were obtained by FEM prior to the actual analysis. These structural properties were then used to obtain the blade elastic deformation by the structural solver, which is directly integrated into the flow solver.

Within FINE/Turbo the RANS equations were solved for the steady case, whereas for the unsteady flow a non-linear harmonic approach has been used. The advantage of this approach is that the computational time decreases, as it is assumed that the unsteadiness is occurring periodically. By solving the flow equations in frequency domain a significant decrease in computational time was achieved.

Although the Non-linear Harmonic approach was utilised the first time for offshore wind turbines, it could be concluded that it showed decent results. Especially the computational time was relatively low with 8 hours to 4 days on 8 processors for the different set-ups (steady versus unsteady).

2.4. Tools

Within this research project an early selection of tools to be used has been made. This includes the CFD code OpenFOAM, which will be coupled to NREL FAST via the neutral interface MpCCI. In order to obtain a better overview of what these tools actually do, relevant literature has been reviewed in this section.

2.4.1. OpenFOAM for Wind Turbine Simulatons

OpenFOAM is an object-oriented library written in C++ for CFD and in some aspects also structural analysis. It is freely available, open-source and actively developed by the community. It allows for relatively simple implementations of complex physical problems. Thereby, it supports the handling of difficult geometries by mesh generation, an efficient solution by several implemented solvers using discretization methods as well as implemented post-processing and data analysis tools [21]. For an overview of OpenFOAM and its capabilities see Figure 2.2.

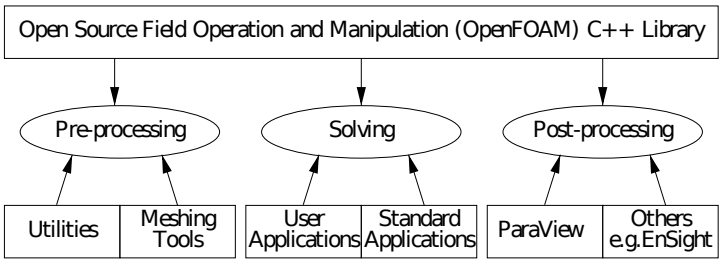


Figure 2.2: Overview of OpenFOAM [41].

In OpenFOAM a discretization of space and time is achieved by creating a mesh in space, while dividing the time dimension into several time-steps [21]. For mesh generation built-in tools such as blockMesh or snappyHexMesh are available in order to generate finite volume meshes [21]. Also external meshing tools can be used. OpenFOAM uses polyhedral meshes by generating cells from lists of faces, which are composed by an ordered number of points in cartesian coordinates. Boundary

conditions are applied via so called patches, which are formed simply by boundary faces. Besides that, multiple functions, such as dynamic mesh motion for instance, are available in OpenFOAM greatly increasing its flexibility.

In OpenFOAM several physical modelling libraries are already included such as turbulence modelling through RANS using k - ω models or Reynolds stress transport models, Newtonian and non-Newtonian viscosity models, material property models, combustion models and so on [21]. Moreover, there exists a large amount of modelling libraries from researchers from different fields.

Moreover, in OpenFOAM stand-alone physical solvers are implemented. As such customized and optimized solvers do exist for the physical application to be solved. Next to that, own solvers can be easily implemented from available tool-kits. The parallelization of solvers is supported using the message passing interface (MPI) approach [21].

In total, OpenFOAM capabilities include, besides others, mainly a high-performance linear algebra package with integrated fluid flow solver for complex physics including integrated post-processing tools [21]. OpenFOAM also allows for multi-physical problems such as FSI applications through its support for self-contained coupled simulations including mesh mapping. Overall its object oriented approach has certain advantages over monolithic function approaches, due to its modularity and flexibility [21].

OpenFOAM has been used for several years now in wind energy research and industry, especially for aerodynamic simulations of wind turbines. Due to its great flexibility it has been used for different applications in the field of wind energy.

For instance, Churchfield et. al performed aerodynamic, system-dynamic, and structural-dynamic simulations of two wind turbines located behind each other at different turbulent inflow winds at neutral and or unstable atmospheric conditions [5]. Therefore, they used two actuator line turbine models coupled with NREL FAST system and structural dynamics models [5]. These turbine models were placed into the atmospheric boundary layer simulated by LES using OpenFOAM [5]. Thus, their approach combines the simulation of atmosphere and simplified wind turbine modelling [5].

Song and Perot did CFD simulations of the NREL Phase VI turbine using OpenFOAM by utilizing the pimpleDyMFoam solver, which solves the RANS equations [39]. For turbulence modelling the Spalart-Allmaras turbulence model has been used, solving for turbulent eddy-viscosity [39]. The mesh was divided into an inner rotating refined cylinder including the rotor and an outer mesh representing the wind tunnel [39]. The rotor motion is represented by the mesh motion of the cylinder mesh, which is achieved using generalized grid interface (GGI) approach [39]. CFD simulations were performed at zero yaw angle, but with three degree pitch angle at different uniform inflow wind speeds [39]. Results from these simulations were matching well with experimental data [39].

Utilizing OpenFOAM for CFD simulations of wind turbines has been done by several researchers. In addition to the aforementioned research, one could mention the research of Stovall et al. [40] concerning wakes or the research of Kirrkam et al. [25], who simulated a 7.5MW wind turbine within OpenFOAM.

In addition to the previous applications, there are also different specific applications of OpenFOAM in the area of wind energy. These are for instance airfoil optimization, see for instance the work of Schramm et al. [38], floating offshore turbine related research, see Liu et al. [28], vertical axis wind turbines see Zamani et al. [45] or wind turbine noise modelling see Czajka et al. [7]. This shows that OpenFOAM is very flexible and due to its open-source approach an active community has been established, leading to several improvements and new functionalities.

2.4.2. NREL FAST

The National Renewable Energy Laboratory of the United States abbreviated as NREL is actively developing a wind turbine CAE tool called FAST. It is an aero-hydro-servo-elastic tool capable of simulating entire (offshore) wind turbines in several operating conditions [23]. FAST is utilised for obtaining performance parameters and doing loads and stability analysis. A major advantage of FAST is that it is open-source and freely available. FAST has been widely accepted also by industry as it has been verified and also has been validated by measurements. An overview of FAST is given in Figure 2.3.

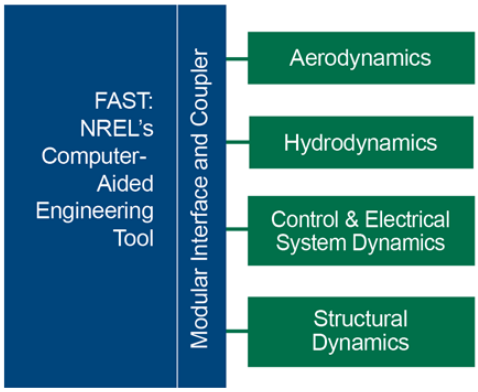


Figure 2.3: Overview of FAST [23].

In the most recent version 8 of FAST, a new modularization framework has been introduced. FAST is composed of several modules each modelling physics through environmental excitations, interactions and dynamic response. It utilizes low-complexity engineering methods to reduce computational effort, as such there are simplifications included.

The main modules used within FAST are AeroDyn for rotor aerodynamics, HydroDyn for hydrodynamics, ServoDyn for control and electrical system dynamics as well as ElastoDyn for structural dynamics [23]. Coupling of these modules results in a tool capable of aero-hydro-servo-elastic analysis of wind turbines. Moreover, there is also a module called TurbSim for generating turbulent inflow wind. The modules interact which each other due to connection of several inputs and outputs. For instance the generated inflow wind field from TurbSim is forwarded to AeroDyn. AeroDyn then uses its BEM solver with engineering add-ons to solve for the aerodynamic loads. These in turn are then communicated to ElastoDyn, whose submodule BeamDyn obtains the deformations of the blade using beam theory.

Due to the improved modularization in FAST version 8 there is the possibility to exchange modules by capable modules with similar inputs and outputs. One example is the replacement of the structural module by MSC.ADAMS [23]. MSC.ADAMS is a commercial software with much higher fidelity as the structural module of FAST. For instance its DOF are relatively unlimited while the standard module within FAST has sharp limits on DOF. Similar levels of model fidelity also exist for the other modules. As such the idea of replacing the low-fidelity standard AeroDyn module of FAST based on BEM by high-fidelity CFD module has been emerged. There is great value of having a tool with a certain range of fidelity within modeling of rotor aerodynamics, hydrodynamics, servo-dynamics and structural-dynamics. This would allow for using an appropriate set of modules with a chosen fidelity closely related to the application the user wants to model.

The new modularization framework enables improved module sharing and implementations and with it flexibility as well as more robust and improved performance [23]. In addition, entire wind farm simulations are now possible by dynamically allocating the respective modules to turbines. The modules are interchanged by a code coupling interface which acts as a driver program. By communication of module-independent inputs and outputs the variables required for the calculations of each module are obtained.

The modules in the new FAST modularization framework are set-up in a state-space formulation. Thus, every module has certain inputs, outputs, states and parameters. For the aerodynamic module AeroDyn the inputs are turbine displacements and velocities, while outputs are aerodynamic loads. In addition, there are states, which are intermediate values, such as the induction factor calculated in BEM. Then there are parameters which include constants such as the turbine geometry or airfoil data or nondisturbed inflow wind.

The new FAST modularization framework supports loose and tight couplings of modules [23]. Whereas in a loose coupling each module uses its own solver for integrating its equations, in a tightly coupled set-up there is a common solver which does this for the entire set of modules. The advantage of loose coupling is that it is computational efficient and allows for implementing code, which is written in other languages from external parties for instance. However, it can lead to numerical problems resulting in errors. In order to implement loose coupling the modules must share a fixed coupling time step. Tight

coupling has certain numerical advantages, but requires a developed form which allows for this type of coupling. In tight coupling modules can have different discrete-time steps, but within each module each states must share the discrete-time step of the module. If different time steps are applied the coupling will be achieved by interpolating and extrapolating the inputs and outputs of the module in time.

Within the new modularization framework also different spatial discretizations for aerodynamic, hydrodynamic and structural modules are allowed [23]. Each module can thus use its own discretization appropriate to the pyhsics it must resolve. Different discretizations are mapped by a developed library, which allows for mapping of one up to three dimensional meshes.

Different modules are coupled by a module interface and coupler [23]. They are directly interfaced to the module interface and coupler. The module interface and coupler connects the different modules inputs and outputs, including the mapping of time and spatial discretizations. It acts as a driver to solve the coupled system and for tight couplings it also integrates the coupled system equations by an own solver. The resulting data is stored in the module interface and coupler, being the main program.

By allocating multiple instances of several modules entire wind farm simulations are possible. By coupling of FAST with OpenFOAM multiple turbines representing a wind farm can be modelled. This modelling includes the aeroelastic turbine interactions as well as wake and array effects modelled through OpenFOAM as a result of the layout of the wind farm.

A software which couples OpenFOAM to FAST for the aforementioned purpose is Simulator fOr Wind Farm Applications (SOWFA), also developed by NREL [4]. However, SOWFA has not been released for the new modularization framework within FAST version 8 at the time of this report. Notice that for this purpose OpenFOAM is mainly utilized for wind farm aerodynamics, while the turbine aerodynamics are still resolved by actuator line turbine models coupled with the turbine dynamics model of FAST.

2.4.3. MpCCI Code Coupling Interface

In multi-physics problems such as fluid-structure interaction of wind turbine blades, simulations from several disciplines need to be included. Therefore, either one tool can be used including a multi-physics formulation or multiple tools can be coupled, where each code corresponds to one discipline. For instance a coupling of a CFD code with a CSD code accounting for fluid-structure interactions is possible. An overview of MpCCI including available codes for coupling is given in Figure 2.4.

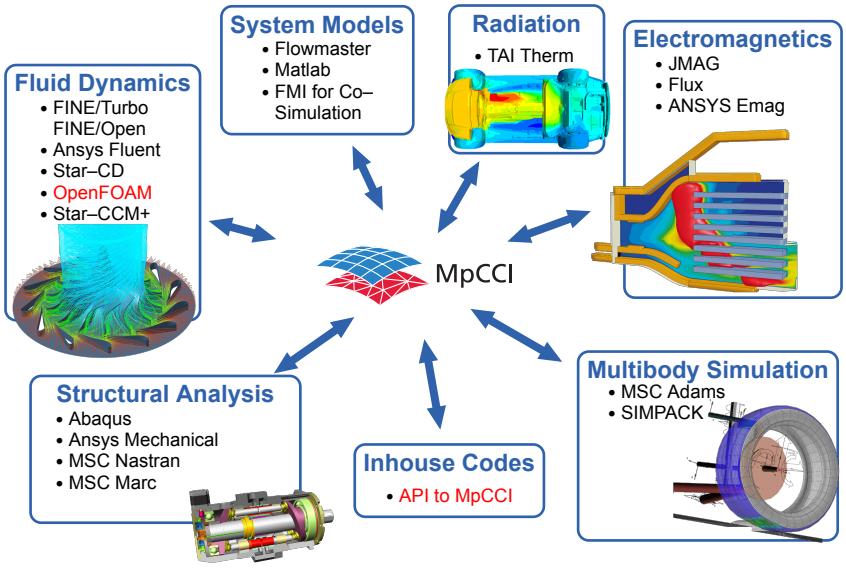


Figure 2.4: Overview of the MpCCI coupling possibilities.

To couple such different codes a code coupling interface is required acting as the driver and main program. The MpCCI code coupling interface is acknowledged as a standard for this simulation code coupling [43]. It is actively developed by Fraunhofer SCAI.

MpCCI is a coupling interface which accounts for the mapping of data between different meshes

of several simulation codes in a coupled domain [43]. The mapping is performed through interpolation and allows for any kind of data exchange. It is relatively independent of the exact application as it fully supports several codes, which can be chosen by the user according to the problem. The codes to be coupled mostly do not have to be adjusted as MpCCI code adapters directly connect to their application programming interface (API). Thus, commercial codes such as for example ABAQUS, Ansys or Fluent can be coupled as well as in-house developed codes.

MpCCI in version 3 uses code adapters for each code which shall be coupled [43]. These code adapters consist of a coupling manager, a communication client and a code driver, which drives the code forward. The coupling manager accounts for the runtime behaviour by reading the model setup and during coupling it controls the boundary conditions of model regions obtained from other codes. The code driver is connected to the coupling manager and the data structures of the code to be coupled. The driver is the only component in the code adapter which is specific to each coupled code as it must be able to access the data of the code. Furthermore, there is a communication client which communicates the data via network to the MpCCI server, where several code adapters are interfaced.

The simulation is configured within the MpCCI GUI, where start and stop environment variables of the simulation are set up as well as which regions to be coupled [43]. Common coupling problems which can be set-up using MpCCI and coupled codes are for instance FSI problems, where a deformable structure, such as for example a flap deforms due to exerted fluid forces. In addition, thermo-electrical couplings, structural-structural couplings up to N-code couplings with several coupling codes can be solved.

2.5. Summary

At first, BEM has been reviewed as a heavily used modelling approach in rotor aerodynamics. BEM is a low-fidelity method in principle only valid for steady two-dimensional flow, it uses engineering add-ons for more universal operational situations [37]. Within its methodology it combines both blade element theory and momentum theory. By convergence of the induction factors, it is indicated that the obtained thrust and torque from both momentum and blade element calculations are equal [12]. Thus, a solution for thrust and torque is obtained using an iterative convergence scheme.

In contrast to this CFD is a high-fidelity, but also highly complex method to implement and to be run. It solves the Navier-Stokes equations, thereby it is generally deemed more accurate. However, due to its complexity it requires huge computational time and also larger set-up time, such as for instance a finite volume mesh needs to be generated first. Its approach includes accurate turbulence modelling. Different turbulence models exist such as RANS, LES, very large eddy simulation (VLES), DES or delayed detached eddy simulation (DDES) for instance. In wind turbine CFD it was found that mostly RANS is applied. CFD may still deliver accurate results for unsteady situations such as extreme yaw or power shutdown. Furthermore, it was found that it does not require airfoil data such as lift and drag polars, which is different compared to BEM.

Next to the previously mentioned methods, BEM and CFD, other methods exist. These were found to be, for instance, actuator disc and line models as well as vortex wake models.

It was found that for wind turbine modelling both a structural solver as well as a fluid solver is required. This is due to interaction between the fluid and the structure, which is a result of the flexibility of the structure. Thus, a simple CFD calculation without structural modelling cannot be accurate for large turbines, where huge structural deformations are interacting with the flow. As such, the fluid-structure interactions need to be carefully modelled.

Within a short review on wind turbine elasticity it was found that similarly to rotor aerodynamics methods with lower as well as with higher complexity and fidelity exist. The most common methods for wind turbine structural modelling seem to be MBD, FEM as well as the lumped parameter method and modal analysis. The MBD approach is very popular and as such also implemented in most wind turbine simulation tools such as HAWC2, FAST or Bladed for instance. FEM is commonly used for the blades using simplified methods, such as Euler–Bernoulli beam theory for example. Moreover, FEM may also be applied using specialist software such as NASTRAN. The lumped parameter method and modal analysis were found to have more specific applications. Within wind turbine simulation tools most commonly a MBD method is used, while the blades are modelled as beam elements through FEM.

Besides aerodynamic and structural modelling, the existence of the controller will have great influ-

ence, which must be included in wind turbine modelling. Therefore, for full aero-servo-elastic computations the three ingredients of aerodynamics, servodynamics and elasticity must be coupled. It was found that the state-of-the-art approach for this modelling is based on coupling MBD and BEM including the controller. Due to BEM limitations in recent years new approaches based on CFD have emerged.

In the literature review part on FSI coupling methods for wind turbine simulations, see Section 2.3, it has been shown that different coupling methods exist for coupling of high-complexity CFD to structural solvers. Within the previously named Section 6 different approaches to couple CFD with structural solvers were reviewed.

A relatively similar approach to a coupling of FAST with OpenFOAM is the coupling of HAWC2 with EllipSys3D such as achieved by Heinz et al. in [14]. This is due to the reason that both, FAST and HAWC2, are popular wind turbine simulation tools and OpenFOAM may be compared to EllipSys3D. Moreover, these are heavily used tools in the field of wind energy.

Within the choice of CFD codes for the six reviewed coupling methods it seems that most of them include RANS models. Although, the code used by Li in [26] is also capable of DES and Corson et al. used DES for transient simulations, see [6]. Coupling methods utilizing CFD codes which use DDES, LES or VLES have not been investigated. Even though such coupling methods may exist, these seem to be not the first choice of models. In addition, most CFD codes are including the $k-\omega$ SST turbulence model. Corson et al. showed that decent results may also be obtained by using the Spalart-Allmaras model, see [6].

For utilizing CFD, meshes needed to be generated first. Most researchers used commercial software or in-house software for this task. However, the generated meshes looked all quite similar being composed of a background mesh and highly refined mesh around the blade.

Similarly, different approaches for dynamic mesh motion were implemented, for instance a three step method used by Carrion et al. [2], or a new developed approach by Corson et al. [6]. In addition, interpolation and integration technique has been used by most researchers for relating the forces from three dimensional CFD mesh to one dimensional beam elements, representing the blade in structural solvers.

For the structural solvers different approaches were used within the coupling methods. The codes of Yu and Kwown were based on FEM using Euler-Bernoulli beam elements, see [44]. Similarly, Carrion et al. [2] used FEM by including the commercial code NASTRAN. Moreover, Corson et al. [6] used a modal shape functions approach for the structural computations. Structural solvers based on MBD were used by Heinz et al. [14] and Li [26].

The methods, which have been investigated for accounting for FSI in wind turbines, mainly use a partitioned approach. Thus, the fluid and structure are solved sequentially by different solvers and information is communicated between them. For the communication a coupling method is implemented.

Although, all methods investigated make use of coupling, the coupling used by the researchers for the codes is different. While most researchers explicitly couple two different codes (partitioned coupling), see for instance the approaches of Li [26] or Heinz et al. [14], Carrion et al. [2] utilize both a CFD and CSD solver implemented in one single code namely HMB2. Similarly Corson et al. [6] and Horcas et al. [15] implemented a CSD method within AcuSolve, resepctively FINE/Turbo. These approaches can be summarized as including a CSD solver into a CFD code to account for the FSI. Then, no coupling between possible different programming languages in CSD and CFD codes must be achieved, thus simplifying the coupling effort.

Furthermore, the coupling methods present are different in terms of loose and tight couplings. Li [26], Yu and Kwon [44] and Heinz et al. [14] used a loose coupling method, which might also be related to their partitioned approach with different flow and structural codes. In contrast to that Carrion et al. used a tightly coupled method, see [2].

Finally, the tools to be coupled (FAST and OpenFOAM) and the coupling interface MpCCI have been reviewed. OpenFOAM was found to be a relatively flexible library, which allows to solve different specific physical problems. In wind energy it is mainly used for modelling aerodynamics using CFD. NREL FAST was identified as aero-hydro-servo-elastic wind turbine design tool being highly modularized. Its state-of-the-art aerodynamic model called AeroDyn is based on BEM. Moreover, the neutral interface MpCCI has been investigated as a highly flexible tool for coupling different discipline codes to solve multiphysical problems.

2.6. Discussion

Within this literature review several studies and methods have been analyzed leading to a discussion within this section. Within the discussion a reasoning for selecting several particular methods used in the project is given.

The literature review section on rotor aerodynamics leads to the reasoning that there is a clear advantage for the high-fidelity CFD over low-fidelity BEM in terms of accuracy, see 2.1. This was found to be the case as BEM is an engineering method only valid for two dimensional steady flow in principle, see [37]. However, within BEM engineering add-ons are applied, which are not always exact for the different operating conditions, especially in extreme conditions such as yawed inflow or emergency shutdown cases. It was found that BEM is heavily used within industry and research, especially due to its low computational effort while generating relatively reliable results. As such it is the state of the art by being implemented in most wind turbine design tools such as FAST, HAWC2 or Bladed. These tools are also used by wind turbine manufacturers, which of course also develop their own versions, as such Siemens for instance created the BEM-based BHawC.

The great advantage of BEM was found to be its low computational time while still being relatively accurate. However, due to its assumptions it was found that there might be problems with the accuracy for the new generation of turbines. For these turbines aerodynamic phenomena such as Mach number effects may occur for which BEM is not yet verified, see [36].

In contrast, the usage of CFD has been increased in recent years especially in research, while in industry its usage is still limited by the computational time making it unfeasible for entire IEC load calculations for instance. As such CFD is in industry mainly used for specific detailed design, for example think of airfoils, but not for entire wind turbine load calculations. Although the drawback of huge computational costs is present in CFD, it is expected that a CFD method integrated into a wind turbine design code such as FAST may lead to improvements. These improvements include the possibility of advanced detailed design, for instance of future turbines including modelling of slats and flaps, using CFD coupled to structural solvers. This gap was identified and reveals the motivation of this project. In addition, findings from these advanced methods may be related back to the engineering methods such as BEM. Thus, increasing the accuracy of engineering add-ons by using results from more advanced methods may be a future benefit, which could be dealt with in a future research project for instance.

Finally, through systematically reviewing studies of CFD based FSI methods for analyzing wind turbines several approaches have been identified. Therefore, it can be questioned what the added value of a coupling method between FAST and OpenFOAM is. Moreover, how could this coupled method be related, in terms of what will be used from the previous research that has been done?

Overall, it was found that most of the implemented coupling methods were based on codes which are not freely available, neither are they open-source. The codes, which will be used in this project, are mostly freely available (FAST and OpenFOAM). Although, the coupling environment (MpCCI) requires a license. Both codes (FAST and OpenFOAM) can be accessed and modified. In addition, both of them have gained a high reputation in recent years in wind energy research. Especially FAST is nowadays a very popular design tool, which is also heavily used by industry. Therefore, it was chosen to base the CFD-CSD coupling on these two codes.

The differences within the reviewed approaches are mainly related to the chosen coupling schemes as well as the fluid and structural solvers. Although CFD has been used within all studies, the chosen turbulence model was different. It was found that both RANS with either $k-\omega$ SST or Spalart-Allmaras turbulence modelling as well as DES a combination of RANS and LES have been used. The actual tools for CFD were all different, showing that there is a good amount of CFD software available. In addition, different tools for meshing were used and several techniques for implementing dynamic mesh motions were employed.

The CFD tool within this research will be OpenFOAM, which is the main CFD tool at Fraunhofer Institute for Wind Energy and Energy System Technology (IWES) and ForWind. The chosen turbulence modelling approach will be RANS to resolve the blades (or a combined DES method) using $k-\omega$ SST, respectively Spalart-Allmaras modelling, as both were found to be a solid approach being heavily used in recent years.

The mesh will be generated similarly to the studies consisting of a background and refined pitch mesh around the blade. Moreover, dynamic mesh motion will be integrated by using the arbitrary mesh interface (AMI) technique as well as specific point translations and rotations accounting for the elasticity of the blade. Therefore, the available work of Rahimi et al. [34], Daniele [8] and Dose et al. [9] will be

modified for these tasks.

Concerning the structural solver, the most used modelling methods within the studies include FEM using beam elements for blade modelling. Also MBD and lumped parameter methods have been used. As for the CFD codes, the used CSD tools were different. The chosen structural solver is either ElastoDyn or BeamDyn as implemented in FAST. ElastoDyn utilizes an MBD approach except for the blades which are resolved based on Euler-Bernoulli beam theory. BeamDyn uses a combination of MBD with a refinement at the blades, which are modelled by an FEM method called GEBT, see [42].

In terms of coupling approaches mostly partitioned methods combining two different CFD and CSD tools were used within the reviewed studies. The actual coupling was established by preferably using a loose coupling method, due to reduced computational and programming effort. However, there were also methods using a tightly coupled approach. Within this research project a loose coupling partitioned method will be used, mainly due to the reason that a loose coupling method is easier to implement. This is especially the case if a neutral interface such as MpCCI can be utilized. The codes (FAST and OpenFOAM) remain relatively untouched allowing for possible updates. In addition, one has to admit that developing a tightly coupled method is not a simple task, which may not be appropriate for such a research project with limited time available.

Finally, the review on the chosen tools such as FAST, OpenFOAM and MpCCI showed that this choice of tools seems to be reasonable. This is due to the case that these tools function similar to tools, which already have shown a working coupled CFD-CSD method. Think for instance of the HAWC2CFD method which includes the coupling of HAWC2 and EllipSys3D. As such it was found that due to the new modularization within FAST and the great community tools within OpenFOAM the coupling can be achieved by using the well established neutral interface MpCCI.

3

Wind Turbine Simulation Cases

Two different turbines will be simulated. The first simulated turbine is the NREL phase VI turbine, which was used due to the reason that for this turbine a large set of measurement data is available. As such simulation results in yawed and pitched operational conditions could be validated against similar measurement sets. To address the effect of a fully functional controller and highly elastic blades the NREL 5MW turbine has been simulated at a second stage.

In this chapter first an overview of the specifications for the two different turbines to be simulated is given, see Section 3.1. Afterwards the simulations to be performed are discussed in Section 3.2. The simulations are based on currently existing methods such as FAST (BEM) and OpenFOAM (CFD), explained in Chapter 4, as well as the developed FAST-OpenFOAM coupled method, called fastFoam, see Chapter 5.

3.1. Turbine Specifications

The specifications of the two different turbines are presented within this section. First the smaller NREL phase VI is elaborated. Afterwards the NREL 5MW is discussed, which is closer to a state-of-the-art turbine nowadays.

3.1.1. NREL phase VI

The turbine which will be simulated first is the NREL phase VI turbine, see Figure 3.1. It was used for a large measurement campaign in the NASA Ames wind tunnel in the year 2001, see [30]. Due to the availability of the experimental data it has been heavily used in wind energy research for validation purposes.

Figure 3.1: The NREL phase VI turbine in the NASA Ames wind tunnel [30].

It is a two bladed stall regulated turbine. Its main specifications are given in table 3.1 and the distribution of chord and twist are shown in table 3.2.

Table 3.1: NREL phase VI turbine specifications [30].

Parameter	Value/Property
Number of blades	2
Rotor diameter	10.058 m
Hub diameter	1.016 m
Hub height	12.192 m
Rated power	19.8 kw
Rotor overhang	1.401 m
Tilt angle	0 deg
Cone angle	0 deg
Configuration	upwind
Power regulation	stall
Rotor speed	71.63 rpm
Blade tip pitch angle	3 deg

Table 3.2: NREL phase VI blade chord and twist distributions [30].

Radial distance r (m)	Span station (r/5.029)	Chord length (m)	Twist (deg)	Airfoil
0.508	0.100	0.218	0.000	Cylinder
1.510	0.300	0.711	14.292	NREL S809
2.343	0.466	0.627	4.715	NREL S809
3.185	0.633	0.542	1.115	NREL S809
4.023	0.800	0.457	−0.381	NREL S809
4.780	0.950	0.381	−1.469	NREL S809
5.029	1.000	0.355	−1.815	NREL S809

3.1.2. NREL 5MW

The second turbine to be simulated is the NREL 5MW turbine, see Figure 3.2.

Figure 3.2: The NREL 5MW turbine rotor geometry.

It was defined by Jonkman et al. in [22] and since then has widely been recognised as a reference turbine in the wind energy research community. It is a three-bladed multimegawatt upwind turbine with variable-speed and variable-pitch control. One drawback compared to the NREL phase VI turbine is that this turbine has only been defined conceptually, but was never manufactured, thus no experimental data exists.

Similarly, the specifications of the NREL 5MW turbine are given in table 3.3 and the corresponding chord and twist distributions are shown in table 3.4.

Table 3.3: NREL 5MW turbine specifications [22].

Parameter	**Value/Property**
Number of blades	3
Rotor diameter	126 m
Hub diameter	3 m
Hub height	90 m
Rated power	5 MW
Rotor overhang	5 m
Tilt angle	5 deg
Cone angle	2.5 deg
Configuration	upwind
Control	variable speed and collective pitch
Cut-in, rated, cut-out wind speed	3 m/s, 11.4 m/s, 25 m/s
Cut-in, rated rotor speed	6.9 rpm, 12.1 rpm
Rated tip speed	80 m/s

Table 3.4: NREL 5MW blade chord and twist distributions [22].

Radial distance r (m)	Span station (r/63)	Chord length (m)	Twist (deg)	Airfoil
2.867	0.046	3.542	13.308	Cylinder
5.600	0.089	3.854	13.308	Cylinder
8.333	0.132	4.167	13.308	Cylinder
11.750	0.187	4.557	13.308	DU40_A17
15.850	0.252	4.652	11.480	DU35_A17
19.950	0.317	4.458	10.162	DU35_A17
24.050	0.382	4.249	9.011	DU30_A17
28.150	0.447	4.007	7.795	DU25_A17
32.250	0.512	3.748	6.544	DU25_A17
36.350	0.577	3.502	5.361	DU21_A17
40.450	0.642	3.256	4.188	DU21_A17
44.550	0.707	3.010	3.125	NACA64_A17
48.650	0.772	2.764	2.319	NACA64_A17
52.750	0.837	2.518	1.526	NACA64_A17
56.167	0.892	2.313	0.863	NACA64_A17
58.900	0.935	2.086	0.370	NACA64_A17
61.633	0.978	1.419	0.106	NACA64_A17

3.2. Simulations

Both aforementioned Turbines will be simulated with three different methods. These are namely the CFD method based on OpenFOAM, such as described in Section 4.1 of Chapter 4 and the BEM-based FAST simulations, see Section 4.2. Finally, simulations were executed with the developed FAST-OpenFOAM coupled method, described in Chapter 5.

The simulations can be arranged into several categories, depending on the type of simulation if a controller and/or structural solver is included. These defined categories are shown in Table 3.5.

Table 3.5: Categories of test cases to be considered for comparison.

Category:	Aero	Aero-servo	Aero-elastic	Aero-servo-elastic
Torque control	No	Yes	No	Yes
Pitch control	No	Yes	No	Yes
Elasticity	No	No	Yes	Yes

Next, a simulation matrix was defined based on the two turbines and the three different methods to be investigated. The test matrix is shown in Table 3.6. The simulation cases for the NREL phase VI turbine (case 1 to 4) are matched to the experimental cases such as described in [30]. They follow exactly the same setup as used in the unsteady aerodynamics experiment.

The first case relates to the test sequence S0700000, which considers the rotor in normal power production at 0 deg yaw and a constant pitch of 4.815 deg. The second test case considers the experimental sequence S07YSU00, which relates to a 360 deg yaw sweep, but due to limits in computational time only 30 seconds are simulated. This equals a yaw sweep from 0 to 30 deg at a yaw rate of about 1 deg/s. Finally, the third case corresponds to the sequence R0600RD0 in the experiment. For this case the pitch angle is increased at a rate of about 0.18 deg/s. The starting value of pitch was taken as 4.815 deg again. In addition, in case 4 a partial power curve test case is obtained by different simulations at several wind speeds below the stall regime. This was done in order to investigate the convergence at different wind speeds.

The second stage (case 5 to 6) deals with simulations of the NREL 5MW turbine. Case 5 considers the normal power production. The initial yaw error for this case is zero. However, in the aero-servo-elastic simulations yawing due to the activated controller is allowed. In case 6 the yaw error is fixed to 30 deg and yawing motions of the rotor are not allowed. Both cases 5 and 6 are simulated with ElastoDyn as a structural solver for the blade.

The purpose of these different test cases was to be able to compare the three methods both in steady as well as unsteady operational conditions such as highly yawed inflow. This should then be

helpful to answer the research questions to be answered. The results for the different cases are shown in Chapter 6.

Table 3.6: Simulation matrix for the considered cases.

Case	Turbine	Method	Category	Description	Experiment	Structure
1 (a)	phase VI	OpenFOAM	Aero	Normal cond.	S0700000	Rigid
1 (b)	phase VI	FAST	Aero-elastic	Normal cond.	S0700000	ElastoDyn
1 (c)	phase VI	fastFoam	Aero-elastic	Normal cond.	S0700000	ElastoDyn
2 (a)	phase VI	OpenFOAM	Aero	Yaw sweep	S07YSU00	Rigid
2 (b)	phase VI	FAST	Aero-elastic	Yaw sweep	S07YSU00	ElastoDyn
2 (c)	phase VI	fastFoam	Aero-elastic	Yaw sweep	S07YSU00	ElastoDyn
3 (a)	phase VI	OpenFOAM	Aero	Pitch slope	R0600RD0	Rigid
3 (b)	phase VI	FAST	Aero-elastic	Pitch slope	R0600RD0	ElastoDyn
3 (c)	phase VI	fastFoam	Aero-elastic	Pitch slope	R0600RD0	ElastoDyn
4 (a)	phase VI	OpenFOAM	Aero	Power curve	Several	Rigid
4 (b)	phase VI	FAST	Aero-elastic	Power curve	Several	ElastoDyn
5 (a)	5MW	OpenFOAM	Aero	Normal cond.	None	Rigid
5 (b)	5MW	FAST	Aero-servo-elastic	Normal cond.	None	ElastoDyn
5 (c)	5MW	fastFoam	Aero-servo-elastic	Normal cond.	None	ElastoDyn
6 (a)	5MW	OpenFOAM	Aero	Fixed yaw error	None	None
6 (b)	5MW	FAST	Aero-servo-elastic	Fixed yaw error	None	ElastoDyn
6 (c)	5MW	fastFoam	Aero-servo-elastic	Fixed yaw error	None	ElastoDyn

4

Available Methods

In this chapter two of the currently available methods for wind turbine simulations are presented. First of all, in Section 4.1 the simulations based on OpenFOAM are described. OpenFOAM contains multiple high fidelity CFD solvers with RANS, LES or even DES possibilities. However, for wind turbine simulations purposes its standard integrated capabilities are limited as a fluid solver. In contrast, the CAE tool FAST from NREL is capable of executing aero-hydro-servo-elastic simulations. The tool is especially tailored for horizontal axis wind turbine simulations. Its aerodynamic simulations are based on a lower fidelity BEM model, which allows for a computational efficient approach. The wind turbine simulations using FAST are described in Section 4.2. Both tools are essential for the developed coupled method, which is later presented in Chapter 5.

4.1. OpenFoam Simulation Method

In the following section the method for simulating wind turbines with OpenFOAM is described. This includes the exact procedure which was carried out within this project concerning the simulations of the NREL phase VI and 5MW turbines. First of all the meshing strategy, see Section 4.1.1, is discussed. Afterwards, the approaches to account for motions (Section 4.1.2) are presented. Finally, the applied simulation setup, see Section 4.1.3, concerning the OpenFOAM simulations in Table 3.6 is presented.

4.1.1. Mesh Generation

Before solving the Navier-Stokes equations using OpenFOAM or another CFD software first a mesh or grid of often complex geometries such as wind turbines needs to be generated. The governing equations are then solved in discretized form on the mesh, where it is assumed that the resulting solutions are constant over one cell. A mesh within OpenFOAM can either be generated using built-in functions or using external software and importing the externally generated mesh. There are specific requirements on these meshes in order to successfully implement them.

These requirements are for instance the ability of representing motions applied on a geometry by dynamic mesh motion. Generally, it is aimed to generate a high quality mesh, which has a appropriate level of refinement in order to ensure that flow phenomena such as stall can be accurately reproduced. However, the mesh should not be too fine to limit the computational effort.

Therefore, the mesh for a wind turbine simulation using CFD has requirements on certain regions. For instance the region around the blades have to be represented by a very fine mesh to account for aerodynamic phenomena acting at the complex geometries. For instance, the effect of shed vorticity at the trailing edge and the strong tip and root vortices must be accounted for. In contrast to this highly refined regions, there are regions with a distance of several rotor diameter in the rotor plane, which account only for the surroundings and do not include the wake. These regions are generally of lower refinement as their distance to the rotor is large enough and therefore no complex aerodynamic phenomena are acting there. In addition, the wake region must be relatively refined especially in the near wake, whereas in the far-wake not a very high refinement is required as the inluence of the rotor is smaller.

Different types of meshes can be distinguished such as structured and unstructured mesh types. Whereas structured meshes consist of hexahedral cells, unstructured meshes are mainly composed of tetrahedrons. Structured meshes often result in high quality, but they are more difficult to generate especially around complex geometries. In contrast to that, unstructured meshes can be generated relatively fast. However, the resulting mesh quality may be not as good compared to the resulting quality of structured meshes.

4.1.1.1. Mesh Tools

Built-in utilities for mesh generation in OpenFOAM are blockMesh, explained in the following paragraph and snappyHexMesh, which is described later in this Section. Following the block decomposition approach blockMesh divides the domain into several regions of blocks. However, due to the block approach there exist problems when complex geometries have to be dealt with. Such geometries include the sharp trailing edges of airfoils for instance [34]. Therefore, a large amount of blocks is required to account for these complex geometries, which by manual set-up is a time-consuming exercise to achieve [34]. Also by using snappyHexMesh utility no fully automated procedure can be used [34]. Thus, generating a wind turbine mesh can be a complicated task.

BlockMesh The OpenFOAM utility blockMesh is utilized for generating meshes composed of blocks including curved edges and mesh grading [41]. Therefore, the mesh is generated from a `blockMeshDict` dictionary file located in the `constant/polyMesh` directory of the case by running the `blockMesh` command in the case directory. This `blockMeshDict` may be automatically generated by using the GNU preprocessor m4. It alows for changing the blockMesh parameters such as vertices, edges, blocks and boundaries more practically.

BlockMesh divides the domain for which a mesh shall be created into several blocks, which mostly have hexahedral shape. The hexahedral block shape is derived from the block definition which requires 8 vertices at each corner. However, also for instance tetrahedral block shapes may be obtained by overlapping of vertices pairs onto each other. In general hexahedral blocks are preferred over tetrahedral ones as they result in possibly high quality structured meshes and allow for mesh refinement via the OpenFOAM `refineMesh` command. The edges of these blocks can simply be straight or curved lines, but also splines, allowing for every possible curvature, may be used.

At a first step, vertices are defined in a three dimensional Cartesian coordinate system. The vertices are automatically labelled starting from 0 according to C++ convention [41]. For 2D simulations the third dimension, which always at least has the depth of one block, is ignored by setting proper boundary conditions.

Next, the hexahedral blocks can be defined by calling 8 vertex labels per block via the `hex` function. Attention needs to be taken on the way these labels are called. Within blockMesh the convention holds that looking from the inside of the block the vertices must be traversed in a clockwise manner along each face [41]. In addition, the number of cells in the direction of the three local coordinate axes, given by the right-handed local block coordinate system with its origin at the first vertex, must be specified. Also a grading for each direction can be applied if appropriate. A single block constructed from 8 vertices is shown in Figure 4.1 for clarification. The corresponding main parts of a `blockMeshDict` for the single block such as shown in Figure 4.1 is given in Listing A.1 in Appendix A.

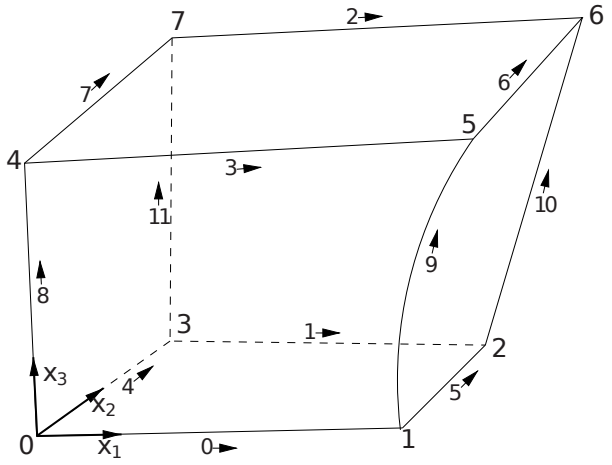


Figure 4.1: Example of a block generated with blockMesh [41].

Moreover, special edges with curvature can be specified. Two vertex points are automatically assumed to be connected by a straight edge. If this should not be the case the edges can be set as `arc`, where a single interpolation point is needed or as `spline`, `polyLine` or `BSpline` where multiple interpolation points are required [41].

Finally, mesh boundaries can be specified, which are given by patches composed of a number of block faces. Later specific boundary conditions on these created boundaries can be applied for the governing flow variables. A boundary is given a certain name for instance inlet and a certain type for example the more general patch or wall. Also cyclic boundary types, which are substantial for certain types of mesh motions are specified here. To specify a boundary via patches the block faces are called in a list, where each entry consists of 4 vertices corresponding to the face of one block [41].

For nearly all problems the generation of one block is not sufficient due to the complex geometries to be modelled. Therefore, several blocks have to be generated and need to be connected between each other. Within blockMesh there are two distinguished approaches to connect blocks. First of all, face matching can be applied, meaning that neighbouring blocks share the same faces by calling the same vertices. By ignoring these faces from the patch entries, blockMesh automatically identifies them as internal faces between the neighbouring blocks [41].

In addition, face merging might be used. In contrast to face matching within face merging different sets of vertices are used. These are then added to the list of patches with a separate patch entry for each block. Finally, the function `mergePatchPairs` is called to merge both patches. This procedure follows a technique, where one patch the so called master patch remains as defined including its vertices, whereas the faces of the slave patch are projected onto the master patch [41]. As such an internal face is constructed. This allows for simple merging of patches, but a disadvantage is that at the overlap some polyhedral cells may be generated.

BladeBlockMesher An example of an automated mesh generation tool for wind turbine blades utilizing blockMesh is provided by Rahimi et al. in [34]. It models the mesh around a wind turbine blade based on the geometrical shape of airfoils at several blade sections. It is claimed to have a complete automation and fast mesh generation resulting in high quality meshes. Therefore, it uses adjustments for the dimensionless wall distance based on the Reynolds number of each blade section.

In addition to the airfoil geometry, the spanwise position, section chord length and twist angle as well as the alignment point of each section to the pitch axis must be delivered by the user [34].

The tool then generates a two dimensional O-mesh at each blade section based on the provided geometrical data as well as configuration parameters related to the refinement of the mesh for instance [34]. This generation utilizes a hyperbolic partial differential equation solver. First, the airfoil surface is generated by a discrete number of points and by subsequently marching outwards the O-mesh is formed.

As a next step, the generated two dimensional O-mesh at each sections are interpolated in between each other using a special procedure, in order to generate a three dimensional mesh around the blade, see Figure 4.2. Therefore, a finite difference approach for arbitrarily spaced meshes according to

Fornberg is used [34]. By generation of splines between different blade sections the blade shape is fitted and smoothed. Afterwards, it is corrected for cell non-orthogonality by checking the perpendicularity of the cells close to the blade surface.

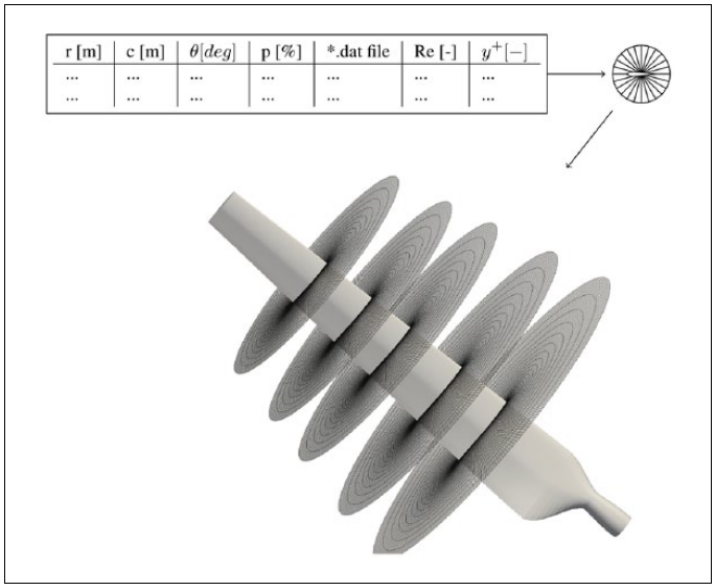


Figure 4.2: The process used within BladeBlockMesher generating a blade mesh from two dimensional O-meshes [34].

Attention needs to be taken on the blade tip, which is smaller in geometry and thus may result in larger cell density [34]. There are two methods which can be utilized to solve this problem. Either a flat tip closure or a rounded tip closure can be applied by the user.

Finally, the generated three dimensional mesh is obtained as a cylinder with a flat or spherical top surface including the blade [34]. For the background mesh consisting of the domain around the wind turbine and structure such as hub or nacelle, an external script must be used. The resulting mesh around the blade, which is in the form of a blockMesh dictionary, can then be assembled to a rotor by accounting for the number of blades.

Using the tool a mesh for the NREL phase VI turbine was generated [34]. Next, validation and verification was executed by a mesh independence test and comparing CFD results based on OpenFOAM and the generated mesh to measurements and other CFD results. The comparison showed good agreement and as a result it is assumed to significantly reduce the time for mesh generation.

SnappyHexMesh The second built-in mesh generation utility within OpenFOAM is snappyHexMesh. It constructs three dimensional hexahedral or split-hexahedral cells from triangulated surface geometries, which must be delivered by the user in specific file formats. These file formats include Stereolithography (STL) or Wavefront Object (OBJ), which can for instance be exported from computer-aided design (CAD) software [41]. In addition to the required tri-surface files a starting mesh or background mesh needs to be given. It can be generated using blockMesh and must only consist of hexahedral cells, thus being structured. Moreover, a `snappyHexMeshDict` file is required in the `system` folder of the case. Within this dictionary the meshing process of snappyHexMesh is controlled.

Finally, the actual meshing process of SnappyHexMesh can be executed. It uses a cell splitting technique on the background mesh at locations specifically near the tri-surface geometry. Next, the corresponding cells in the background mesh are replaced by the tri-surface geometry and thus removed. A certain level of refinement at required locations can be set by the user in order to ensure that these processes are accurate. Afterwards, vertices which are close to the tri-surface geometry are snapped onto it to ensure that the surface of the geometry is actually accurately matched.

However, as there might be some irregularities near the surface boundary an additional step may be conducted depending on the complexity of the surface. This step includes adding of additional mesh layers after the surface snapping, which may be appropriate if the initial number of layers is not

sufficient to actually represent the surface geometry. An example of the entire procedure is given in Figure A.1 in the Appendix A.

4.1.1.2. Mesh for NREL phase VI

First of all, a blade mesh hereafter called pitch mesh for the NREL phase VI turbine was generated. The name pitch mesh is due to the mesh symmetry allowing it to rotate about the blade pitch axis. Therefore, the aforementioned utility BladeBlockMesher as described in [34] has been utilized. It generates a `blockMeshDict` which can then be processed. For the generation of the pitch mesh different discretizations were chosen. The chordwise amount of cells and the number of cells normal to the airfoil were varied, while the spanwise cell number was kept constant.

Table 4.1: Different generated pitch and final meshes for the NREL phase VI turbine.

Mesh:	**Baseline**	**Coarse**	**Fine**
Chordwise cells	275	250	300
Cells normal to Airfoil	40	30	50
Spanwise cells	90	90	90
Cells for pitch mesh	1,331,840	854,280	1,904,800
Total cells	9,563,440	6,719,220	13,025,796

The blade was meshed in BladeBlockMesher with a rounded tip, although in the NREL phase VI experiments partly a flat tip for the blade was used, see Hand et al. in [30]. The choice of a rounded tip is justified as the generated mesh by BladeBlockMesher is of much higher quality with a rounded tip. For the flat tip problems, especially non-orthogonalities, will occur at the tip edges due to the sudden jump in the geometry. The generated pitch mesh is shown in figure 4.3.

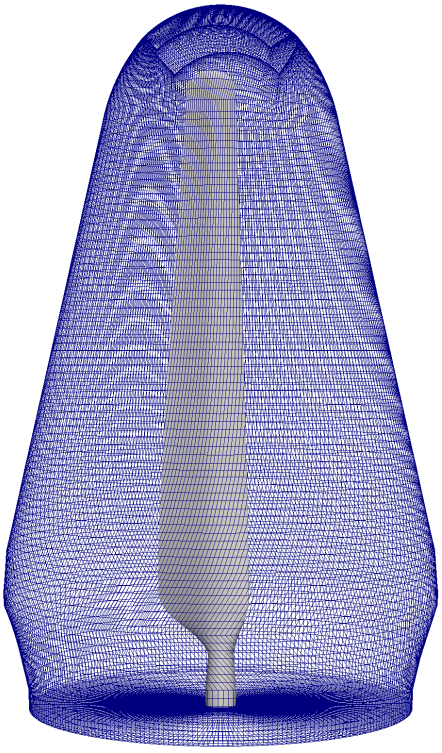


Figure 4.3: The generated pitch mesh with the blade in its centre.

The mesh around the rounded tip can be seen in Figure 4.4.

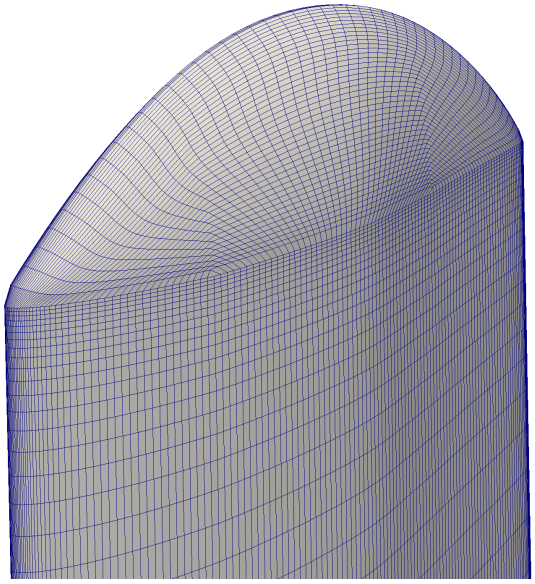


Figure 4.4: The rounded tip blade mesh.

After the pitch mesh has been generated the surrounding needs to be taken into account representing the wind tunnel used within the NREL phase VI experiment. Therefore, an outer mesh hereafter called background mesh has been generated using `blockMesh`. A cyclindrical domain was meshed with approximately the size of the NASA Ames wind tunnel for its cross-section. The domain dimensions can be seen in Figure 4.5, where one diameter corresponds to 5.029 m. The actual wind tunnel cross-section is rectangular with a size of about 24.4 times 36.6 m, see [32].

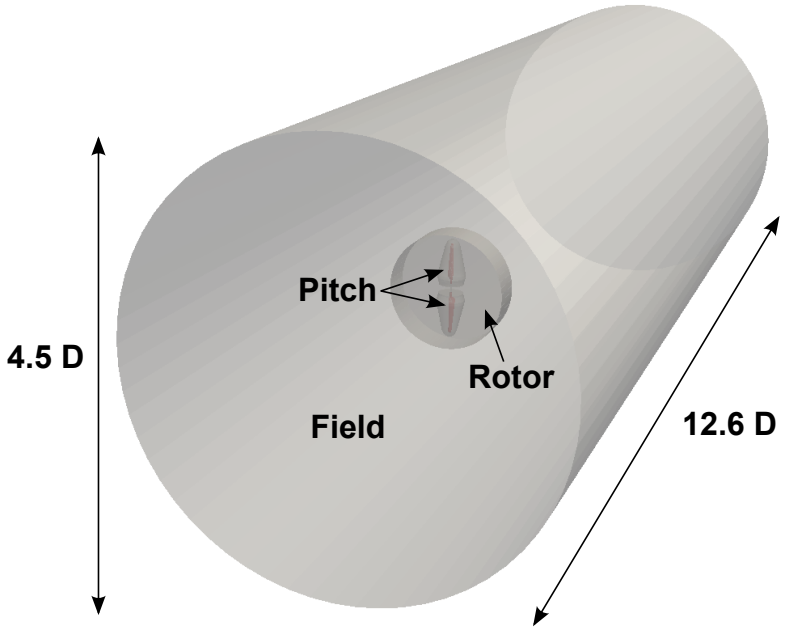


Figure 4.5: Dimensions and structure of the NREL phase VI mesh.

The background mesh consists of the farfield, which does not require special refinement, as well as the rotor disk, which later incorporates the pitch meshes accounting for the two blades. While the pitch motion is achieved through pitch rotations of the pitch meshes, the actual rotation of the rotor is obtained through rotation of the rotor disc. During the meshing process only half of the background mesh needs to be generated as the other half can easily be obtained by symmetry using OpenFOAM's `mirrorMesh` utility.

Therefore, only half of the rotor disc, such as shown in Figure 4.5, was generated using `blockMesh` and was merged with half of the cylindrical field mesh. The cylindrical field mesh includes a cut-out for half the rotor disc. One half of the rotor disc such as generated for the background mesh is shown in Figure 4.5. The cut-out which later fits the pitch mesh is clearly visible. In addition, a cut-out for a cylindrical hub can be seen.

As a next step the pitch mesh is merged with the entire background mesh using OpenFOAM's `mergeMeshes` tool. Following this procedure the entire mesh is generated. Again for the background mesh the refinement was varied slightly, especially in the rotor disc and wake regions, in order to obtain the three meshes with different number of cells in Table 4.1.

It has to be noted that the tower nor the nacelle of the turbine are represented in the mesh as only the rotor is meshed. This neglection simplifies the meshing procedure by a great amount, but also may lead to neglections of effects such as tower shadow for instance. The significance of neglecting tower (and nacelle) will later be adressed within the results Chapter see 6.

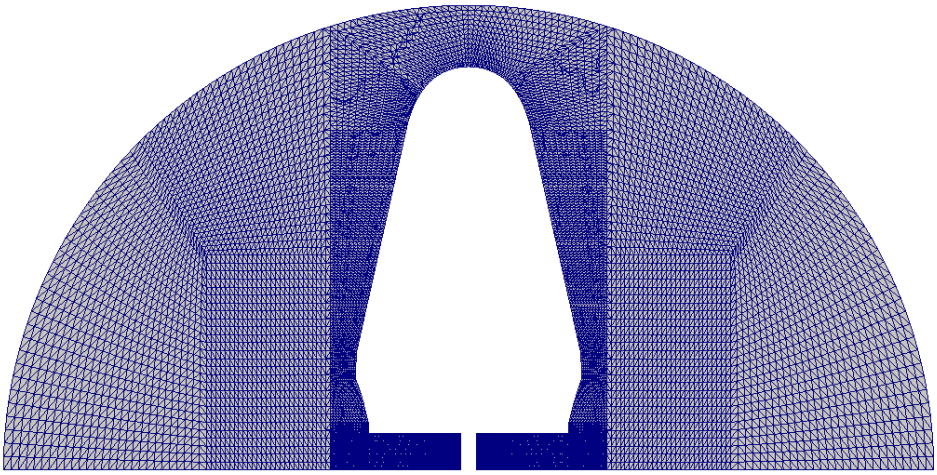


Figure 4.6: One half of the rotor disc without the pitch mesh.

To show the quality of the mesh a view at a blade section is of interest. Therefore, a slice view is given in Figure 4.7 at a middle section of the blade. In addition the mesh at the leading-edge, respectively trailing-edge, is shown in Figures 4.8 and 4.9.

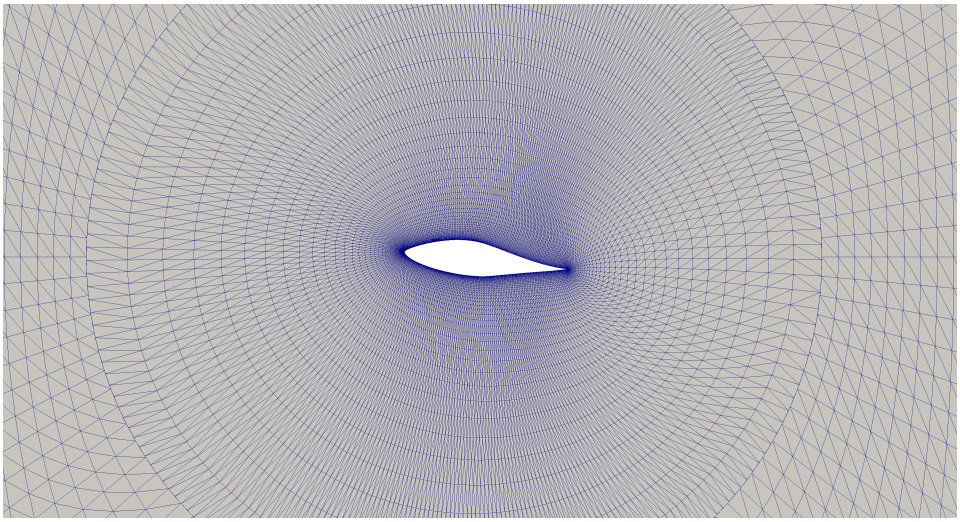


Figure 4.7: Slice view of the blade at 0.67R.

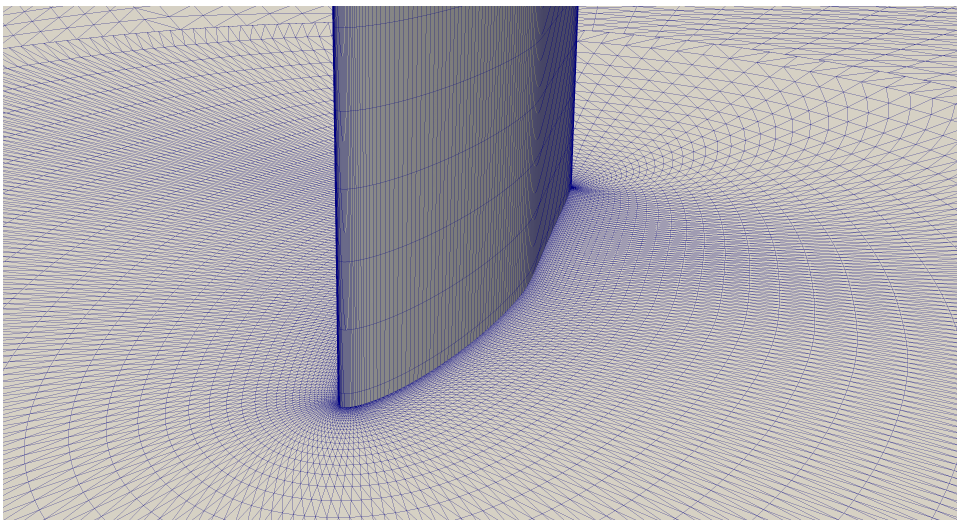


Figure 4.8: The mesh at the leading-edge at 0.67R.

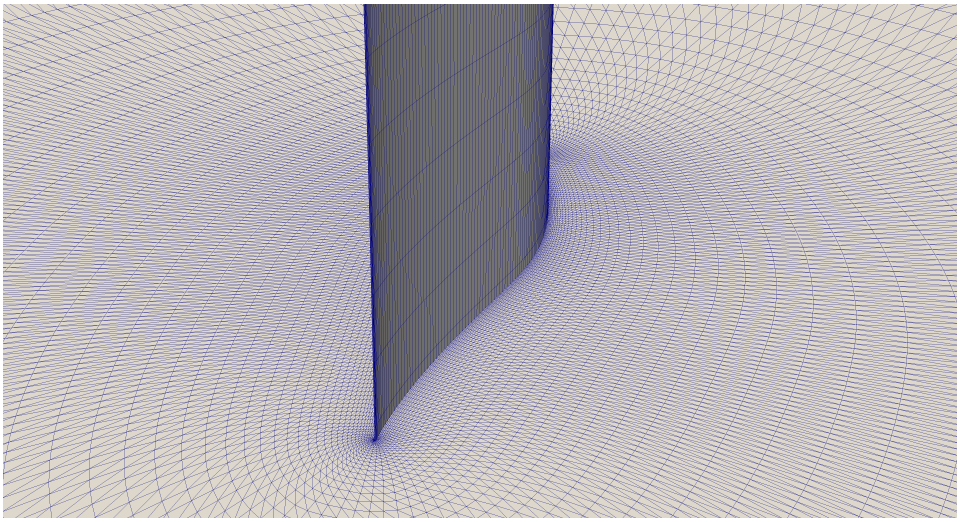


Figure 4.9: The mesh at the trailing-edge at 0.67R.

In addition, a side-view of a vertical cut is given in Figure 4.10 showing the different refinement zones of the Mesh. The different levels of refinement in the far field have been obtained using OpenFOAM's `refineMesh` utility.

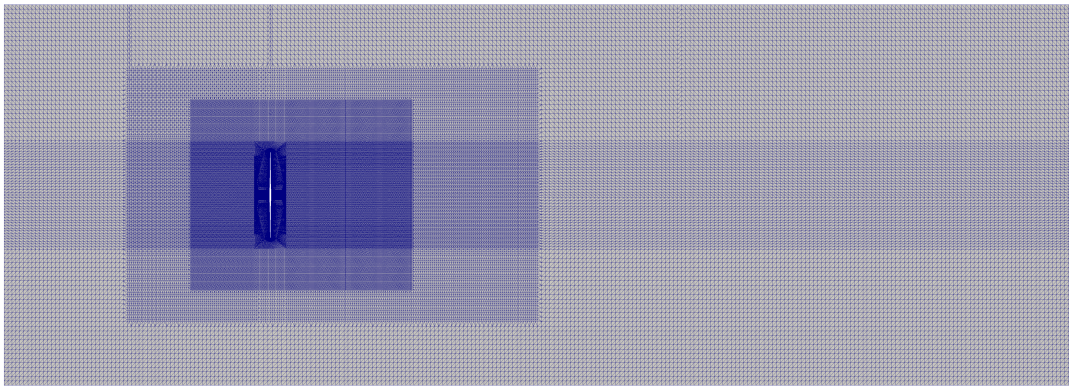


Figure 4.10: A vertical cut through the entire NREL phase VI mesh.

4.1.1.3. Mesh for NREL 5MW

The mesh for the NREL 5MW turbine was provided by IWES. It was already used for previous projects and thus could be reused within this project. Thereby no additional time needs to be spent to generate a mesh for the NREL 5MW turbine.

The pitch mesh for this turbine was generated again by BladeBlockMesher, see [34]. The structured mesh consists of 4.5 million cells. The number of cells in chordwise direction is 276. The three pitch meshes, accounting for the three blades, were merged into a background mesh. The merging was done using the snappyHexMesh utility of OpenFOAM, as described in Section 4.1.1.1. The final mesh has a size of 27 million cells. This approach allows for an automatic procedure, which is faster compared to the approach used for the NREL phase VI. This is due to the reason that the background mesh does not require specific cutouts for the pitch meshes, as this is accounted for with snappyHexMesh. However, a drawback of the automized approach by snappyHexMesh is that polyhedral cells are generated and therefore the final mesh is partly of structured type.

Another difference is that due to this meshing approach no specific rotor disc region is generated. The rotation of the rotor is then accounted for by a rotation of the entire mesh. Similar to the phase VI turbine no nacelle or tower is included as this significantly simplifies the meshing procedure. The final mesh is tilted by 5 degree to account for the tilt angle of the NREL 5MW turbine. This is not shown in

the Figures.

It has to be noted that for the NREL 5MW mesh no mesh convergence study has been done within this project, as the mesh was a result of previous projects where such a study was achieved.

The mesh dimensions and its structure are shown in Figure 4.11.

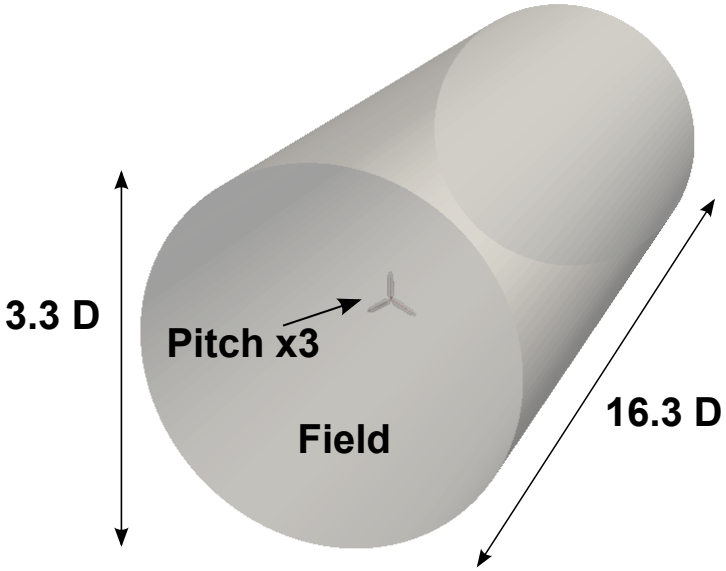


Figure 4.11: Dimensions and structure of the NREL 5MW mesh.

A front-view of the rotor mesh region is shown in Figure 4.12.

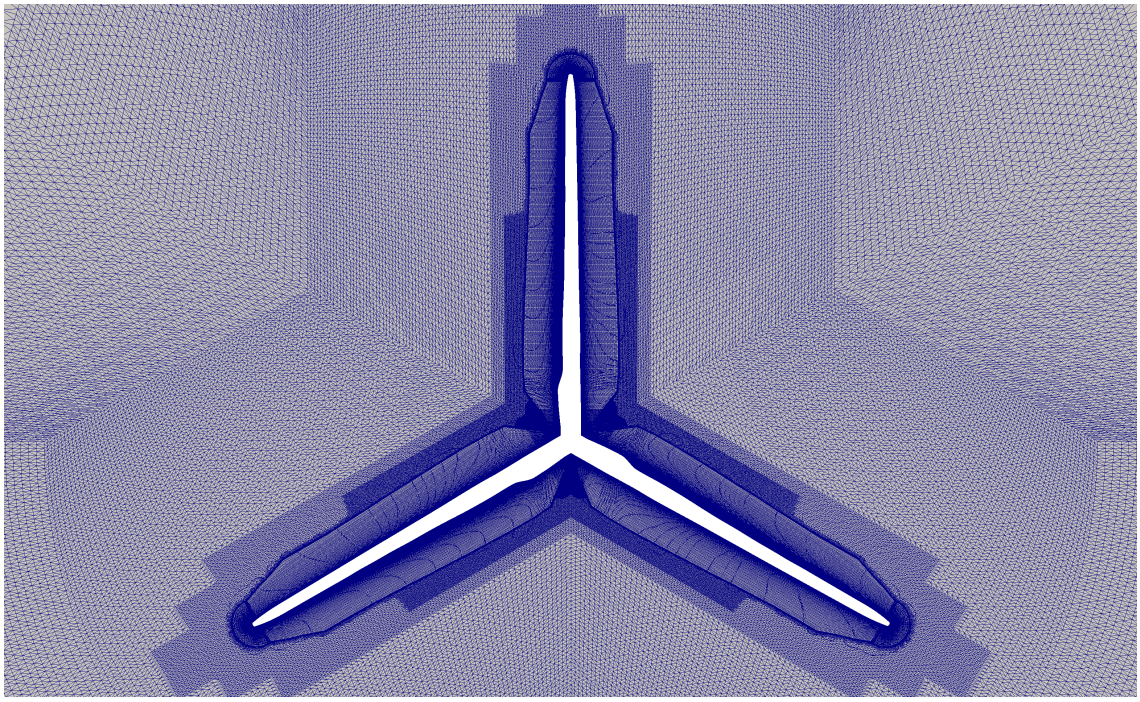


Figure 4.12: Front-view of the rotor of the NREL 5MW mesh.

The side-view of a vertical cut through the entire mesh is shown in Figure 4.13.

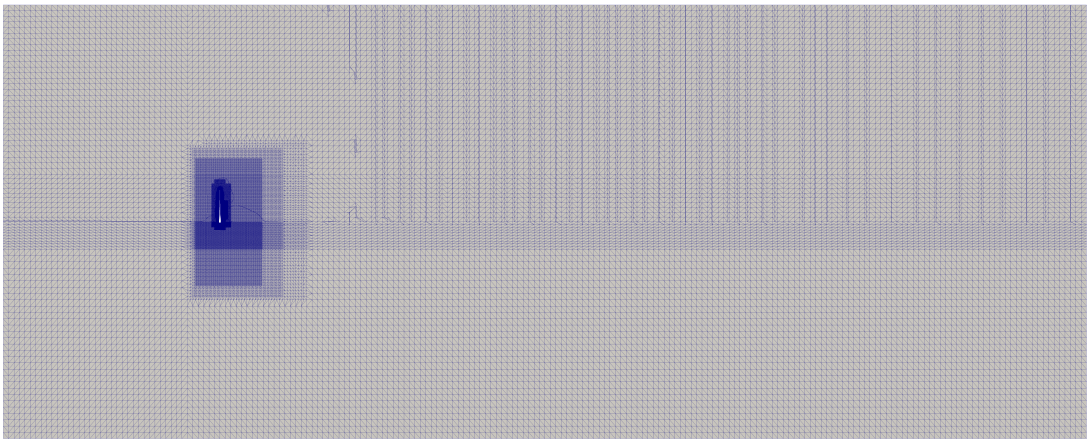


Figure 4.13: A vertical cut through the entire NREL 5MW mesh.

4.1.1.4. Mesh Quality

To distinguish a high quality mesh from a low quality one several mesh quality indicators are used within OpenFOAM. These quality indicators refer to the quality of the generated cells and the entire mesh.

For a quick check of the mesh quality the OpenFOAM command `checkMesh` can be used on a generated mesh. This command checks the validity of the mesh and reports possible errors or indicates if attention needs to be taken on possible badly generated cells.

The boundary and maximum cell openness as reported by `checkMesh` should be of very low value approaching zero, thus indicating that no open boundaries or cells are included in the mesh. Thus a closed domain will be simulated. Moreover, the cell aspect ratio describing the ratio of longest to shortest edge is reported, for an example see Figure 4.14 with the ratio of edge A to B. This value should generally be limited to the order of hundreds if possible. The ideal value would be one, but this cannot be satisfied for difficult geometries for example a sharp trailing edge. In addition, `checkMesh` investigates possible errors with the face area magnitudes and volumes, for instance it reports an error if negative volumes are detected.

An important measure of good mesh quality is the mesh non-orthogonality. It is defined by the angle between a line connecting the cell centres (P and Q) of neighbouring cells and the normal of their common face (here $\hat{n}$). This angle is illustrated in Figure 4.14 for a small non-orthogonality (about 20 deg). The optimal value would be an angle of 0 deg. Critical values which are reported by `checkMesh` as severely non-orthogonal lie above 70 deg.

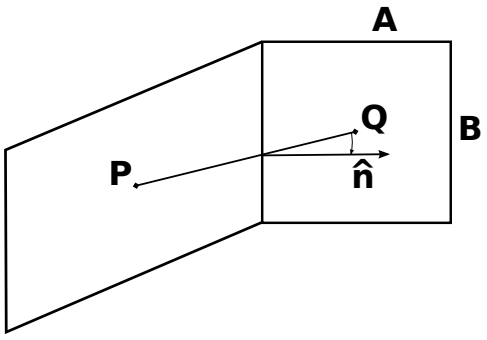


Figure 4.14: The mesh non-orthongality criterion.

Finally, `checkMesh` also reports the maximum skewness in the mesh. The skewness is derived from the distance (e) between the intersection of a line (d) connecting two cell centres (P and Q) with the common face and the centre (C) of their common face as shown in Figure 4.15. It is defined as the ratio between e and d. Generally, it should be as low as possible, but `checkMesh` will complain about highly skewed faces if a threshold of 4 is exceeded.

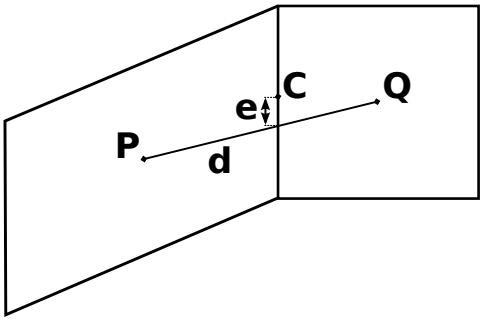


Figure 4.15: The skewness between two cells.

The resulting overall mesh generated for the NREL phase VI turbine passes the `checkMesh` command. The maximum skewness and aspect ratio are 1.7 and 61. The maximum non-orthogonality is 71.7 deg wile 22 faces show a non-orthogonality higher than 70. However, this is still considered acceptable as only a small amount of faces have a severe non-orthogonality, which is not extremely high in magnitude.

Similarly, the NREL 5MW mesh is expected to be satisfactory concerning the mesh quality as documented by `checkMesh`. Its maximum skewness is 3.6 and the maximum aspect ratio is 94. The critical mesh-non-orthogonality criterion of 70 deg is exceeded by about 1700 faces with a maximum of 76.1 deg. Previous projects have shown that the mesh is still acceptable.

A mesh quality comparison between both meshes is given in Table 4.2.

Table 4.2: Mesh quality for the different meshes.

Parameter	NREL phase VI Mesh	NREL 5MW Mesh
Type	Mostly structured	Mostly structured
Cells	9,563,440	27,697,518
Percentage of hexahedral cells	99.5	97.5
Max. skewness	1.7	3.6
Max. aspect ratio	61	94
Max. mesh non-orthogonality	71.7 deg	76.1 deg
Severely non-orthogonal faces	22	1697
Min. cell volume	2.73e-10 m^3	1.14e-08 m^3

4.1.2. Motion in OpenFOAM

There are several possibilities to account for the movement of structures in OpenFOAM. Two main options exist within OpenFOAM to simulate rotating structures like the wind turbine rotor. First of all, the multiple reference frame (MRF) method can be used accounting for steady rotations, see Section 4.1.2.1. As an alternative option the user may choose the dynamic mesh approach, which is suitable for transient simulations as well. This approach is described in Section 4.1.2.2. The chosen approach especially tailored for wind turbines is then outlined in Section 4.1.2.3.

4.1.2.1. Multiple Reference Frame Rotation

The MRF model accounts for both stationary and rotating frame of references. The mesh is kept constant, but for the defined rotating zone a solid-body rotation velocity is directly included into the solution of the Navier-Stokes equations [33]. The Navier-Stokes equations are thus solved differently for the rotating and stationary zone. In particular, in the equations for the rotating zone the coriolis and centrifugal forces are included [33].

The MRF method can only model steady rotations. For a wind turbine this modelling technique is similar to the Frozen-Rotor concept. The resulting flow is of steady nature and does not include any transient phenomena. This leads to certain disadvantages as for instance interactions between blade and tower cannot be modelled as well as other unsteady effects.

To apply the MRF method the user needs to supply an `MRFProperties` file located in the `constant` folder. This file is shown for the case 5 (c) in Listing 4.1. The rotating cell zone can be generated using the OpenFOAM command `topoSet`. The cell zone can be described as being defined by certain

number of cells within the mesh. The rotation is specified by the origin of the rotation, the axis around which it is rotated and the rotational speed. For the wind turbine simulations a cylindrical region around the rotor is selected as the rotating cell zone. The patches or boundaries, which are non-rotating are specified as well.

For the simulations with MRF steady solvers such as `simpleFoam`, based on the SIMPLE algorithm see Section 4.1.2.3, can be used. Due to the involved simplifications, these computations are often first executed to get initial results. Then at the next stage, transient simulations with more advanced mesh motion techniques including dynamic mesh rotations, for instance using the AMI approach, can be executed. Therefore, the previously obtained results from the steady solvers can be used as initial flow field, thereby improving the convergence rate of the transient solvers.

Listing 4.1: Example of the MRFProperties file used for case 5 (a).

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      MRFProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

MRF1
{
    cellZone        ROTOR;
    active          yes;

    nonRotatingPatches (PITCH_AMI_OUT_0 PITCH_AMI_IN_0 PITCH_AMI_OUT_1
                        PITCH_AMI_IN_1 PITCH_AMI_OUT_2 PITCH_AMI_IN_2 TOP INLET OUTLET);
    origin      (0 0 0);
    axis        (0.9961946980917457 0 -0.08715574274765818);
    omega       1.2671090355; // rad/s
}

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

4.1.2.2. Dynamic Mesh Rotation

Dynamic mesh rotation in OpenFOAM is achieved by moving mesh regions which shall rotate. Thus at every time a new state of the mesh is obtained and written to the respective time folder. However, this procedure is computationally far more expensive compared to MRF as every cell respectively point needs to be updated.

At first, the rotating region needs to be set up specifically. Its interface must be declared as mesh interface. To declare it as such, the two options cyclic AMI or GGI can be utilized.

The cyclic AMI can be declared easily as only the `constant/polyMesh/boundary` file and the files in the `0` folder need to be adjusted. More specifically, the patch type needs to be set to `cyclicAMI`.

In contrast, for the GGI approach also `faceZones` need to be specified at the interface. Also both type of interfaces AMI and GGI may allow for the same motions, although there optimal applications may be slightly different. AMI is known to be stable also if multiple regions are rotating even in an overlapping manner. This occurs for instance for a wind turbine, where the blades are rotating about their pitch axis, while the rotor rotates about the downstream axis. However, the GGI approach seems to be faster than AMI, see [3].

After having specified the interface as AMI or GGI, the rotations are set-up in the `dynamicMeshDict` file in the `constant` folder. This is done by selecting the region (`cellZone`), which shall rotate, as well as the origin of the rotation and the angular velocity.

In contrast to MRF, special solvers such as `pimpleDyMFoam` are required, which besides solving the fluid equations also update the mesh at runtime.

4.1.2.3. Wind Turbine Dynamic Mesh Motion

An example for an implementation of dynamic mesh motion in OpenFOAM due to pitch and torque is given by the work of Daniele, see [8]. The author implemented a hierarchical mesh motion accounting first for the rotor rotation and secondly for the blade pitch rate [8]. A torque and pitch controller has been implemented for this [8]. The user therefore needs to supply the torque-speed curve or for the pitch control a proportional integral control logic, respectively a look-up table [8].

The implementation makes use of the cell zones in OpenFOAM as well as the AMI rotation concept. The pitch control zone mainly consists of the pitch mesh, thus a highly refined zone around the blade surface [8]. In addition, the torque control zone is the zone spanning the rotor plane with some dimension in downstream respectively upstream [8]. This zone includes the pitch zone, as the entire rotor including blades needs to be rotated by the torque [8]. Both pitch and torque control zone are created as cyclic AMI to account for the mesh rotations.

Following this procedure, the implementation of both pitch and torque control is achieved and resulting in mesh motions [8]. However, the author states that the control strategy includes approximations which may be further mitigated by utilizing control implementations integrated in wind turbine design codes such as FAST [8]. The motion due to elasticity was not accounted for [8].

The work of Daniele will be used as a basis for the implementation of the dynamic mesh motion due to the rigid body motions related to yaw, torque and pitch control. Therefore, the mesh motion implementation has been extended by the dynamic mesh motion due to yaw, which is applied first in the hierarchy of motions now. The mesh motions are followed by a control action giving the angles or angular rates of these motions. For simulations only based on OpenFOAM this action is specified as a look-up table in the `dynamicMeshDict` file. An example of an excerpt of such a `dynamicMeshDict` file is given in Appendix B, see Listing B.1. For an overview on the utilized coordinate systems see Figure C.1 in Appendix C. For a coupled approach with FAST these actions (angles) are directly communicated from FAST, for details see Chapter 5.

For the generated mesh of the NREL phase VI turbine the control zones are shown in Figure 4.5. It has to be noticed that while the torque and pitch control zones are defined as cyclic AMI, for the farfield zone this is not required. This is due to the reason that the farfield zone accounts for the yaw motion, which can simply be achieved by a rotation of the entire mesh around the vertical axis. For the NREL 5MW mesh the yaw and torque control zones are similar, as the yaw and torque motions are achieved by full mesh rotations. The remaining pitch control zones for the three blades are related to cyclic pitch AMI rotations. The control zones for the NREL 5MW are shown in Figure 4.11.

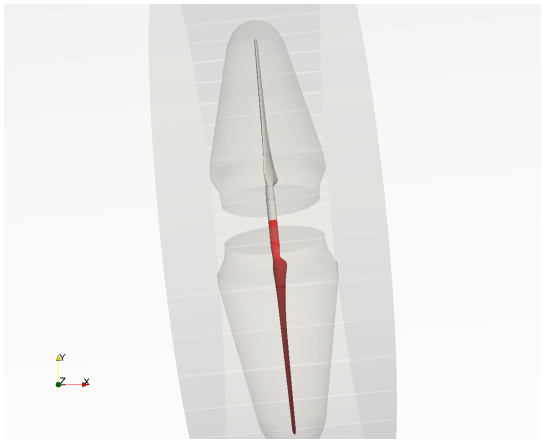
An overview of the mesh motions that are used for the simulations using OpenFOAM is given in Table 4.3. These have all been achieved by the implemented dynamic mesh motion strategy using, a `dynamicMeshDict` file similar to the one given in the Appendix B, see B.1.

Table 4.3: Mesh motions for the different cases to be simulated.

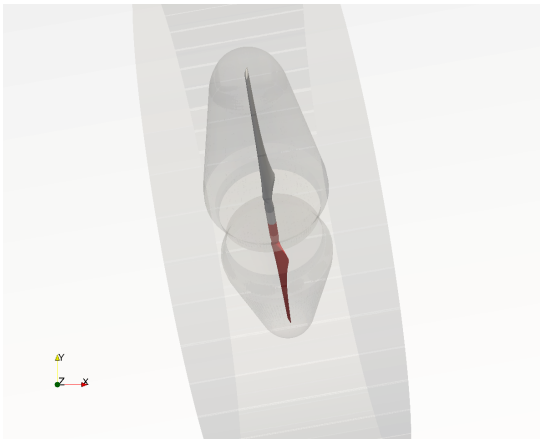
Case	Yaw	Torque	Pitch
1 (a)	0 deg	431.200 deg/s	4.815 deg
2 (a)	1 deg/s	431.137 deg/s	4.815 deg
3 (a)	0 deg	430.383 deg/s	+0.180 deg/s
5 (a)	0 deg	72.600 deg/s	0 deg
6 (a)	30 deg	72.600 deg/s	0 deg

In Figure 4.16 it is shown how the rigid body motions due to yaw, torque and pitch rotations are applied to the NREL phase VI turbine. Therefore, the mesh rotations are shown from above the rotor disc. Notice that the small coordinate system in Figure 4.16 indicates the wind direction, which is to the right and equals the X-vector. The corresponding yaw, azimuth and pitch angles at the different time steps related to these mesh states are given in Figure B.1 in Appendix B.

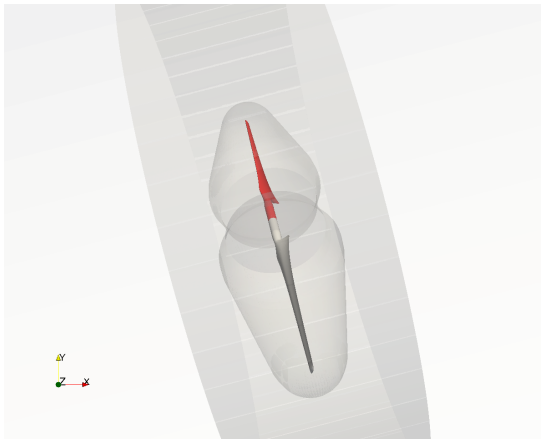
(a) Time = 0 s.



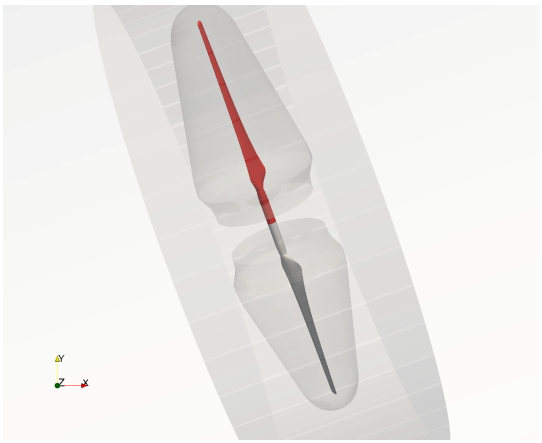
(b) Time = 1 s.



(c) Time = 2 s.



(d) Time = 3 s.



(e) Time = 4 s.

(f) Time = 5 s.

Figure 4.16: The NREL phase VI turbine mesh undergoing yaw, torque and pitch dynamic mesh motions (view from above).

4.1.3. Simulation Setup

In this section the setup of the OpenFOAM simulations is described. First, the initial conditions that were used are presented in Section 4.1.3.1. Afterwards, the solution method including the used solvers etc. is elaborated in Section 4.1.3.2.

4.1.3.1. Initial Conditions

The mesh has been generated with different patches. The type of these boundary patches has been set in the `boundary` file in the `constant/polyMesh` folder. It can be summarized such as seen in Table 4.4.

Table 4.4: Type of boundary patches.

Patches	Type
Inlet, Outlet, Sides	patch
Blades, Hub	wall
Rotor zone, Pitch zones	cyclicAMI

The boundary conditions applied to these patches are set in the `0` folder in the `U`, `p`, `k` and `omega` files for a $k-\omega$ SST turbulence model such as used for the NREL phase VI. For a Spalart-Allmaras model, which will be used for the NREL 5MW, the turbulence model variables are different, which will be further described in Section 4.1.3.2. The selected boundary conditions for the $k-\omega$ SST model are shown in Table 4.5.

Table 4.5: Boundary conditions on boundary patches for the $k-\omega$ SST turbulence model.

Patches	Condition
Inlet	fixed value (inletOutlet) for U, k and ω, zero gradient (outletInlet) for p
Outlet	fixed value (outletInlet) for p, zero gradient (inletOutlet) for U, k and ω
Sides	slip condition for phase VI except for yaw case and 5MW (inletOutlet)
Blades, Hub	no slip condition for U, wall functions for k and ω, zero gradient for p
Rotor zone, Pitch zones	cyclicAMI

Notice that for cases which consider yawing of the turbine (case 2 (a)) or tilting (5MW cases 5 and 6 (a)), the boundary conditions on the (tunnel) sides as well as on the inlet and outlet are changed to `inletOutlet` for U, k and ω while for p `outletInlet` is taken. These boundary conditions are specific conditions in OpenFOAM, which take a `zeroGradient` or `fixedValue` condition depending on the direction vector of the quantity.

This is important for the phase VI yaw cases as yawing is applied by rotating the entire mesh around the vertical axis, while the direction of inflow stays constant. Thus, the tunnel sides would rotate as well, which is not the case for the experiment where only the rotor yaws. Therefore, the tunnel sides in the simulation set-up must be permeable now. As such the tunnel wall effect can no longer be accounted for, however the effect of yawing may then be physically considered. As the tunnel wall effect is now neglected, the cylindrical domain has been increased by about 2.4 times in diameter to better account for the effect of the skewed wake.

Similarly, the same boundary conditions are used for the NREL 5MW as a tilt angle is included by tilting the entire mesh by 5 deg. However, as the direction of inlet velocity stays the same while tilting, the velocity vector is now no longer parallel to the sides. Therefore, `inletOutlet`, respectively `outletInlet` boundary conditions have to be applied.

The initial values for the boundary conditions as given in Table 4.5 can be calculated from the experimental data for the NREL phase VI turbine (case 1-4). The value for the velocity was taken as the mean value of the tunnel inlet velocity for the time to be simulated (about 30 s). Similarly, the values for the rotational speed and air density are taken. Moreover, the turbulence kinetic energy k was calculated from Equation 4.1.

$$k = \frac{3}{2}(UI)^2 \tag{4.1}$$

Where U is the velocity, relating to the tunnel velocity for the phase VI cases, whereas for the 5MW the mean wind velocity is taken. I is the turbulence intensity. For the turbulence intensity a constant value of 1 percent was chosen. Having obtained the value for k, the specific dissipation rate ω from kinetic to internal thermal energy can be approximated using Equation 4.2.

$$\omega = \frac{\sqrt{k}}{l} \tag{4.2}$$

Where l is the turbulent length scale describing the size of the largest energy-containing eddies. It can be estimated as approximately the size of the rotor diameter. Thus, for l a value of about 10 m, respectively 126 m, is chosen representing the diameter of the NREL phase VI and 5MW turbines.

In addition, also the dynamic viscosity μ needs to be computed. It has to be given in the `transportProperties` file within the `constant` folder. It was approximated for every case using Sutherland's law as given in Equation 4.3.

$$\mu = \mu_{ref} \frac{T_{ref} + C}{T + C} \left(\frac{T}{T_{ref}} \right)^{3/2} \tag{4.3}$$

Where μ_{ref} is the viscosity at the reference temperature T_{ref}. The viscosity μ_{ref} was taken to be 18.27e-06 Pa s and T_{ref} 291.15 K. C is Sutherland's constant which is 120 K. Then the actual dynamic viscosity μ can be calculated using the temperature data from the experiment for the phase VI cases. For the 5MW a common value was chosen for μ_{ref}.

The resulting values for all the variables were calculated for the different cases and are shown in Table 4.6. The values for the power curve cases (case 4 (a)) are not shown here, but the calculations were similar and matched the experimental data.

Table 4.6: Initial conditions for the different cases to be simulated.

Case:	**1 (a)**	**2 (a)**	**3 (a)**	**5-6 (a)**
Experiment	S0700000	S07YSU00	R0600RD0	None
Wind speed U (m/s)	7.016258	6.986503	6.016105	11.400000
Rotational speed Ω (rpm)	71.866689	71.856115	71.730473	12.100000
Turbulence kinetic energy k (m^2/s^2)	0.007384	0.007322	0.005429	0.020000
Specific dissipation rate ω (1/s)	0.008544	0.008507	0.007326	0.001800
Air density ρ (kg/m^3)	1.245786	1.247981	1.242534	1.225000
Dynamic viscosity μ (kg/ms)	1.792681e-5	1.790240e-5	1.795813e-5	1.500000e-5
Time step dt (s)	0.00115955	0.00115979	0.00116176	0.00688705
Azimuth step $d\psi$ (deg)	0.5	0.5	0.5	0.5
Rotor revolution time (s)	0.8348760	0.8350488	0.8364672	4.958676

4.1.3.2. Solution Method

Finally, the simulations were executed by running the transient solver `pimpleDyMFoam` using the updated dynamic mesh approach as explained in Section 4.1.2.3. The `pimpleDyMFoam` solver is based on the PIMPLE algorithm as explained in Section 2.1.2.3 with dynamic mesh motion capabilities. Therefore, the mesh motions for each case were specified within the `dynamicMeshDict` according to Table 4.3 and the initial conditions of Table 4.6 were applied. The time step in the `controlDict` file was obtained by taking an azimuth step $d\psi$ per time step of 0.5 deg. Then, the time step dt follows from Equation 4.4 if the average rotational speed Ω of the rotor is known.

$$dt = \frac{d\psi \left(\frac{60\text{s}}{1\text{min}} \right)}{\Omega \left(\frac{360\text{deg}}{1\text{rev}} \right)} \tag{4.4}$$

For faster convergence the `0` folder in the case file was based on a `simpleFoam` solution. Therefore, first the steady solver `simpleFoam` was run using the MRF approach to account for the rotor rotation, see Section 4.1.2.1.

The simpleFoam run was initialized by using potentalFoam, to improve convergence by first solving the potential flow. Following this procedure the simulations with pimpleDyMFoam were found to converge quite fast also showing very stable behaviour.

The convergence criteria were mainly the resulting power, which was obtained through the turbinePerformance library, as well as the residuals and continuity. When the simpleFoam run was converged, several measures were taken in order to use the results as initial conditions for pimpleDyMFoam. First of all, the last time folder in the simpleFoam cases was renamed to 0, which is the default folder for the initial conditions. Moreover, the no slip boundary condition for the velocity U defined in the U file is slightly different within pimpleDyMFoam and must be adjusted. Therefore, the no slip condition established in simpleFoam through a zero fixedValue, was altered to a zero movingWallVelocity.

The NREL phase VI turbine was simulated with unsteady RANS based on the $k-\omega$ SST turbulence model. Therefore, the initial conditions for the parameters U, p, k and ω were located in U, p, k and files within the newly created 0 folder. The pimpleDyMFoam simulations could then be run based on these calculated conditions.

For the NREL 5MW simulations a hybrid DDES model is used based on Spalart-Allmaras turbulence modelling. In OpenFOAM it can be selected in the constant/turbulenceProperties file as SpalartAllmarasDDES. The reasoning to use this model is that due to the hybrid approach, regions near the wall are treated by RANS, while the outer flow is treated similar to a LES model. For the particular mesh this model already delivered good results in previous simulations and thereby it was justified to use it.

For the NREL 5MW nut and nuTilda files were included in the 0 folder, in order to run the SpalartAllmarasDDES model. The file nut sets the boundary conditions for the turbulent eddy viscosity ν taken as 3e-6, whereas within nuTilda the conditions for the Spalart-Allmaras variable $\tilde{\nu}$ are specified (4.5e-5).

To successfully run the simulations appropriate schemes and solution behaviour must be specified in the fvSchemes, respectively fvSolution files in the system folder. For the time schemes, which are given in the fvSchemes file, a backward Euler scheme is selected and thus applied for solving the Navier-Stokes equations. The convergence tolerance selected within the fvSolution file was in the order of 1e-5 for the velocity U and 1e-4 for the pressure p.

Finally, the simulations could be run. For the demanding computations the computational cluster Eddy of University of Oldenburg was utilised, see [17]. The phase VI cases were simulated with about 160 to 200 cores each, whereas for the 5MW cases due to the higher number of cells about 250 to 360 cores were chosen. Per day of simulation time about two to three rotor simulations could then be simulated. For faster simulations more cores would be required, but of course the available computational resources are limited.

4.2. NREL FAST Simulation Method

In this section the applied wind turbine simulation approach using NREL FAST is presented. The CAE tool FAST is capable of executing aero-hydro-servo-elastic simulations of horizontal axis wind turbines. Its composed of several modules, which have different functions such as modelling the aerodynamics or structure of the wind turbine, see Figure 4.17. At first, the aerodynamic model AeroDyn is described in Section 4.2.1. Next, both the lower fidelity ElastoDyn and higher fidelity BeamDyn structural models are discussed in Section 4.2.2. Finally, the simulation setup for the simulation cases using FAST, according to the matrix in Table 3.6, is presented in Section 4.2.3.

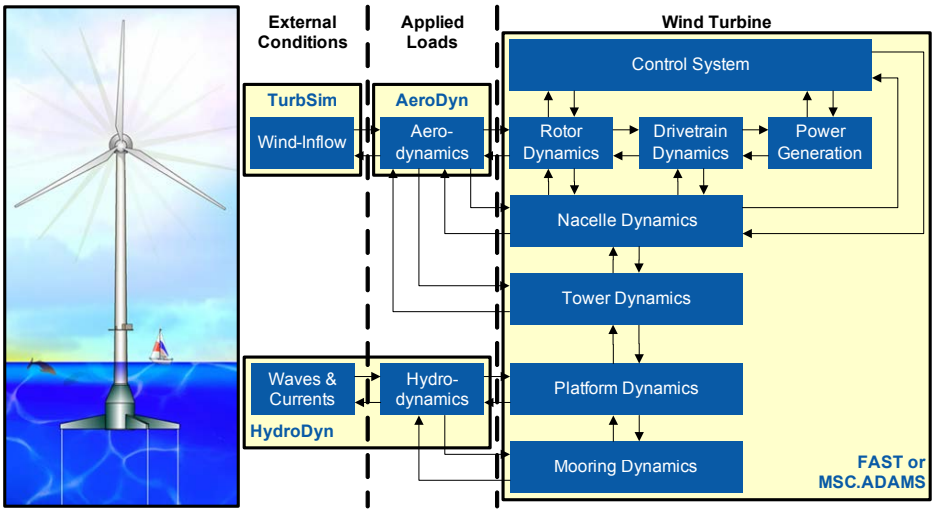


Figure 4.17: Overview of the FAST modularization framework [23].

4.2.1. Aerodynamic Model

As aerodynamic module for FAST AeroDyn version 15 is utilized for the simulations. The version 15 is a complete overhaul compared to earlier AeroDyn version, which is in accordance with the FAST modularization framework, see [20].

It is based on the actuator line principle, thus the three dimensional flow around the blades is estimated from two-dimensional cross sections via interpolation and integration technique. AeroDyn calculates the aerodynamic loads as distributed loads per spanwise unit length at several blade node locations [20].

If used with FAST AeroDyn receives the turbulent wind field from TurbSim [23]. In addition, it gets the deflected structural positions, orientations and velocities of the nodes from the applied structural module. Thereby, the nodes used within AeroDyn can be different compared to the nodes used within the structural module. This is accounted for through applied mesh mappings within FAST.

The two dimensional loads are calculated via BEM within AeroDyn. The wake influence is taken into account by calculation of the induction factors within the BEM via an iterative approach. The implemented BEM uses engineering correction methods to correct for three dimensional unsteady flow in extreme conditions (e.g. yaw). The effect of a skewed wake, sheared flow, influences due to the tower and structural motions is accounted for within the BEM [20]. Glauert's correction is applied at high axial induction factors [20]. In addition, Prandtl tip-loss and root-loss correction is used as well as a skewed wake correction from Pitt and Peters [20]. Therefore, three dimensional effects are captured as well through corrections within the BEM method.

The BEM can be set to a steady or unsteady airfoil aerodynamic model. For the steady model the static airfoil polars supplied to AeroDyn from e.g. measurements are directly utilized for the load calculations. In contrast to this, the unsteady airfoil aerodynamic module takes flow hysteresis, unsteady attached flow, flow separation at the trailing edge, dynamic stall as well as flow reattachment into account [20]. These semi-empirical corrections can be seen as dynamic corrections to the static airfoil results. There are three unsteady aerodynamic models within FAST: the original Beddoes-Leishman model and the Beddoes-Leishman versions from González and Minnema/Pierce [20].

In the blade aerodynamics also the influence of the tower is included via a potential flow, respectively tower shadow model or a superposition of both. The wind disturbance due to the tower can generally be explained as a velocity decrease upwind and downwind of the tower and an increase left and right of it. The calculated disturbance is applied at the blade nodes and used within the BEM calculations [20].

In addition, the loads on the tower, which are mostly due to form drag, can be calculated by AeroDyn based on tower diameter and drag coefficient. These calculations are again executed at several preselected tower nodes.

To successfully run AeroDyn within FAST several input files are required. First of all, in the FAST

case folder within the `AeroData` folder the airfoil polar data is located. Additionally, the main file for specifying the options for AeroDyn 15 is the file `AeroDyn15.dat`. Moreover, the blade aerodynamic sectional properties are specified in the `AeroDyn_blade.dat` file.

4.2.2. Structural Models

In this section the structural models of FAST are described. First of all, the lower fidelity model ElastoDyn is discussed in Section 4.2.2.1. Afterwards, it is elaborated on the higher fidelity model of BeamDyn, see Section 4.2.2.2.

4.2.2.1. ElastoDyn

The main structural-dynamic model implemented in FAST is ElastoDyn. It is capable of modelling the rotor, the drivetrain, the nacelle, the tower and the platform in case of an offshore turbine. Its inputs are the aerodynamic and hydrodynamic loads as well as the controller parameters and subtructure reactions for offshore turbines [18]. Its approach is based on a combined multi-body and modal-dynamics formulation. Whereas the blades and the tower are modelled through Euler-Bernoulli beam theory, the platform, nacelle, generator and hub are simulated by multi-bodies.

Within the Euler-Bernoulli beam theory for the blades only bending is taken into account. Thereby the effects of shear or axial and torsional DOFs are neglected [18]. The blade and tower motions which are considered, are assumed to result in small to moderate deflections. The mode shapes representing the motion through shape functions are constructed from polynomial coefficients via the Rayleigh-Ritz method [18]. The blades are modelled straight and isotropic [19]. The mode shapes are then applied on the nonlinear beam model. Inertial loads are included and added to the aerodynamic and hydrodynamic ones.

For a three bladed turbine more than twenty DOFs can be used [18]. The degree of freedoms to be considered can be switched on or off in the main ElastoDyn input file `ElastoDyn.dat`. For the blades two flap modes can be taken into account together with one edgewise mode [18]. For the drivetrain a torsional and azimuthal DOF can be activated. The tower is modelled with two DOFs for both fore-aft and side-to-side bending modes.

To run ElastoDyn within FAST structural properties (such as mass and inertia) must be specified in the main file. In addition, for the blades the stiffness in both edgewise and flapwsie directions has to be given for several sections within the `Blade.dat` file. ElastoDyn then interpolates those at a defined number of structural nodes. Similarly, the properties for the tower are defined in the `ElastoDyn_Tower.dat` file.

The outputs of the ElastoDyn structural model are the displacements, velocities and accelerations at the blade and tower nodes and if applicable at the hub, nacelle, drivetrain or platform. In addition, the reaction loads can be output.

4.2.2.2. BeamDyn

ElastoDyn has its limitations in some aspects for the modelling of large slender rotor blades. Especially, through the neglection of torsion, which is getting more important for these modern blades, errors may be introduced. In addition, effects such as bend-twist coupling (BTC) cannot be modelled at all due to the missing torsional DOF. Therefore, within FAST version 8 it is possible to use a higher fidelity structural modal for modelling the blades called BeamDyn.

BeamDyn is based on the GEBT, it therefore uses Legendre spectral finite elements [19]. It uses the full 6x6 cross sectional mass and stiffness matrices [19]. In comparison to ElastoDyn, it is thereby able to calculate axial, shear and torsional deflections as well due to the added DOFs. It allows for structural couplings such as structural BTC due to possible off-diagonal terms in the stiffness matrix. It can model curved or swept blades as well, thus it is also able to model effects as geometrical BTC. It uses nonlinear geometrically exact large deflections. Compared to ElastoDyn typically the number of cross-sections is larger, but the number of finite element nodes is generally quite low. This is due to the reason that a low number is often already sufficient. A Gauss or trapezoidal spatial integration technique is then used for interpolation.

The in- and outputs to BeamDyn are practically similar to ElastoDyn for the blades. The aerodynamic loads are used from AeroDyn and the blade root motions are taken from ElastoDyn. The outputs of the BeamDyn module are the blade motions as well as the calculated reaction loads.

If it is decided to use BeamDyn for the Blades, this needs to be specifically chosen in the ElastoDyn main file. The BeamDyn options can then be specified in the `BeamDyn.dat` input file. If BeamDyn is chosen, also a `BeamDyn_Blade.dat` file needs to be provided. In this file basically the full cross sectional mass and stiffness matrices are given at several stations.

4.2.3. Simulation Setup

In this section the simulation setup for both the NREL phase VI FAST simulations (Section 4.2.3.1) and the NREL 5MW simulations, see Section 4.2.3.2, is discussed.

4.2.3.1. NREL phase VI

Fortunately, there are already FAST input files for the NREL phase VI turbine delivered within the `CertTest` folder in the `FAST` main folder. In the `UAE_VI` folder one will find two cases for the NREL phase VI turbine. This means that no airfoil data and structural data needs to be generated by preprocessors and thus a lot of time can be saved. However, the given cases have to be modified in order to simulate and match the experimental cases. Therefore, the time step, wind speed, rotational speed, air density and dynamic viscosity were adjusted in the FAST certification Test10 according to table 4.6. In addition, the speed of sound c needs to be input in the AeroDyn main input file. It was calculated according to the Equation 4.5 derived from ideal gas law.

$$c = \sqrt{\gamma R T} \tag{4.5}$$

In Equation 4.5 γ is the adiabatic index, which is about 1.4 for air. R is the gas constant, which is 287 J kg^{-1}K^{-1} for dry air. Then, the temperature T from the experimental data can be used to calculate the speed of sound for each case. It is then applied within the AeroDyn input file. The used values for T and the calculated values for c are given in Table 4.7.

Table 4.7: Additional initial conditions for FAST simulations.

Case:	**1 (b)**	**2 (b)**	**3 (b)**	**5-6 (b)**
Experiment	S0700000	S07YSU00	R0600RD0	None
Temperature (K)	284.282	283.796	284.905	279.300
Speed of sound (m/s)	337.971	337.682	338.342	335.000

The NREL phase VI turbine is simulated using ElastoDyn also for the blades. It can be expected that although the lower fidelity structural model is selected accurate results can be obtained. This is due to the reason that the blade is straight and very rigid. Thus, the torsion is expected to be very close to zero and thereby the assumption of ElastoDyn of no torsion is deemed to be valid.

Within the ElastoDyn main file the DOFs to be activated were chosen. Moreover, the pitch slope, respectively yaw sweep, can be set in the ServoDyn main file `ServoDyn.dat`, similar to the motions as mentioned in Table 4.3. Also the wind speed is changed in the `InflowWind.dat` file according to the same values given in Table 4.8. In addition, within the AeroDyn main file several settings were altered. The most important settings for the different cases simulated are given in Table 4.8.

Table 4.8: Settings for the simulations with FAST.

Case:	**1-4 (b)**	**5-6 (b)**
ElastoDyn Settings:		
Blade flapwise DOFs	On	On
Blade edgewise DOF	On	On
Rotor teeter DOF	Off	Off
Drivetrain rot. flexibility DOF	Off	On
Generator DOF	Off	On
Yaw DOF	Off	On (off in case 6)
Tower bending DOFs	Off	On
Platform DOFs	Off	Off
Cone	0 deg	0 deg
Tilt	0 deg	5 deg
ServoDyn Settings:		
Torque control mode	None (constant rpm)	Bladed-style DLL
Yaw control mode	None (yaw rate case 2)	Bladed-style DLL (fixed yaw case 6)
Pitch control mode	None (pitch rate case 3)	Bladed-style DLL
AeroDyn Settings:		
Blade airfoil aerodyn. model	Beddoes-Leishman unsteady	Beddoes-Leishman unsteady
Beddoes-Leishman variant	Minemma/Pierce	Minemma/Pierce
Skewed-wake corr. model	Pitt/Peters	Pitt/Peters
Hub and Tip loss	Yes	Yes
Tower potential flow	No	No
Tower shadow	No	No
InflowWind Settings:		
Wind Type	Steady	Steady
Wind Shear	None	None

4.2.3.2. NREL 5MW

For simulating the NREL 5MW with FAST one can make use of the already existing cases in the `5MW_Baseline` folder within the `CertTest` folder. The folder includes a fully functional controller for yaw, torque and pitch control located in the `ServoData` subfolder. The controller was developed by Jason Jonkman et al. within the famous project in which the 5MW turbine was designed, see [22]. It is in the dynamic link library (DLL) format used by Bladed.

As a basis for the simulations the FAST certification case Test18 was modified. Test18 uses ElastoDyn as a structural model for the blades. Similarly, in the simulation cases 5 and 6 (b) ElastoDyn will be used. As the degree of torsion is very small for this turbine, with only a magnitude of about one degree below 30 deg yaw, see [14], relatively accurate results seem obtainable.

The most important settings for the simulations with the NREL 5MW turbine using FAST are again shown in Table 4.8. Notice that for cases 6 the yaw DOF was turned off compared to the cases 5 and a fixed yaw angle of 30 deg was used as initial condition in the ElastoDyn input file.

Other settings such as the time step for instance were chosen the same as in the OpenFOAM simulations as given in Table 4.6. In addition, the speed of sound was taken according to Table 4.7. As no experimental data for this turbine is available, no specific tuning in the FAST input files was required such as for the phase VI.

5

Developed Method

In this chapter it is explained how the coupled simulation method between FAST and OpenFOAM, named fastFoam, based on the code coupling environment MpCCI has been achieved. First of all, an overview of the coupling approach is given in Section 5.1. This includes detailed information on the changes that were done within the codes of FAST and OpenFOAM. Next, within Section 5.2.1 the mesh motion approach is outlined, which was divided into a mesh update accounting for solid body motions and the elasticity related motions. Then the calculation of the aerodynamic loads is explained in Section 5.2.2. Finally, the setup of the simulations using the coupled solver is presented in the last Section 5.3.

5.1. Coupling Approach

The coupling between FAST and OpenFOAM was established with the code coupling environment MpCCI developed by Fraunhofer SCAI. This has the advantage that an already functional and well proven coupling environment can be utilized and only needs to be extended for the specific codes. In particular, each code which should be coupled to MpCCI requires a code adapter. For OpenFOAM a code adapter was already existing, but it needed to be extended. For FAST such a code adapter was not available yet and thus needed to be implemented from scratch. The work on the implementation and extension of these code adapters was supported by employees from Fraunhofer SCAI due to their expertise. A detailed overview of the newly developed code adapter for FAST is given in Section 5.1.1, while the extended OpenFOAM code adapter is described in Section 5.1.2.

The coupling was set-up as a partitioned coupling, where both codes are kept on their own and are not merged into one. Thereby, only some changes in both codes need to be made and the codes are coupled by a coupling environment in this case MpCCI. The changes in the codes are thus minimally intrusive. This has other advantages for example it allows for quick updates of the different codes. Moreover, due to used standardised functions other extensions of the coupling, for instance to other parameters, are easily included.

In addition, the coupling was implemented based on a loose coupling strategy. Although MpCCI also allows for iterative or strong couplings a loose coupling strategy was deemed to be easier to implement and was found to be accurate enough for wind turbine FSI, see for instance [14]. Within the loose coupling the exchange between structural and fluid solvers is only applied once per time step.

A schematic representation of the implemented coupling is given in Figure 5.1. The red arrows indicate the transfer of the instantaneous positions and global turbine angles (yaw, azimuth and pitch for all blades) from FAST to OpenFOAM. Moreover, the blue arrows show the communication of the calculated loads (forces and moments) from OpenFOAM to FAST. This communication takes place via the integrated code adapters into the respective codes and the MpCCI server. The server can be seen as the brain of the coupling, which interfaces the code adapters and stores the received quantities.

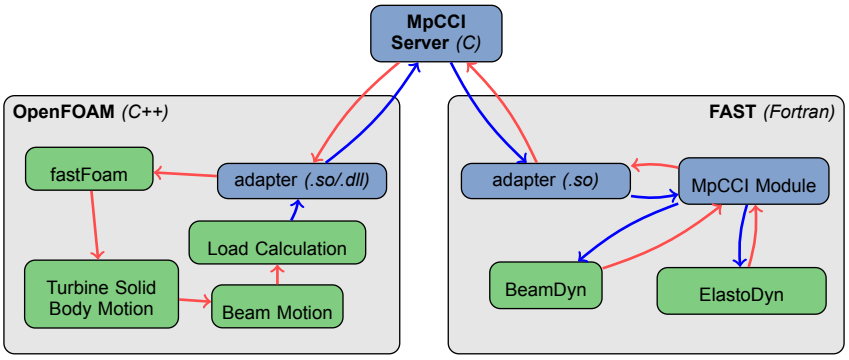


Figure 5.1: Schematic representation of the implemented coupling between FAST and OpenFOAM using MpCCI.

For the coupling both a FAST case folder and an OpenFOAM case folder are required. Of course the cases must match, thus the turbine to be simulated must be the same in both codes. Then the coupling can be set-up using the MpCCI graphical user interface (GUI), which automatically recognises the coupling possibilities if both case folders are given as an input. The procedure to set-up the coupling by using the GUI is shown in detail in Section 5.3. In addition, an overview of the entire workflow of the coupling by utilizing MpCCI is given in the Appendix D in Figure D.1.

5.1.1. FAST Adapter

The MpCCI FAST code adapter consists of two modules. In the MpCCI installation folder called `MpCCI` one finds the main adapter which interfaces the MpCCI-Server. In addition, within the FAST installation subfolder `dependencies` in the `Source` folder the MpCCI module is located in the `MpCCI` folder. This module is now part of the FAST modularization framework and called if FAST is run in coupled mode.

The adapter is written in C and compiled as .so. Its main functionality is to get the data from the MpCCI module of FAST and exchange it to the server. Furthermore, it receives the available coupling quantities from the server. Basically, it gets the positions and orientations as wells as the turbine control variables (yaw, azimuth and pitch) from the FAST MpCCI module. Moreover, it puts the loads calculated from OpenFOAM and received from the server into the MpCCI module.

The FAST MpCCI module calls at every time step functions of the MpCCI FAST adapter to send or receive data for the coupled quantities. It also allocates the required arrays for the coupling quantities. In addition, it overwrites the aerodynamic forces for the selected coupled geometries used within the structural models of ElastoDyn, respectively BeamDyn in the FAST simulation. As the meshes of the structural input load mesh of ElastoDyn and BeamDyn and the mesh for the received loads from MpCCI are not necessarily equal mesh mapping methods are applied. In addition, the loads do not necessarily have the same unit as for instance in ElastoDyn point loads are used (unit of N), while in BeamDyn distributed loads are applied (unit of N/m). This is taken care of by the applied mesh mappings.

The received loads from OpenFOAM are always distributed loads in N/m as this has several advantages as explained in Section 5.2.2. While in ElastoDyn the input load and output position and orientation mesh are the same, this is not the case for BeamDyn. Thus for both structural models mesh mappings are used within the MpCCI module. For ElastoDyn the mesh mapping only converts the obtained loads from N/m to N. The conversion follows the NREL mesh mapping algorithms as outlined by Spraque et. al. in [29]. It uses a nearest-neighbor algorithm to map different mesh discretizations and point to distributed loads and vice versa.

A complete overview for the FAST MpCCI module is given in Figure 5.2 as a flow diagram.

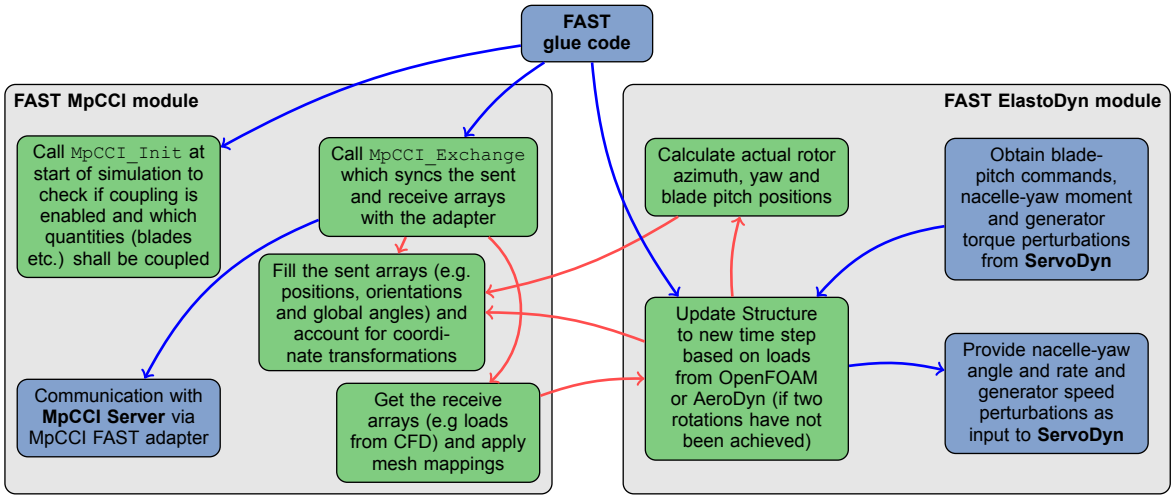


Figure 5.2: Flow diagram of the implemented MpCCI module within the FAST `Source` folder and its tasks at each time step.

5.1.2. OpenFOAM Adapter

The MpCCI OpenFOAM code adapter is also located in the MpCCI installation folder. It is compiled as C++ code using OpenFOAM `wmake` compilation. Basically it is a .so library which can be interfaced by OpenFOAM. In addition it includes several Perl scripts to identify the coupling capabilities from the OpenFOAM case folder within the MpCCI GUI.

The code adapter mainly gets the data for requested quantities from the selected OpenFOAM solver at each time step and communicates them to the MpCCI server. In addition, it stores requested values into allocated arrays from the MpCCI server, such that they can be accessed from the OpenFOAM solver. In particular for the FAST-OpenFOAM coupling it communicates the loads from the OpenFOAM solver. In this case this solver is called fastFoam, which is explained in Section 5.2. Moreover, it puts the obtained positions and global controller variables as yaw, azimuth and pitch into arrays.

The OpenFOAM adapter was able to couple face and volumetric data at the start of the project. However, due to the nature of the FAST-OpenFOAM coupling also coupling of lines (one dimensional beam node data such as positions or forces) or points (for instance if the hub is respresented as a point) needs to be established. Therefore, the OpenFOAM adapter was extended to this functionality.

Additionally, the possible quantities to be coupled were extended. Coupling positions and orientations as well as forces and moments is now possible as points or one dimensional lines. Also additional global variable coupling was included, which accounts for the coupling of the turbine controller related variables such as yaw, azimuth or blade pitch angles. This was also implemented into the Perl scripts, which scan the OpenFOAM case, such that if applicable these coupling quantities can be selected in the MpCCI GUI.

5.2. fastFoam Solver

One part required for the successful coupling of FAST and OpenFOAM is a specialized solver within OpenFOAM. The main functions of this solver, which following the naming scheme of OpenFOAM is called fastFoam, is to update the CFD mesh such that the blade positions correspond to the state in FAST. Moreover, it needs to calculate the loads (forces and moments) at the node locations, thus taking over this task from AeroDyn. The solver is fully integrated and based on OpenFOAM and written in C++. Its main part is included in Appendix E in Listing E.3. The two main tasks of mesh motion and load calculations will be discussed in detail in this Section. The mesh motion aspect is explained in Section 5.2.1 and the load calculations are presented in Section 5.2.2. Below in Figure 5.3 a flow diagram of its main functions is given, starting with the communication with the MpCCI server at each time step.

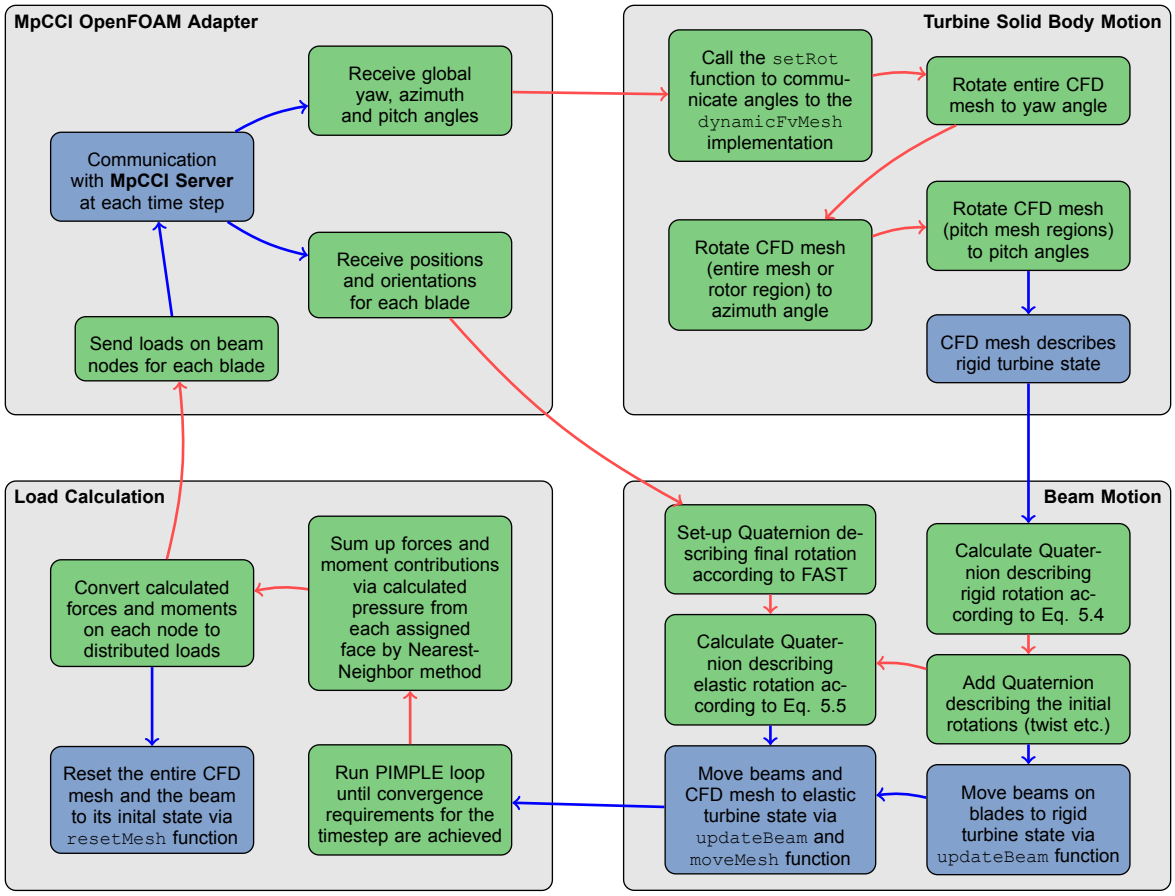


Figure 5.3: Flow diagram of the developed fastFoam solver and its tasks in OpenFOAM at each time step of the simulation.

5.2.1. Mesh Motion

In this section the mesh motion approach used within the fastFoam solver is described. Additionally to the solid body motions due the yaw, torque and pitch DOFs, also the blade elasticity needs to be included now. Thus, an extension of the mesh motion approach, such as used in the OpenFOAM simulations described in Section 4.1.2.3, is required. First off all, the turbine solid body motion approach is described in Section 5.2.1.1. Afterwards, the mesh motions applied due to the elastic nature of the blades are discussed in Section 5.2.1.2.

5.2.1.1. Turbine Solid Body Motion

The solid body mesh motions basically follow the same implementation as already used in the OpenFOAM simulations described in Section 4.1.2.3. However, there is one major difference as now the turbine angles such as yaw, azimuth and pitch are no longer read from a look-up table. In contrast to the previous implementation now these angles are accessible from the fastFoam solver as they are communicated from FAST. The solver then uses an update function called `setrot`, which updates those angles in the dynamic mesh library implementation. Then the mesh update is applied based on these angles and the parameters read from the `dynamicMeshDict` file.

The mesh update is always performed from the initial mesh state such as given in the `constant` subfolder in the OpenFOAM case folder. When the mesh update is applied according to the turbine angles from FAST, it is taken care that the CFD mesh in OpenFOAM always has the same solid body states such as in FAST. This means that the yaw, azimuth and pitch states in both codes are the same.

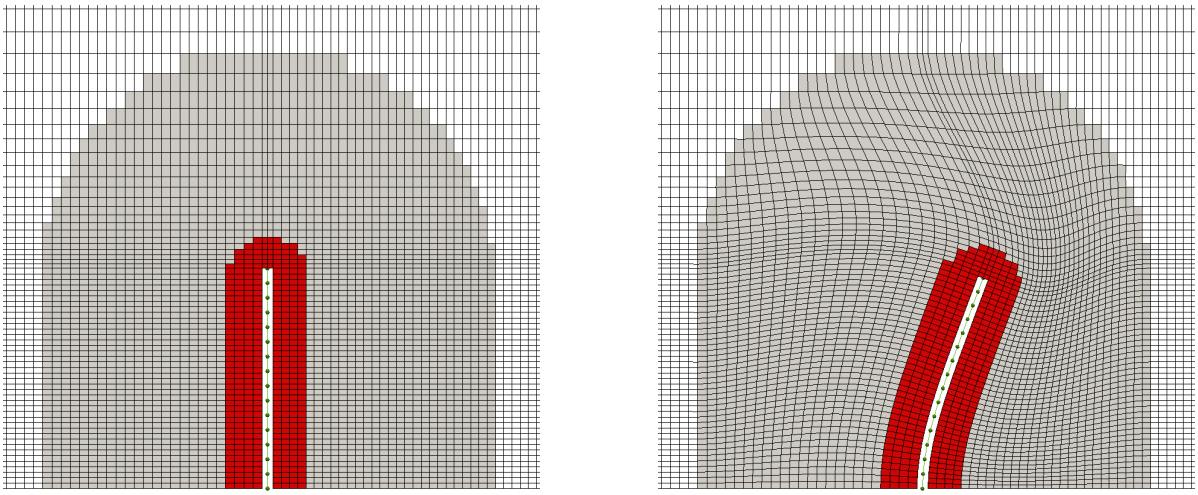
5.2.1.2. Beam Motion

In addition to the dynamic mesh motion due to controller related inputs, such as yaw, azimuth and pitch, the effect of elasticity must be included within OpenFOAM. Due to the elastic structure resulting in deformations the mesh needs to be deformed accordingly. This is especially required for the larger

modern rotor blades, which show an increased flexibility. Assuming rigid blades will lead to introduced errors which should be avoided.

An approach for such a mesh deformation accounting for flexible blades was achieved by Dose et al., see [9]. By implementation of an entire FEM solver into OpenFOAM using non-linear finite beam elements based on GEBT the structural displacements were obtained and the fluid mesh is updated accordingly [9]. The procedure follows a partitioned coupling between structural and fluid solver. The fluid mesh is updated if both flow and structure have converged [9].

As access to the code is provided the mesh updating, which follows the solution of the structural solver, can be used for the elasticity related mesh update. Thus, the structural solver is now directly integrated into FAST, but as FAST uses beam elements as well, the fluid mesh can be updated by the same approach such as achieved in [9]. An example for the mesh update based on the implementation of Dose et. al is given in Figure 5.4. In the example a beam is introduced in the white structure in the centre, see the small green beam nodes. The red mesh region then moves exactly with the beam node displacements, for the grey mesh this movement is smoothed out, whereas the white background mesh does not move at all.



(a) Example mesh before deformation.

(b) Example mesh after deformation.

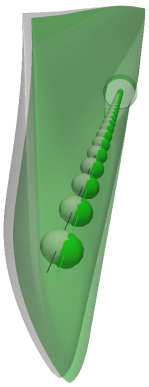
Figure 5.4: The mesh movement approach accounting for elastic structures based on the implementation by Dose et al. [10].

The beam mesh is then specified within a `beamMeshDict` in the `system` folder, where it must be taken care of that the same structural mesh as used in FAST is taken. An example of the `beamMeshDict` is given in Listing E.1 in the Appendix E.

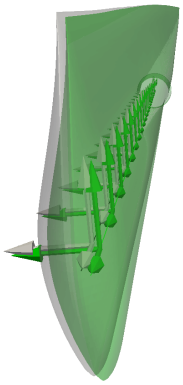
This approach uses non linear finite beam elements. However, as the blades are represented in three dimensions, whereas the beam elements are only in 1D, a special interpolation technique is applied. This technique interpolates the given deflections at the beam nodes to the blade surface. In particular, this means that the points in OpenFOAM, which define the faces and cells related to the blade mesh and with it the blade geometry itself, are translated and rotated according to the translation and rotation of the beam nodes. In addition, a certain region around the blade mesh is moved as well in order to have a smooth cell structure and no sudden jumps in the mesh. If the distance to the blade surface gets larger, the movement of the cells related to the points is reduced. Until a certain distance, which can be specified is met, the mesh stays constant and is not undergoing any elasticity related mesh movement at all.

An overview of the implemented beam is shown in Figure 5.5 for the NREL phase VI turbine. Two different states (rigid and elastic) of the blade are shown including the finite beam elements.

(a) Rigid (green) versus elastic state (grey).



(b) Beam nodes along the blade span.



(c) Local node coordinate systems.

Figure 5.5: The implemented one dimensional beam for the NREL phase VI blade for two different states.

To be able to update the mesh based on the beam motions, the beam must always follow the blade. Therefore first, the beam is initialised with the same node locations such as used in the selected structural model for the blades within FAST, see Section 4.2.2. This initial state includes blade twist, initial pitch and yaw, coning or tilting of the rotor. It is specified within a `beamMeshDict` file in the `system` folder within the OpenFOAM case folder. It must be taken care of that the initial state is the same such as in the FAST case folder. Thus, the same beam node locations and twist, pitch, yaw, cone and tilt angles have to be specified. However, those data can easily be extracted from the FAST case.

The initial rotation of the beam is calculated using rotation quaternions. Quaternions are an efficient way to represent rotations in three dimensionanl spaces. They avoid problems with gimbal lock and are quite efficient. Quaternions can be created in several ways including a set of three given Euler angles, or a rotation about a given axis by a specified angle. In OpenFOAM they can be used via the quaternion class.

The position and orientation of the beam can both be represented through vectors. The position vector simply entails the beam node positions. In addition, the orientation can be represented through a set of three vectors, representing the local coordinate system at each beam node. Let these vectors be given as a vector u, then the initial rotation is described through the rotation matrix R_{init} obtained from quaternion q_{init} by Equation 5.1.

$$u_{init} = R_{init} u_{straight} \tag{5.1}$$

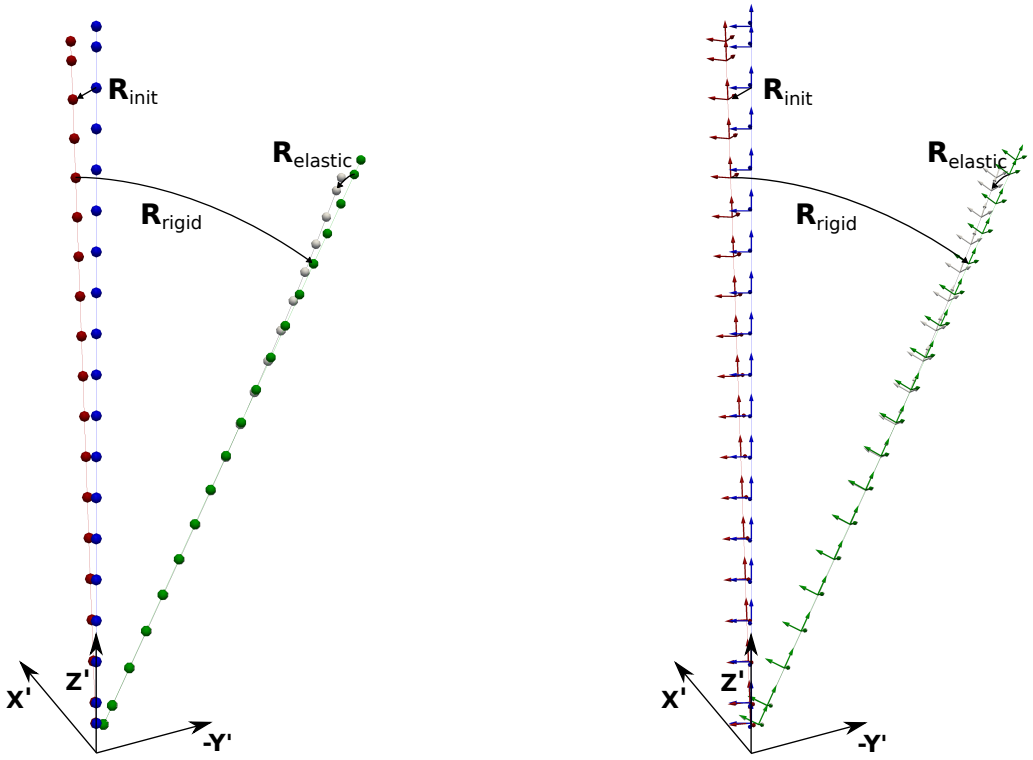
Where $u_{straight}$ is the straight blade beam position vector or the global coordinate system for the beam node orientation vectors. Then u_{init} is the initial position vector for the beam nodes, which includes blade twist, initial pitch and yaw, coning or tilting. Or similarly u_{init} describes one of the three

initial beam node orientation vectors, also including blade twist, initial pitch and yaw, coning or tilting. The initial R_{init} is derived from the initial rotation quaternion q_{init} as given by Equation 5.2.

$$q_{init} = q_{yaw_0} q_{tilt} q_{cone} q_{pitch_0} q_{twist} \tag{5.2}$$

Where q_{yaw_0}, q_{tilt}, q_{cone}, q_{pitch_0} and q_{twist} represent the initial yaw, tilt, cone, initial pitch respectively twist rotations about the related axis. Notice that quaternion multiplication is not commutative and the given quaternion is described by a rotation about the twist axis first and finally a yaw rotation.

The corresponding mesh translation and rotation is shown in Figure 5.6 for the implemented NREL 5MW beam mesh. In Figure 5.6 the initial rotations and corresponding translations of the beam nodes due to R_{init} are related to the known twist, as well as the application of a 5 deg tilt and a 30 deg yaw angle. The corresponding local coordinate system rotations are shown as well. For a summary of the used global coordinate system see Figure C.1 in Appendix C.



(a) The beam mesh nodes.

(b) The local coordinate systems.

Figure 5.6: The NREL 5MW beam mesh states from straight blade (blue) to initial state (red) via the rigid state (green) to the final elastic blade state (grey).

This is the first step to ensure that the beam nodes used within the OpenFOAM CFD have the same initial state as in FAST. This only needs to be initialized once at the beginning of the simulation. However, as a next step the CFD mesh is undergoing solid body motions related to the pitch, azimuth and yaw rotations communicated from FAST such as described in Section 5.2.1.1. Therefore, the beam needs to follow these motions to make sure that the implemented one dimensional beam mesh corresponds to the updated three dimensional CFD blade mesh.

Thus, another rotation sequence is used to update the beam mesh for the yaw, torque and pitch rotations. The updated rigid or solid body motion state u_{rigid} is achieved according to Equation 5.3.

$$u_{rigid} = R_{rigid} u_{init} \tag{5.3}$$

Where R_{rigid} corresponds to the rotation matrix which entails the yaw, torque and pitch rotations obtained from FAST via the coupling. It is obtained from a combined quaternion q_{rigid}, whose decomposition is shown in Equation 5.4.

$$q_{rigid} = q_{yaw} q_{azimuth} q_{pitch} \tag{5.4}$$

Where q_{yaw}, $q_{azimuth}$ and q_{pitch} describe the instantaneous yaw, azimuthal and pitch rotation from the initial state at each time step. This is applied within the fastFoam solver at each time step via an `updateBeam` function, see Listing E.1 in the Appendix E.

This `updateBeam` function takes the calculated displacements for the positions and the Euler angle rotation (in roll, pitch, yaw convention) for the orientations as inputs. Both is easily obtained from the calculated rigid state. The displacements are simply the rigid position minus the initial position. And the Euler angle rotation is directly obtained by conversion from the rigid quaternion q_{rigid} to Euler angles. A function for this conversion is available within the OpenFOAM quaternion class.

The rigid beam update ensures that the one dimensional beam mesh is moved to the correct positions. Also it ensures the correct blade orientations, which are similar to the blade mesh state obtained due to the solid body CFD mesh update.

Finally, the elastic state of the blade is considered. Therefore, the positions and orientations such as calculated from the structural model in FAST are used. To apply these, again the `updateBeam` function is used to move the beam. In addition, now a `moveMesh` function is considered as well. This function takes similar inputs as the `updateBeam` function and moves the mesh from the rigid state to the final elastic state via a movement of the mesh points.

Therefore, for the node positions the displacements due the elastic deformation needs to be used as an input to both functions. They are obtained by taking the difference between the communicated instantaneous deflected positions from FAST and the previously calculated rigid body positions. In addition, the rotation from rigid to elastic state must be specified within the functions.

This is calculated by first constructing a rotation quaternion q_{final} from the obtained instantaneous orientations from FAST communicated as Euler angles (roll, pitch, yaw convention). This quaternion describes the entire rotation from the global coordinate system to the instantaneous local coordinate system for each beam node at each time step. However, within both functions only the Euler angle rotation from rigid state to the elastic state needs to be used. This difference is obtained from the rotation quaternion $q_{elastic}$ as calculated in Equation 5.5.

$$q_{elastic} = q_{final}\left(q_{rigid} q_{init}\right)^{-1} \tag{5.5}$$

Equation 5.5 shows one of the advantages of quaternion calculations, as the difference between two or more rotations can easily be calculated by using the quaternion properties. After obtaining $q_{elastic}$, which corresponds to the rotation from the rigid state to final state, the corresponding Euler angle rotations are obtained by using the quaternion to Euler angle conversion within the OpenFOAM quaternion class.

Then both the one dimensional beam and the corresponding three dimensional CFD blade mesh are updated for the elastic deformations. Finally, the beam is reset to the initial state by a `resetMesh` function. This has the advantage that similar to the solid body motions as described in Section 5.2.1.1 at each time step the mesh update related to the elasticity starts from the initial state. Therefore, possible small errors for instance due to rounding etc. do not add up.

5.2.2. Load Calculation

After the mesh in OpenFOAM has obtained the correct state, corresponding to the turbine state in FAST at each time step, the main task of the fastFoam solver is to obtain the aerodynamic loads. These should be obtained at the beam node locations.

Therefore, first an OpenFOAM solver is utilized in order to compute the pressure and velocity field by solving the Navier–Stokes equations. The selected solver is `pimpleDyMFoam`, whose pimple loop is simply merged into the fastFoam solver.

Then within the fastFoam solver the calculated fields are used. At each faces on the blade surface the value for the pressure can be accessed. Multiplying those by the known face area results in the pressure force. The pressure forces are the main contributors in magnitude to the total force. But also the viscous forces, which can be obtained from the stress tensor, have to be included.

Thus, the forces at each face on the blade surface are known. However, the forces and also the moments need to be known at the beam node locations. These beam nodes are normally centred within the blade. Thus the meshes of fluid and structure are non-matching also in dimension and a matching

technique needs to be used. Therefore, a nearest neighbour method is applied, where the forces on each face are assigned to one beam node. In this method for each face the spanwise distance to the blade nodes is calculated based on the face centre coordinates. The face is then assigned to the beam node to which the spanwise distance is the smallest. The total acting force on each node is then simply the sum of all the forces on all assigned faces.

The procedure is shown in Figure 5.7, where the faces at the outer part of the blade are matched to three beam nodes. The first node corresponds to the faces shown in red and thus accumulates all the surfaces from these faces. Similarly, the second node shown corresponds to the blue faces, while the green faces are related to the last node (furthest from the point of view).

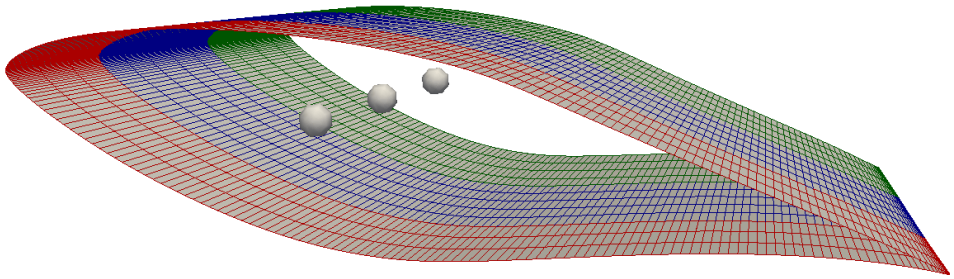


Figure 5.7: Mesh matching of fluid mesh (blade surface faces) and structural mesh (beam nodes) with Nearest-Neighbor method.

In addition, for the moment the assigned faces for each node and the corresponding forces are taken into account. Therefore, the moment arm between face centres and beam node coordinates is considered. Multiplying both pressure and viscous forces by this arm the moment contribution due to one face is obtained. By summing up the moment contributions from all assigned faces the total moment at each beam node is calculated.

Finally, the forces (units of N) and moments (in Nm) are converted to distributed loads, where the forces obtain units of N/m and the moments are in Nm/m. Therefore, the forces and moments per unit span must be considered. Thus, the previously calculated forces and moments were divided by the average spanwise length spanned by the assigned node faces.

The conversion to distributed loads is required as due to the nearest neighbour methods load fluctuations may occur. This can be explained due to a difference in the number of spanwise faces which may be assigned to each node. For instance, for one node a number of 3 faces in spanwise direction may be assigned, whereas for the next node 4 or even 5 faces may be assigned, see for instance Figure 5.7. Thus, resulting in sudden jumps in the forces, which are nonphysical and only a resultant of the force projection method.

Therefore, it was decided to convert to distributed loads which show a much smoother load distribution. In addition, the usage of distributed loads follows the AeroDyn calculations, which also result in distributed loads. Thus, the implementation of the loads within FAST will be easier.

5.3. Simulation Setup

The fastFoam simulations are run based on the FAST and OpenFOAM cases such as set-up in Section 4.1 and 4.2. The coupling is constructed by utilizing the MpCCI GUI. In the first step, the models step, the codes to be coupled and the corresponding cases are selected, see Figure F.1 in Appendix F. The versions of the codes were chosen. For OpenFOAM version 3.0.1 and for FAST version 8.15.00 was taken.

In the next step, the so called coupling step, the quantities to be coupled were assigned as can be seen in Figure F.2. Therefore, the blades which are automatically recocgnized as line meshes and the corresponding variables, the positions and orientations as well as the forces and moments, are selected. In addition, the global variables yaw angle, azimuth angle and the blade pitch angles are chosen.

Next, within the monitors step the quantities to be monitored can be selected as seen in Figure F.3.

The built-in MpCCI monitor can be used to display information such as the transferred and received data. The monitor can either be run in runtime or after the simulation has been run.

As the next step, MpCCI related properties regarding the simulation are set within the edit step, see Figure F.4. This includes for instance the time tolerance, which is the tolerance MpCCI takes into account for matching the times of the different codes.

Finally, in the go step the coupling configuration including the scheme and starting time are chosen, such as shown in Figure F.5. Moreover, for FAST the coordinate system for the transfer is selected and for OpenFOAM the solver in this case fastFoam is chosen. In addition, for OpenFOAM properties related to simulating in parallel have to be selected such as the MPI and the decomposition method as well as the number of processors for the parallel run.

For simulating on the local computer one only needs to start the MpCCI server and both codes FAST and OpenFOAM by clicking on the start button. However, as the simulations were carried out on a computational cluster, where access is only provided by terminal and thus the simulations could not be started via the GUI, a different approach is required. In particular, the coupling is setup as it would be run locally via the GUI on the local computer. The setup can then be saved as MpCCI project within a `.csp` file. Finally, the coupled simulations, which are set-up in the project file, are run on the cluster via an `mpcci batch` command, whose input is the locally generated project file. To allow for this approach the respective FAST and OpenFOAM cases and their file structure must be similar to the one on the local computer.

For the simulations on the cluster the intelmpi is chosen and selected within the GUI. The chosen coupling scheme is of explicit type, thus a loose coupling where exchange between both codes only takes place once per timestep. The implementation in the code adapters is such that FAST exchanges data after each iteration, whereas within the fastFoam solver data is exchanged before the iteration. This has to be accounted for within the coupling scheme. Therefore, in the go step in the initial quantities transfer tab for FAST receive is selected, whereas for OpenFOAM exchange is chosen. This means that FAST will first receive data while OpenFOAM can both receive and send. This scheme is visualized in Figure 5.8.

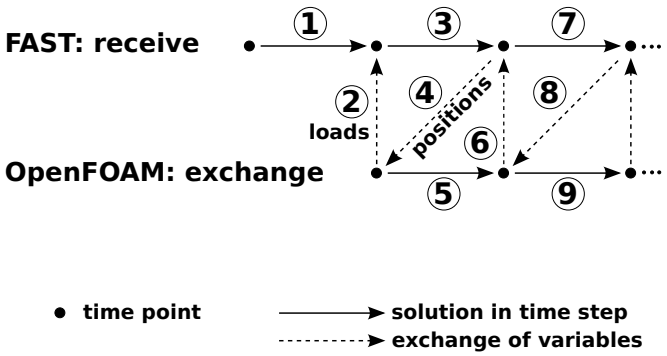


Figure 5.8: The choosen coupling scheme according to MpCCI.

In addition, the time for starting the coupling is chosen. It is unequal to zero for all cases. This is due to the reason that in the first time steps it is not coupled as the CFD code is started with time step control. This means that the time step for fastFoam in the very beginning of the simulation is slowly increased until the final time step as specified in the `controlDict` is reached. This is done to ensure good convergence of the CFD. Moreover, to improve the convergence the CFD code fastFoam is started with an initial solution in the `0` folder obtained from `simpleFoam` based on a rigid blade.

As both codes are not coupled in the first iterations special care must be taken that both codes can be run. This means that for fastFoam the mesh motions have to be applied without knowing the exact values for the yaw, torque and pitch angles. Therefore, it is assumed that these motions undergo constant slopes in the start-up phase. These slopes can be specified within the `beamElementControlDict`, see Listing E.2, in the `system` folder. Then when the coupling is initiated the exact values from FAST are applied for the CFD mesh motions. However, this should be relatively close to the predicted values to disallow sudden jumps for instance in azimuth. Note that this is normally not problematic, because in the beginning of the simulation the influence due to the controller within FAST is rather small. Thus, for instance a predicted azimuth due to a constant rotation will be close to the exact azimuth as obtained

from FAST. The blade is then still assumed to be undergoing a rigid rotation as the elastic displacements are kept as zero, as no displaced blade positions are known.

Moreover, the forces from CFD are not transferred to FAST until the end of the start-up phase when the coupling initialization time is reached. Therefore, the loads as calculated from AeroDyn are taken. When the coupling is initialized the quantities are transferred between both codes. The fastFoam solver obtains the exact yaw, azimuth and pitch angles as well as the elastic blade states (positions and orientations). The CFD mesh is then transformed using these values. However, the elastic displacements should not be applied in one timestep to the CFD mesh as these could result in convergence issues. Therefore, the displacements are smoothed into the CFD mesh over a selectable proportion of a rotation to ensure convergence of the CFD solver.

Then the CFD can converge, which usually takes a couple of rotations. Thus, although the loads obtained from CFD are already transferred to FAST after initialisation of the coupling the loads are not applied yet. For now the FAST MpCCI adapter applies the forces from CFD only after two rotations have been reached. These two rotations are a general experience value, which should ensure that the applied CFD loads have converged. Thus, until the simulation time in FAST is equivalent to two rotor rotations the forces from AeroDyn are applied. This application also follows a smoothing procedure, where according to a linear function the weighting of the CFD loads is increased. Between two and two and a half rotor rotations an average between the BEM and CFD loads is taken, based on this linear function. This is done in order to prevent a sudden change in loads from one time step to another, due to the change in aerodnyamic modelling (CFD versus BEM). Then from two and a half rotations only the CFD loads are considered within FAST.

The entire procedure in time can be seen in Figure 5.9 for different aspects and both of the simulation tools. In the Figure the different times, for which some can either be set via MpCCI and dictionaries or are hardcoded for now, have been highlighted.

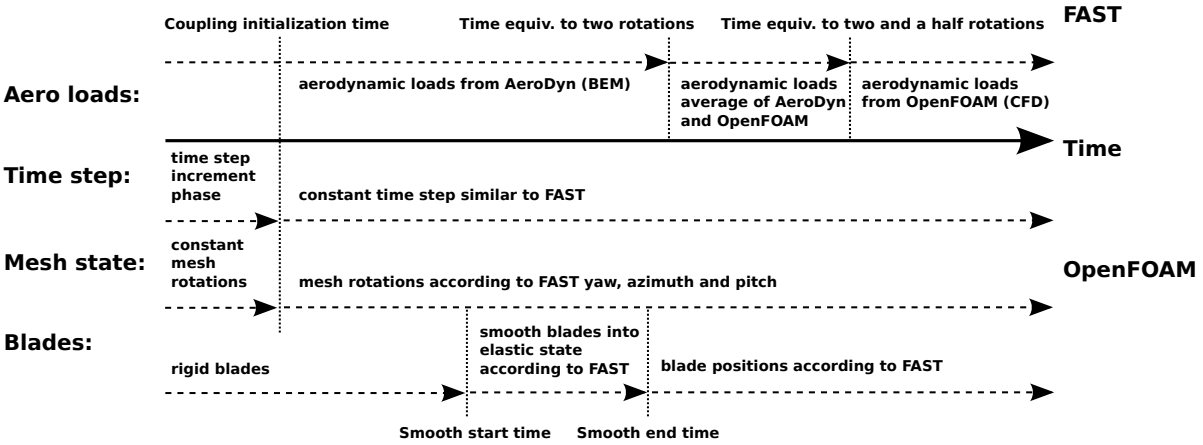


Figure 5.9: The coupling configuration as a function of time.

The selected times for the different simulations executed with fastFoam can be depicted from Table 5.1.

Table 5.1: Different times used in the fastFoam simulations.

Case:	1 (c)	2 (c)	3 (c)	5-6 (c)
Coupling initialization time (s)	0.005	0.005	0.005	0.05
Smooth start time (s)	0.2087	0.2088	0.2091	4.9587
Smooth end time (s)	0.4174	0.4175	0.4182	8.6777
Time equivalent to two rotations (s)	1.6698	1.6700	1.6729	9.9174
Time equivalent to two and a half rotations (s)	2.0873	2.0875	2.0911	12.3968

6 Results

To validate the FAST-OpenFOAM coupled method (fastFoam) simulations will be executed and results will be compared to experimental data. In addition, different simulations will be run based on stand-alone OpenFOAM and NREL FAST. This is done in order to compare the different simulation approaches and to elaborate on their validity.

The setup for the simulations in OpenFOAM is according to Section 4.1 and the simulations in FAST follow the setup in Section 4.2. Finally, the combined approach fastFOAM is utilized as described in Chapter 5.

The simulations were executed based on the simulation matrix given in Table 3.6. First of all, the NREL phase VI turbine has been simulated using the three different methods namely OpenFOAM, FAST and fastFoam. The obtained results are discussed in Section 6.1 based on the experimental data from NREL, see Hand et al. [30]. In addition, the NREL 5MW turbine was simulated representing a modern turbine with more elastic blades. The corresponding results are presented in Section 6.2. Moreover, in Section 6.3 an overview on the computation time for the different methods is given.

6.1. NREL phase VI

For the NREL phase VI turbine four different cases were simulated. At first, a mesh convergence study is done which addresses the requirement on the mesh refinement for the CFD simulations of OpenFOAM and the fastFoam solver. Then in Section 6.1.2 case 1 from the simulation matrix (Table 3.6) is discussed regarding normal operating conditions for the phase VI turbine. Afterwards, case 2 where the turbine undergoes a yaw sweep from zero to 30 deg yaw angle is presented in Section 6.1.3. Then case 3 is considered in Section 6.1.5, where a pitch slope is applied and the effective angle of attack is reduced.

Finally, for the phase VI turbine a partial power curve for lower wind speeds ranging from 5 to 10 m/s was simulated and is examined in Section 6.1.5. This was done in order to ensure that the agreement of the different methods with the experimental results is not only limited to specific wind speeds.

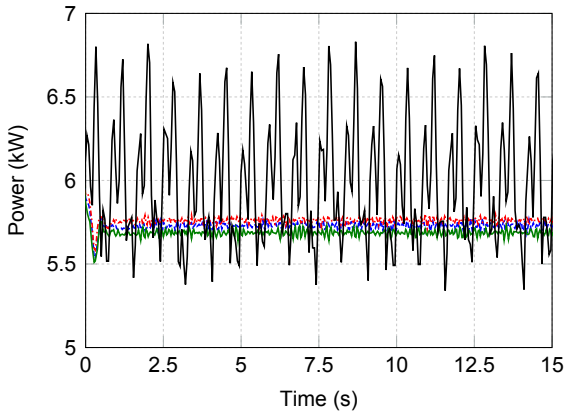
6.1.1. OpenFOAM Mesh Convergence

First of all, a grid convergence study was run based on the different meshes generated such as given in Table 4.1. The results for the low-speed shaft power, thrust and torque of the different simulated meshes are given in Figure 6.1 with respect to the experiment. From Figure 6.1 it can be seen that the mesh with the highest number of cells shows the best agreement with the experimental results. However, the results for the baseline mesh with intermediate mesh refinement are quite close to the refined mesh. The results for the coarse mesh show the largest deviations.

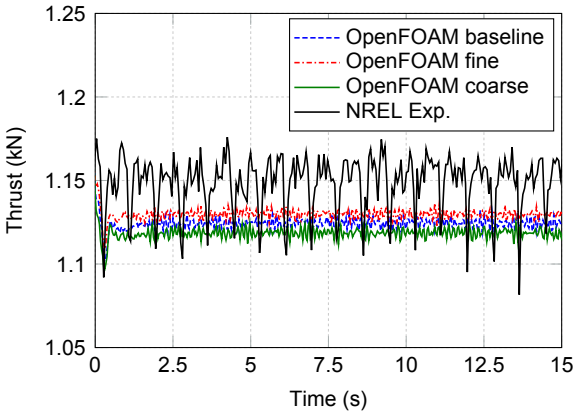
In addition, the power, thrust and torque fluctuations are larger for the results from the coarsest mesh compared to the baseline and refined mesh. The fluctuations for the experimental results are much larger compared to the simulations. Especially the low-speed shaft power shows extremely large fluctuations. An explanation for these fluctuations is that the rotor may not be perfectly balanced. A small deviation of blade mass and center of gravity effects the rotor inertia, thereby leading to imbalances and thus fluctuations. In addition, the experiments of course include the tower, thus resulting in

tower effects such as tower shadow and wake influences.

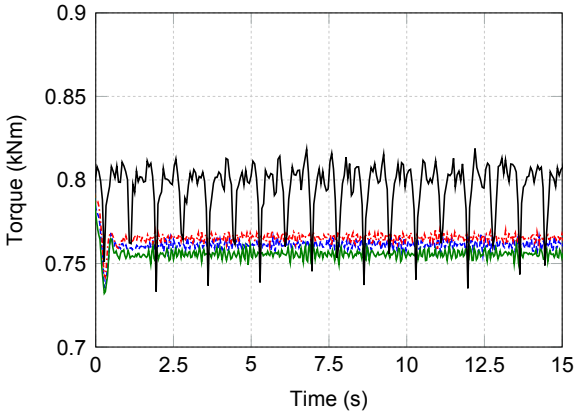
In the diagrams a dip in power, thrust and torque seems to occur at multiples of one rotor rotation taking about 0.8 s. This corresponds to a 1P frequency, thus occurring once per rotor revolution. However, the effect of the tower should correspond to the 2P blade tower passing frequency due to the reason that at each revolution the two blade are passing the tower once. By closer investigations also a decrease in power, thrust and torque happens at the 2P frequency. A pattern can be observed where a large dip occurs followed by an increase and then a much smaller decrease is present. The reason for this may be that the 2P frequency is present, but due to rotor imbalances the power dip is different in magnitude, depending on which blade passes the rotor.



(a) Low-speed shaft power.



(b) Thrust.



(c) Low-speed shaft torque.

Figure 6.1: General turbine parameters according to simulations and experiment for the mesh convergence study at 7 m/s wind speed.

The resulting mean power, thrust and torque were calculated in Table 6.1 for the different simulations and the experimental data. Moreover the Table shows the deviation to the experimental data. As can be seen the deviation of the mean values to the experimental data decreases with the mesh refinement. However, as the resulting differences between the baseline and the fine mesh are not very large the baseline mesh was accepted. This is due to the reason, that with the increased number of cells for a refined mesh the computational requirements and thus time increase as well (here by about 49 percent). To avoid too high computational times, while still ensuring relatively accurate results the mesh with intermediate refinement (baseline) was accepted. The baseline mesh was therefore used for the OpenFOAM and fastFoam simulations of the NREL phase VI turbine (case 1-4 of the simulation matrix in Table 3.6).

Table 6.1: Results for the mesh convergence study.

Type:	Baseline	Coarse	Fine	Experiment
Mean Power (kW)	5.729	5.688	5.761	5.990
Error to Experiment (%)	-4.355	-5.051	-3.830	-
Mean Thrust (kN)	1.124	1.119	1.130	1.150
Error to Experiment (%)	-2.205	-2.698	-1.731	-
Mean Torque (kNm)	0.761	0.756	0.765	0.796
Error to Experiment (%)	-4.346	-5.042	-3.821	-
Computational Time (d)	9.251	8.025	13.790	-
Difference to Baseline (%)	-	-13.252	49.065	-

6.1.2. Normal Operating Conditions

The simulations under normal operating conditions (case 1) are compared to the NREL experiment S0700000, relating to a mean wind speed of about 7 m/s. For the simulations for FAST and fastFoam the set-up of the FAST code is given in Table 4.8. For the fastFoam and OpenFOAM simulations the baseline mesh has been used with the set-up such as described in Section 4.1.3.

The obtained power, thrust and torque of the rotor are given in Figure 6.2 and compared to the experimental data. The fastFoam values are output from both FAST, as it is the main simulation tool and by the OpenFOAM postprocessor directly from the CFD simulation. In Figure 6.2 the fastFoam FA legend entry indicates the FAST output, whereas the fastFoam OF entry relates to the OpenFOAM output of the fastFoam simulations.

It can be seen that for the thrust and torque a better agreement to the experiment is obtained by the OpenFOAM simulations utilizing rigid blades and the fastFoam simulations with deformed blades compared to the FAST only simulations. For the power the agreement is similar, while FAST is over-predicting OpenFOAM and fastFoam simulations show a slight under prediction.

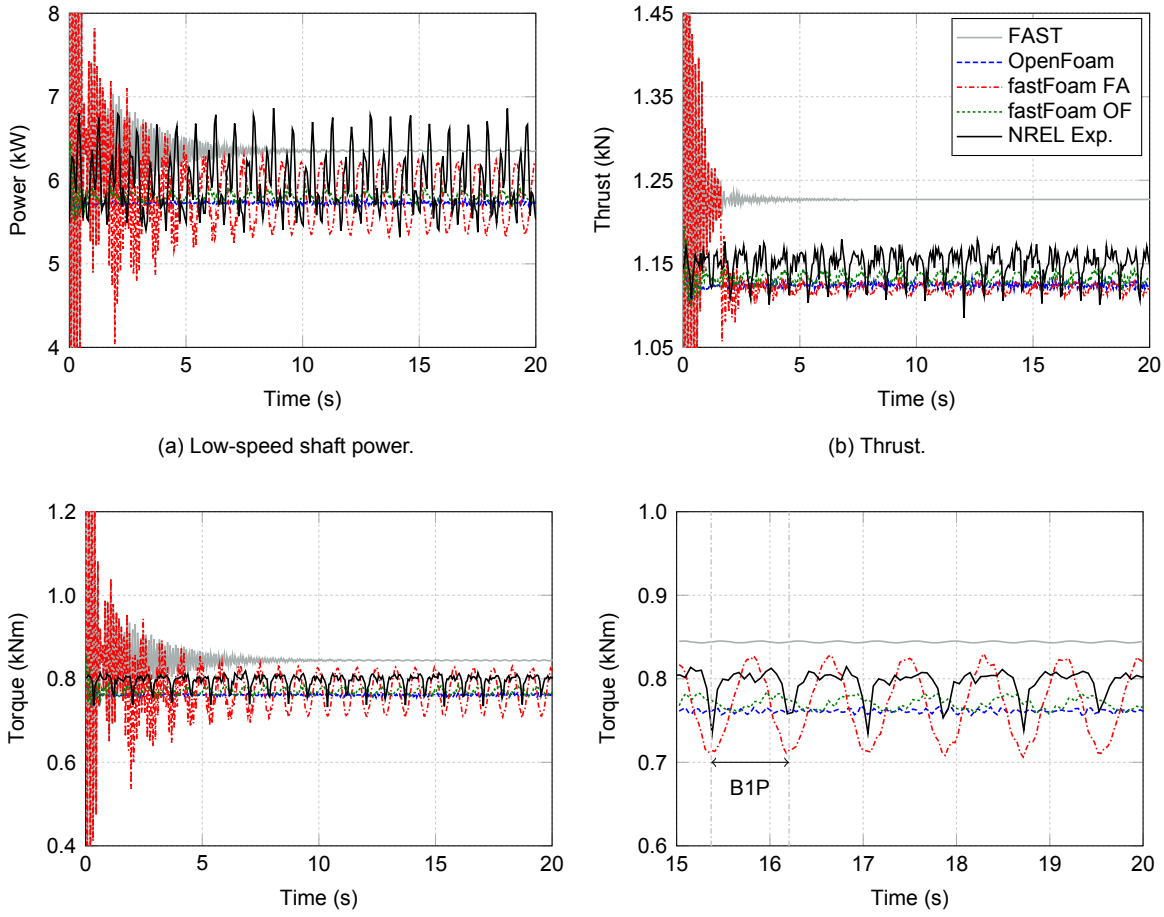
It is interesting to observe that in the fastFoam simulations a sinusoidal behaviour is present in the power and thrust, while FAST and OpenFOAM results converge towards a steady value. The power variations must be driven by sinusoidal variations in torque as the rotational speed is kept constant, which can be proven by investigating Figure 6.2c. For the OpenFOAM output the low-speed shaft torque is nothing else than the aerodynamic moment composed of the aerodynamic forces in the rotor plane. For FAST it is internally calculated in the ElastoDyn module. The ElastoDyn module includes a detailed drivetrain model including drivetrain torsion, acceleration and inertia effects, see [18]. However, for this simulation for the ElastoDyn settings the drivetrain and generator degree of freedoms were disabled, see Table 4.8.

The sinusoidal variations for the fastFaom results occur with approximately a 1P frequency for both output methods, see Figure 6.2d. This could indicate that there might be a slight fluctuation of the aerodynamic moment and thus forces contributing to the torque as a function of the azimuthal position. For the fastFoam FAST results, which include the drivetrain modelling, the fluctuations are amplified and the phase has shifted slightly compared to the pure aerodynamic torque obtained from fastFoam with OpenFOAM postprocessing. The highlighted blade 1 tower passage (B1P) shows that the fastFoam FAST output similar to the experiment shows a strong reduction in torque if blade 1 passes.

That this occurs for fastFoam is interesting to notice as the tower effects were not enabled within FAST and no tower was modelled within the CFD mesh. For the experiment the torque reduction is much larger if blade 1 passes compared to the case when blade 2 passes the tower. This may be explained due to imbalances or drivetrain torsional effects.

For the thrust both fastFoam outputs show the same converging behavior. Whereas the values from the OpenFOAM postprocessing are slightly increased compared to the fastFoam FAST output both show approximately 1P fluctuations with lower amplitude if related to the experiment. It interesting to observe that in contrast to the torque for the thrust the phase of the fluctuations is relatively similar.

It would require further investigations what exactly causes the fluctuations in fastFoam, which are already present for the power, thrust and torque in the CFD solution, but amplified and phase shifted for the power and torque within the FAST ElastoDyn model.



(a) Low-speed shaft power.

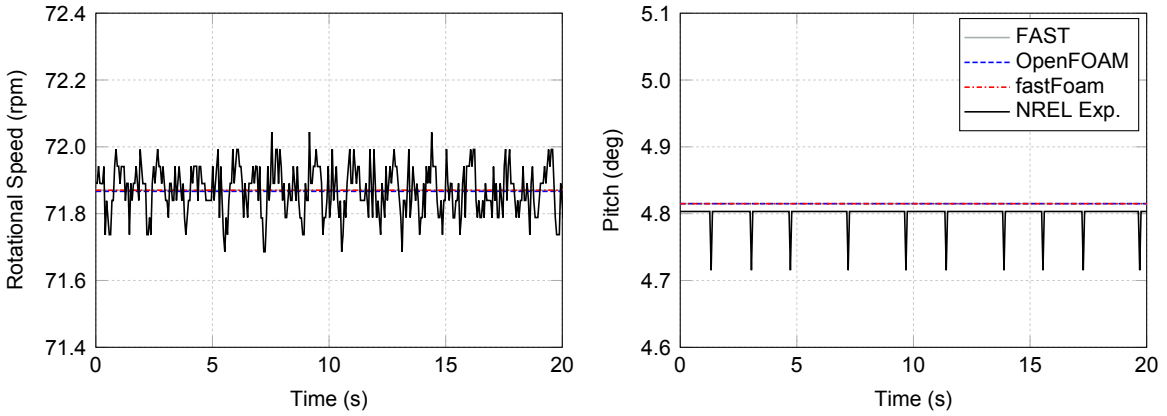
(b) Thrust.

(c) Low-speed shaft torque.

(d) Detailed torque variations at higher simulation times.

Figure 6.2: General turbine parameters according to simulations and experiment for case 1.

Another source of fluctuations for the experiment is due to variations in rotational speed and pitch angle for instance. As can be seen in Figure 6.3 the rotational speed varies quite a bit for the experiment, while for the simulations a constant value equivalent to the mean of the experiment is taken. For the pitch within all simulations for the NREL phase VI a pitch angle of 4.815 deg is taken (for case 3 this increases). However, the experimental data given by the digital blade pitch setting values indicate that this was slightly lower for the S0700000 experiment. Due to rounding errors and bearing in mind that these are only digital blade pitch settings, thus not measured, the chosen 4.815 deg seem to be appropriate. The pitch angle for the experiment also seems to undergo certain valleys, which may be related to errors in the dataset for instance due to rounding.

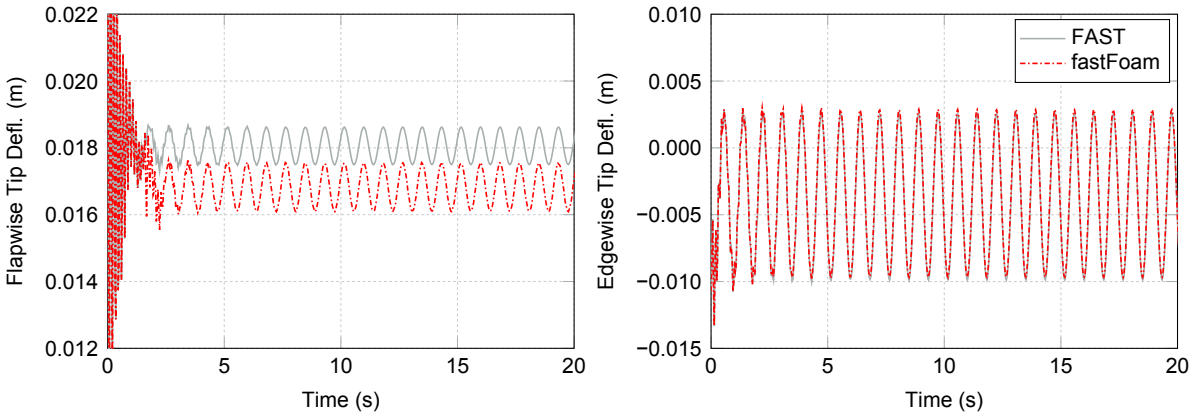


(a) Rotational speed of the rotor.

(b) Blade 1 pitch angle.

Figure 6.3: Controller parameters according to simulations and experiment for case 1.

In addition, the flapwise and edgewise tip displacements (positive towards the leading edge) are shown for the FAST and fastFoam simulation in Figure 6.4. For the experiment no data for these parameters is available and for the OpenFOAM simulations the displacements are constant zero, due to the assumption of rigid blades. It can be seen that both the flapwise and edgewise displacements follow a sinusoidal pattern. The magnitude is very small, which is due to the fact that the blades of the phase VI turbine are quite stiff. The agreement for the edgewise displacements between the different methods is excellent. This may be due to the reason that the edgewise displacements are driven by the gravitational forces. The agreement in amplitude for the flapwise displacements is quite good as well. However, the fastFoam simulations show a more than 5 percent decreased mean value, which could be related to a decrease in the aerodynamic forces as predicted by the CFD method versus BEM.

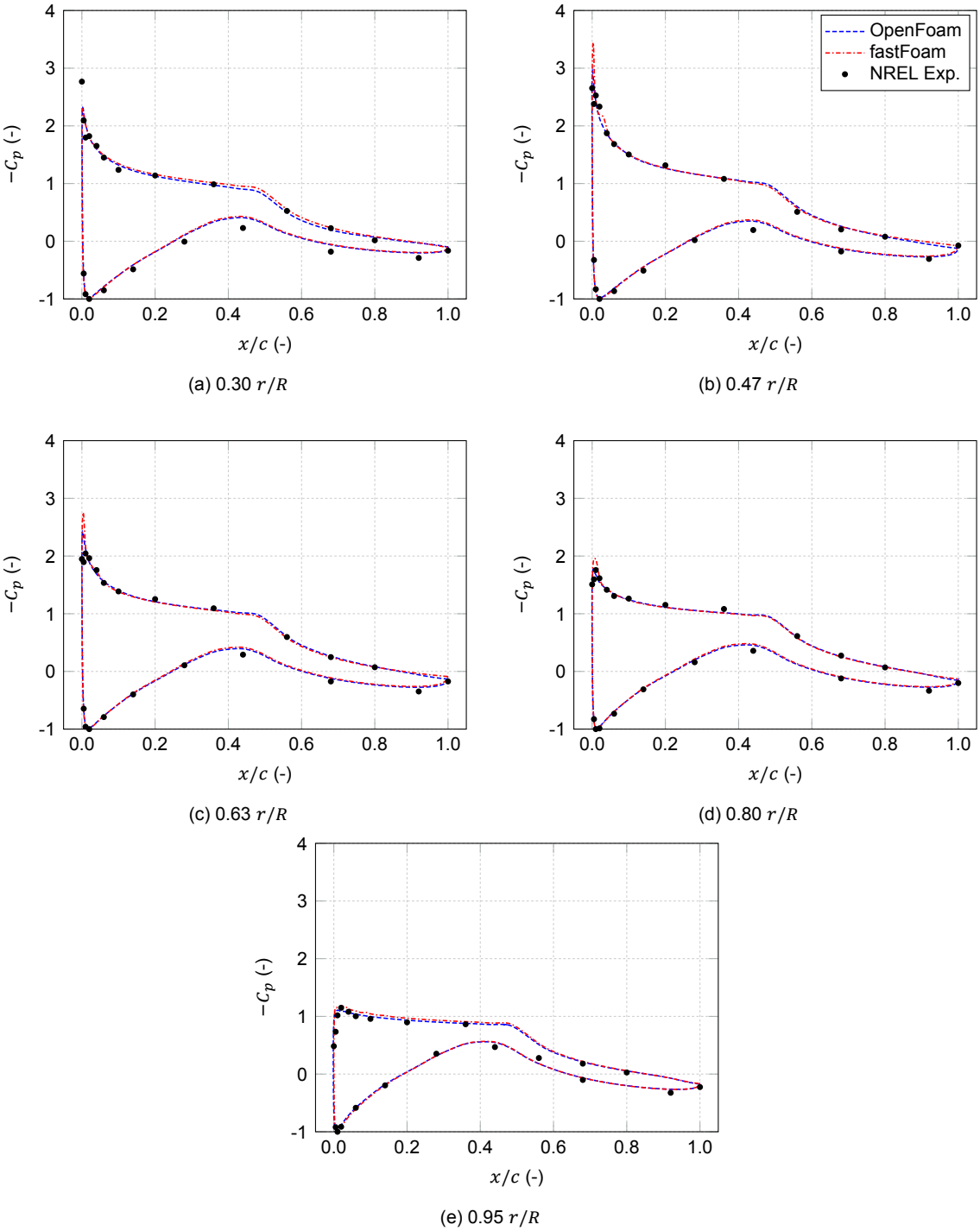


(a) Flapwise direction.

(b) Edgewise direction.

Figure 6.4: Blade 1 tip displacements according to simulations for case 1.

To investigate the aerodynamic forces along the blade span first the pressure distributions at several sections are considered and compared to the experimental data. It has to be noted that for the FAST simulations, where the BEM aerodynamic module is utilized no pressure distributions are obtained due to the nature of the method. As can be seen in Figure 6.5 the agreement for the pressure coefficient (C_p) between the CFD based methods and the experiment is excellent for the different sections along the blade. The difference between the OpenFOAM and fastFoam simulations is negligible for the pressure distributions. Small differences may occur as a result of numerical errors, small blade displacements and due to the reason that different post processing tools were used. Due to the small displacements it is expected that OpenFOAM and fastFoam show the same distribution, which is proven by the data.



(a) 0.30 r/R

(b) 0.47 r/R

(c) 0.63 r/R

(d) 0.80 r/R

(e) 0.95 r/R

Figure 6.5: Pressure distributions at different blade sections for case 1 (0 deg azimuth).

For the load comparison, first the thrust force coefficient C_{Th}, force in-plane coefficient C_{Tq} and pitching moment coefficient C_M are considered. They are plotted in Figure 6.6 at different sections along the blade. For the experimental data error bars are included indicating the maximum and minimum value for an azimuth band of about 10 deg around the selected plotting time, which is approximately similar to the plotting time of the simulations. Each data point for the experimental results corresponds to the data at this selected time, but due to the nature of measurements a value close to the minimum or maximum at that specific azimuth could be selected. This is the reason why error bars have been included in all force and moment distributions at specific times.

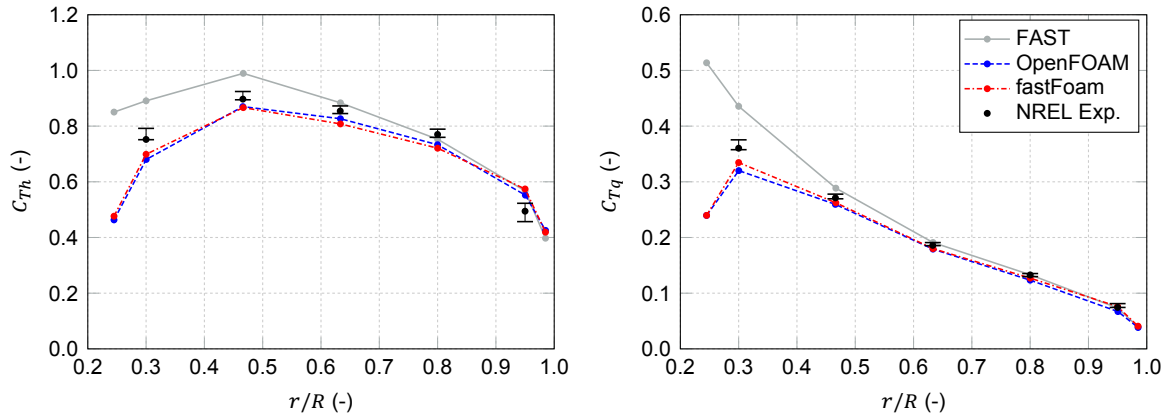
It can be seen that for the force coefficients the OpenFOAM and fastFoam results agree well with the experimental data. Also FAST shows good agreement especially outbords. However, at the inner

sections of the blade the FAST results seem to overpredict compared to the experimental results.

It is interesting to notice that at the first postprocessed section at about 0.25 r/R FAST predicts an increase for C_{Tq} compared to the next more outboard section. In contrast, both OpenFOAM and fastFoam predict a decrease at that section compared to the next more outboard section. Due to the reason that there is no experimental data at the first section (0.25 r/R), it cannot be stated safely which code is correct for this investigation.

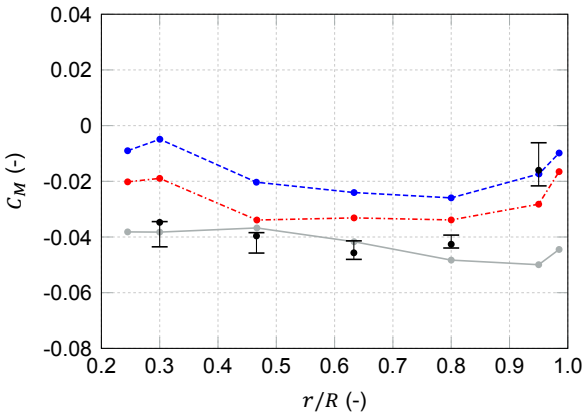
For the pitching moment coefficient C_M it can be stated that FAST shows the best agreement to the experimental data for the inner part of the blade. However, at the outer part especially at the tip the agreement gets worse and FAST overpredicts the magnitude of the moment. It can be seen that the OpenFOAM and fastFoam results deviate quite a bit, which is not expected when taking into account that the pitching moment is composed of the forces. This is especially questionable as the aerodynamic forces seem to agree well between OpenFOAM and fastFoam.

The reason for this may be related to different postprocessing tools used. For the OpenFOAM simulations, which were done at an early stage of the project, an initial postprocessing tool was used. This tool was later revised and the postprocessing was done differently allowing for runtime simulation processing of elastic blades. It seems that the new postprocessing tool used for fastFoam shows better prediction of the pitching moment coefficient. As this is the final tool the error in the OpenFOAM simulations is accepted knowing that one needs to rerun the entire simulations. One reason for this error could be that the moment arm and reference point are different. However, a first investigation did not discover this.



(a) Thrust coefficient.

(b) Torque coefficient.



(c) Pitching moment coefficient.

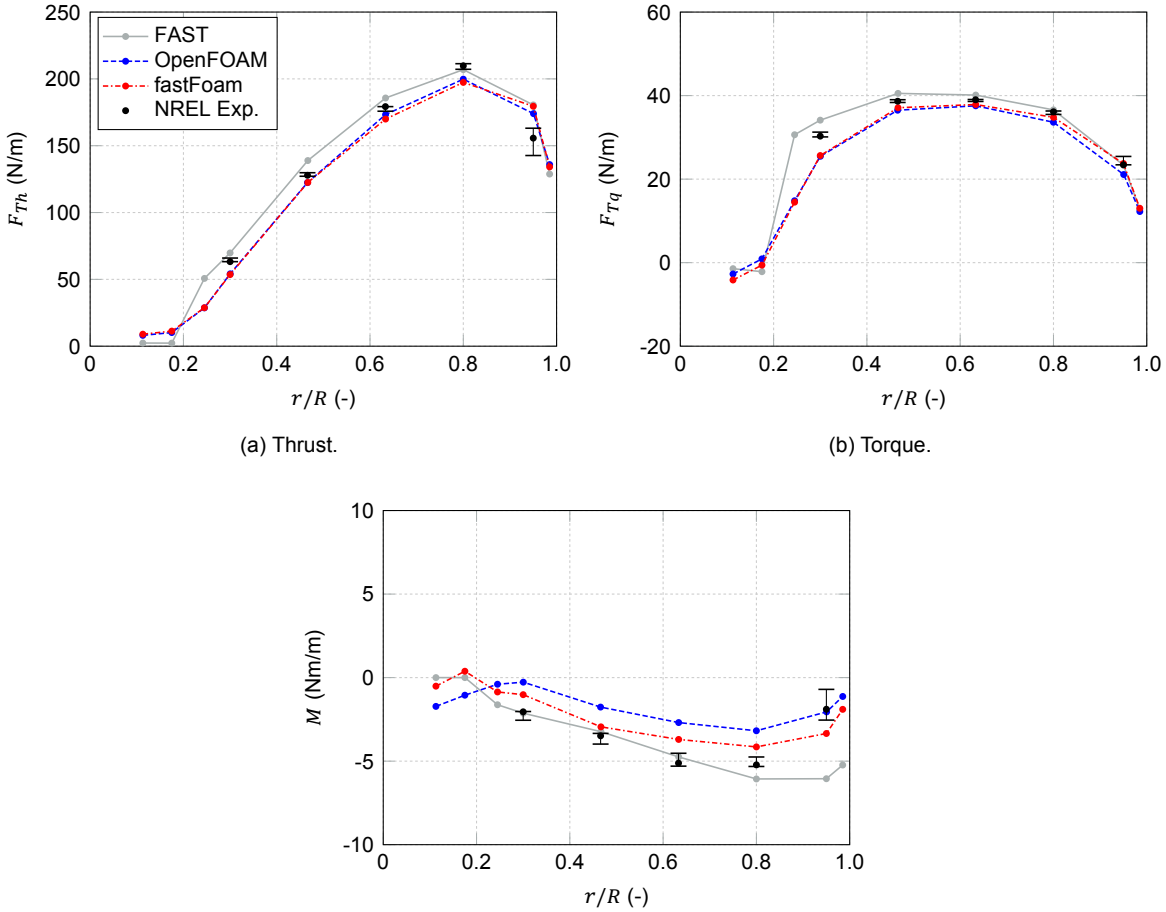
Figure 6.6: Force and moment coefficient azimuthal variations at different blade sections for case 1 (0 deg azimuth).

Additionally, for the comparison the absolute aerodynamic loads are considered. In Figure 6.7 the

thrust force F_{Th}, force contributing to the torque F_{Tq} and pitching moment M are shown. For the forces good agreement between the CFD methods is observed. The comparison with the experiment reveals that the OpenFOAM and fastFoam methods underpredict the forces slightly, except near the tip where a overprediction for the thrust force is shown. The BEM-based method FAST in contrast shows a slight overprediction along the entire blade for the forces. However, the absolute error to the experimental results is quite small for the three different methods. Only for the thrust force larger deviations are visible at the tip, where also a large fluctuation of the experimental data can be seen.

The CFD methods show a bit smoother load distribution for the 9 sections included. Especially, for the in-plane forces a sudden jump at the root is visible for the BEM results. This may be due to a change in the airfoil data used in BEM from a circular section at about 0.18 r/R to an airfoil shaped section (0.25 r/R).

Moreover, the pitching moments were considered. At the inboard part of the blade excellent agreement to the experiment is obtained with FAST, whereas fastFoam deviates slightly and OpenFOAM shows large deviations. At the tip fastFoam and OpenFOAM show better agreement, whereas FAST severely overpredicts the magnitude. Again the differences between the fastFoam and OpenFOAM simulations could be due to different postprocessing tools, where the revised tool used in fastFoam shows better results.



(a) Thrust.

(b) Torque.

(c) Pitching moment.

Figure 6.7: Force and moment distribution at different blade sections for case 1 (0 deg azimuth).

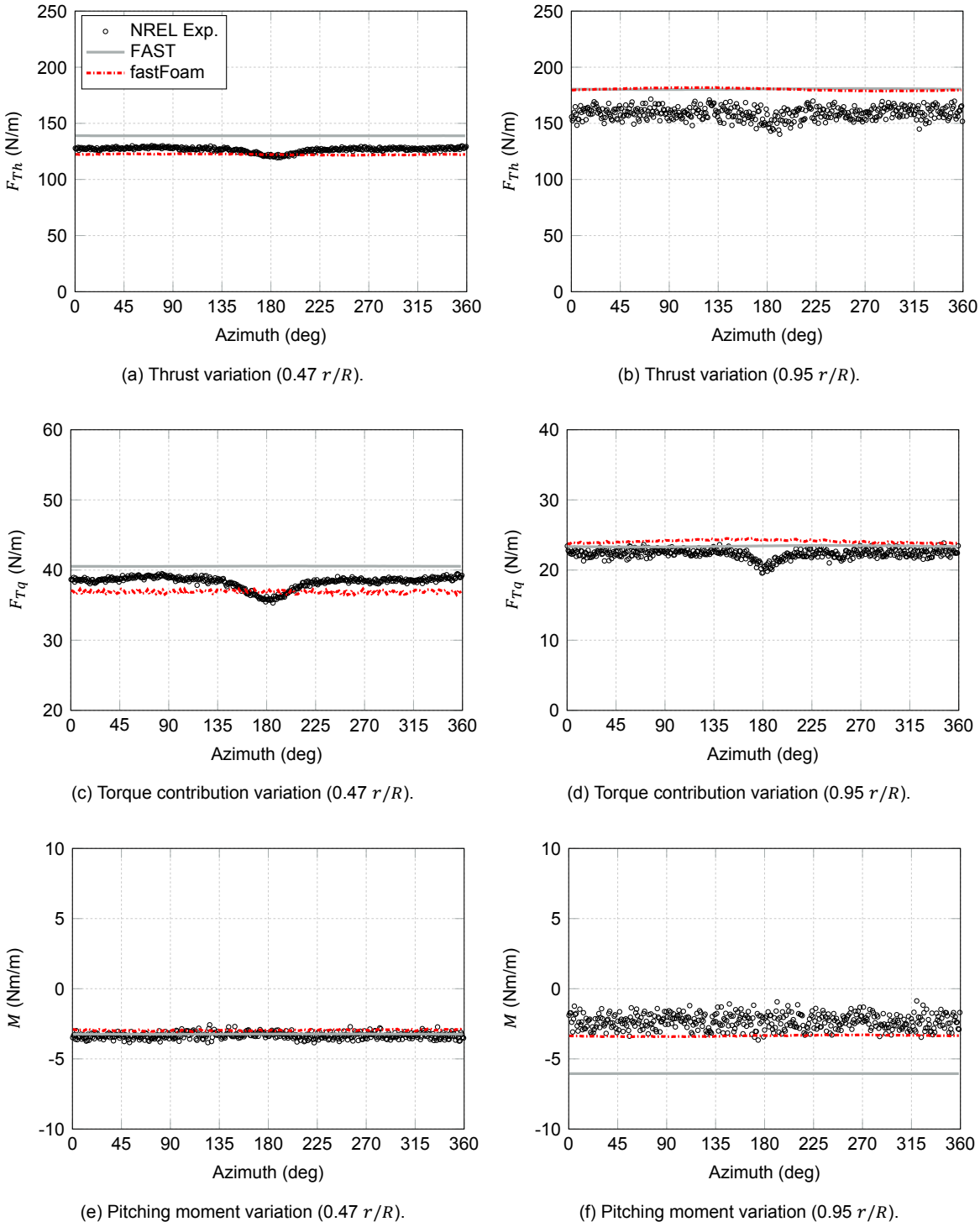
Finally, for the comparison of aerodynamic loads also the load variations over one rotor rotation are taken into account. Thus the loads are considered at azimuths other than zero compared to the previous shown load distributions, which were all done at zero azimuth. Therefore, the forces and pitching moment are shown in Figure 6.8 for an inboard (0.47 r/R) and near tip section (0.95 r/R). The

OpenFOAM results were not included for this case due to the reason that the initial postprocessing tool was used, which would require manual postprocessing of each azimuthal position and thus for 720 steps for the 0.5 deg azimuth steps. However, the aforementioned load comparison showed that the OpenFOAM results are close to the fastFoam results for the forces, whereas for the pitching moment there is a discrepancy.

For the inner section at 0.47 r/R it can be concluded, that for the thrust force fastFoam agrees very well over the whole range of azimuthal blade positions. FAST overpredicts the magnitude of the force slightly. Whereas the simulations do show a steady behaviour, thus no change of the values as a function of azimuth, the experiment shows a valley near the 180 deg blade position. This can be expected as a result of the tower effect at blade passage, which is not modelled by FAST and fastFoam. In FAST this could easily be activated by including the potential flow around the tower and the tower shadow effect. However, for fastFoam the tower would need to be included in the CFD mesh, drastically increasing the meshing effort. For a fair comparison between FAST and fastFoam it was decided to not include the tower in both of them, thereby reducing the CFD meshing effort. That the simulation results then show a relatively steady behaviour could be expected as there are no great sources of unsteadiness such as a tower, yawing, rotor cone or tilt.

Moreover, for the force contributing to torque at 0.47 r/R it can be observered that FAST shows some overprediction, whereas fastFoam underestimates the actual value from the experiment. The difference in magnitude is approximately similar for both simulations compared to the experimental data. Concerning the fastFoam results there are small fluctuations. For the experiment there is a huge decrease in force at blade passage of the order of 10 percent, which again is not predicted by the simulations due to the reason that no tower effects are included. For the moment both fastFoam and FAST show excellent agreement at he inner section.

For the outer section near the tip both fastFoam and FAST overestimate the thrust force magnitude by about 10 percent. The overprediction is lower for the torque contributing force. It is interesting to notice that whereas the FAST results are quite constant, the fastFoam results show a small increase at approximately 90 deg azimuth for the thrust forces and towards the 100 to 200 deg azimuths for the torque forces. It could be that the CFD results would require a bit longer simulation time to reduce these effects and achieve better convergence. Finally, for the pitching moment at the tip it can be concluded that FAST greatly overestimates the magnitude compared to the experimental data. It is about twice as large with a pitching moment towards feather of about -6 Nm/m. The fastFoam results seem to agree well for this section only showing a small deviation.



(a) Thrust variation (0.47 r/R).

(b) Thrust variation (0.95 r/R).

(c) Torque contribution variation (0.47 r/R).

(d) Torque contribution variation (0.95 r/R).

(e) Pitching moment variation (0.47 r/R).

(f) Pitching moment variation (0.95 r/R).

Figure 6.8: Force and moment variations at inner blade section (0.47 r/R, left) and outer blade section (0.95 r/R, right) for case 1.

6.1.3. Yaw Sweep

In order to investigate an increased unsteady condition a yaw sweep of 1 deg/s was simulated corresponding to case 2 in the simulation matrix in Table 3.6. The simulation results were again compared to experimental results now from measurement sequence S07YSU00. The unsteady yaw sweep was chosen as it allows for comparison at different yaw angles and deals as an increased difficulty validation test for the fastFoam solver, due to the changing yaw angle and thus rotational axis which needs

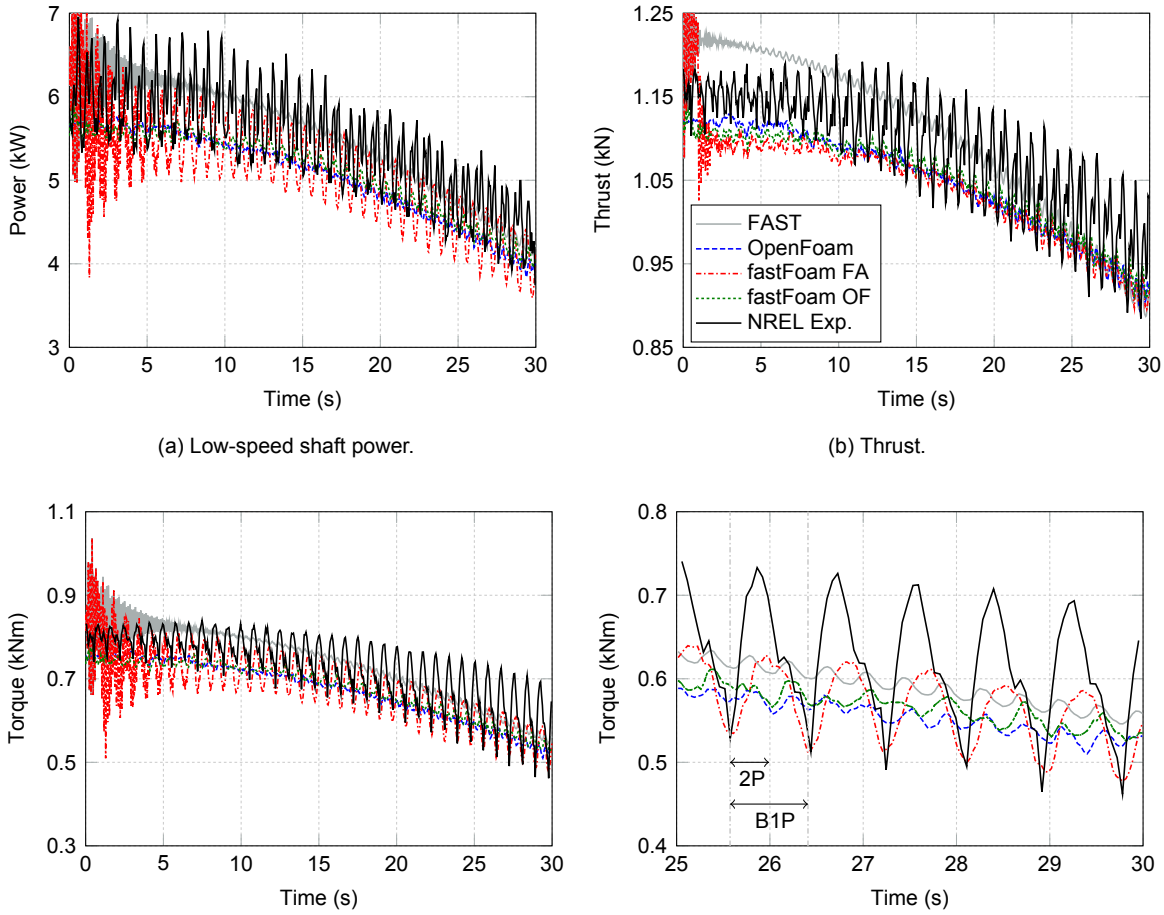
to be correctly implemented in the mesh updates.

The Low-speed shaft power, thrust and torque are given in Figure 6.9. It can be seen that with increasing yaw angle the mean of the values reduces, while the amplitude of the fluctuations due to the unsteady yaw conditions increases. For the power one can observe that whereas FAST and OpenFOAM simulations show only small fluctuations, the experimental results and the fastFoam simulations show larger ones. Again for the experimental power no clear pattern is observable, which may be due to imbalances of the rotor in the experiment and fluctuations in rotational speed. The mean power decrease is well predicted by FAST, while the mean power decrease of OpenFOAM and fastFoam show a slight underestimation.

For the thrust it can be seen that FAST cannot predict the decrease due to the yawing accurately. It overerstimates the reduction due to the yawing effect. OpenFOAM and fastFoam can accurately predict the reduction, however the mean thrust is slightly underestimated at all times. Finally, for the torque a smoother behaviour can be seen with more regular fluctuations in both the experiment and the simulations, although there are initial fluctuations for both FAST and fastFoam. By analyzing the yawing effect on the torque at higher yaw angles, see Figure 6.9d one can identify the effect of the nonuniform inflow angle.

From the Figure it an clearly be seen that the simulations as well as the experiment show a strong reduction in torque for the tower passage of blade 1, indicated as blade 1 passage (B1P). However, for the blade 2 tower passage, which occurs approximately at 26 seconds only FAST and OpenFOAM as well as the fastFoam OpenFOAM output result in a reduction. The experiment as well as the fastFoam simulations actually show an increase of the values. Thereby, it can be seen the torque for the FAST and OpenFOAM simulations varies with the blade passing frequencies (2P). Whereas the experiment and fastFoam FAST data show a 1P dependency.

Although the fastFoam FAST simulation seems to predict the experiment most accurately it is questionable what causes the 1P dependency. One reason could be due to shaft bending and rotor imbalances. However, in the fastFoam simulations in principal no strong imbalances should occur as the blades were modelled symmetrically. An influence could be the missing tower compared to the experiment, however in the OpenFOAM and FAST simulations the tower is missing as well.



(a) Low-speed shaft power.

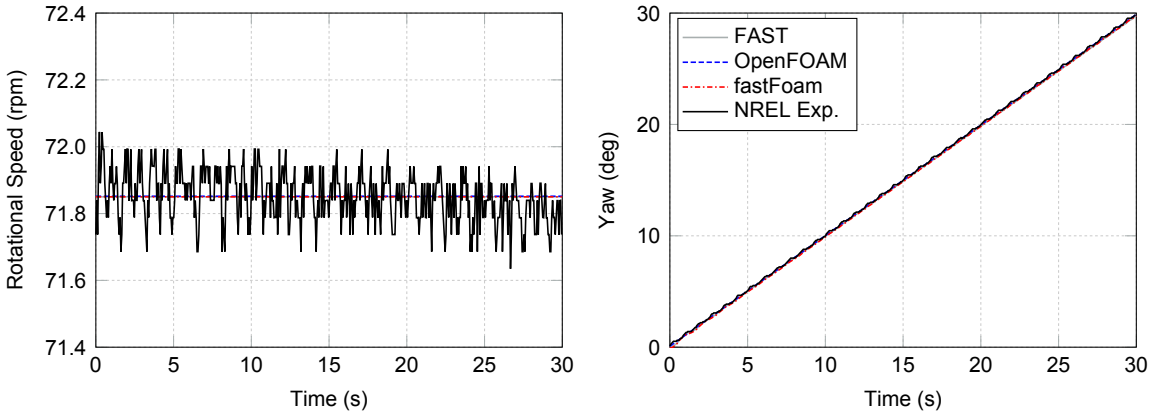
(b) Thrust.

(c) Low-speed shaft torque.

(d) Detailed torque variations at high yaw angles.

Figure 6.9: General turbine parameters according to simulations and experiment for case 2.

In Figure 6.10 the rotational speed and yaw angle are shown. It can be seen that for the experiment the rotational speed decreases slightly for larger yaw angles. In the simulations the rotational speed is kept constant and approximately equal to the mean of the experiment. This could result in small deviations in the azimuth angle for the simulations versus experiment especially increased at larger yaw angles. The yaw angle implemented in the simulations follows exactly the experimental data. The pitch angle although not shown here is chosen to be about the same as for the experiment with a value of 4.815 deg for the simulations. However, there are certain pitch reductions in the experimental data as previously shown for case 1 in Figure 6.3.



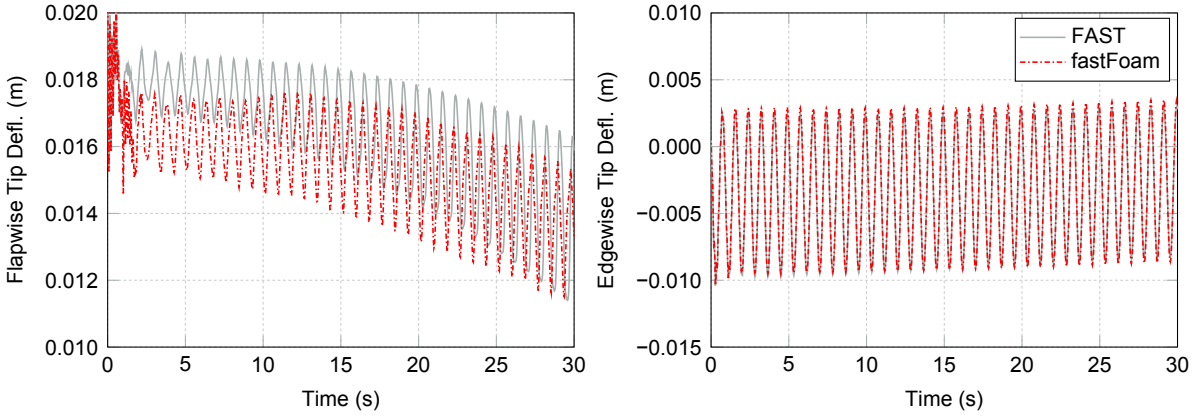
(a) Rotational speed of the rotor.

(b) Yaw angle.

Figure 6.10: Controller parameters according to simulations and experiment for case 2.

Next, the blade tip displacement evolution over time is given in Figure 6.11. It can clearly be seen that the mean of the flapwise tip displacement decreases with increasing yaw angle for both the fastFoam and FAST simulations. This is related to a decrease in aerodynamic loads due to the yawing. However, the fluctuations of the deflection is increasing for flapwise displacements. At the start of the simulation the flapwise deflections in fastFoam are equal to the calculated values from FAST. However, at about 1.6 seconds corresponding to two rotor revolutions the CFD loads are applied and the flapwise deflection reduces compared to FAST. At large yaw angles the results from FAST show a larger mean and amplitude of the fluctuation.

In comparison to the flapwise tip displacement, the edgewise one shows very similar results for both FAST and fastFoam. The mean of the edgewise tip deflection slightly increases over time, while the amplitude stays constant. The reason for this is that edgewise displacements for this turbine are mostly driven by the gravitational loads, which are constant and independent of the yaw angle.



(a) Flapwise direction.

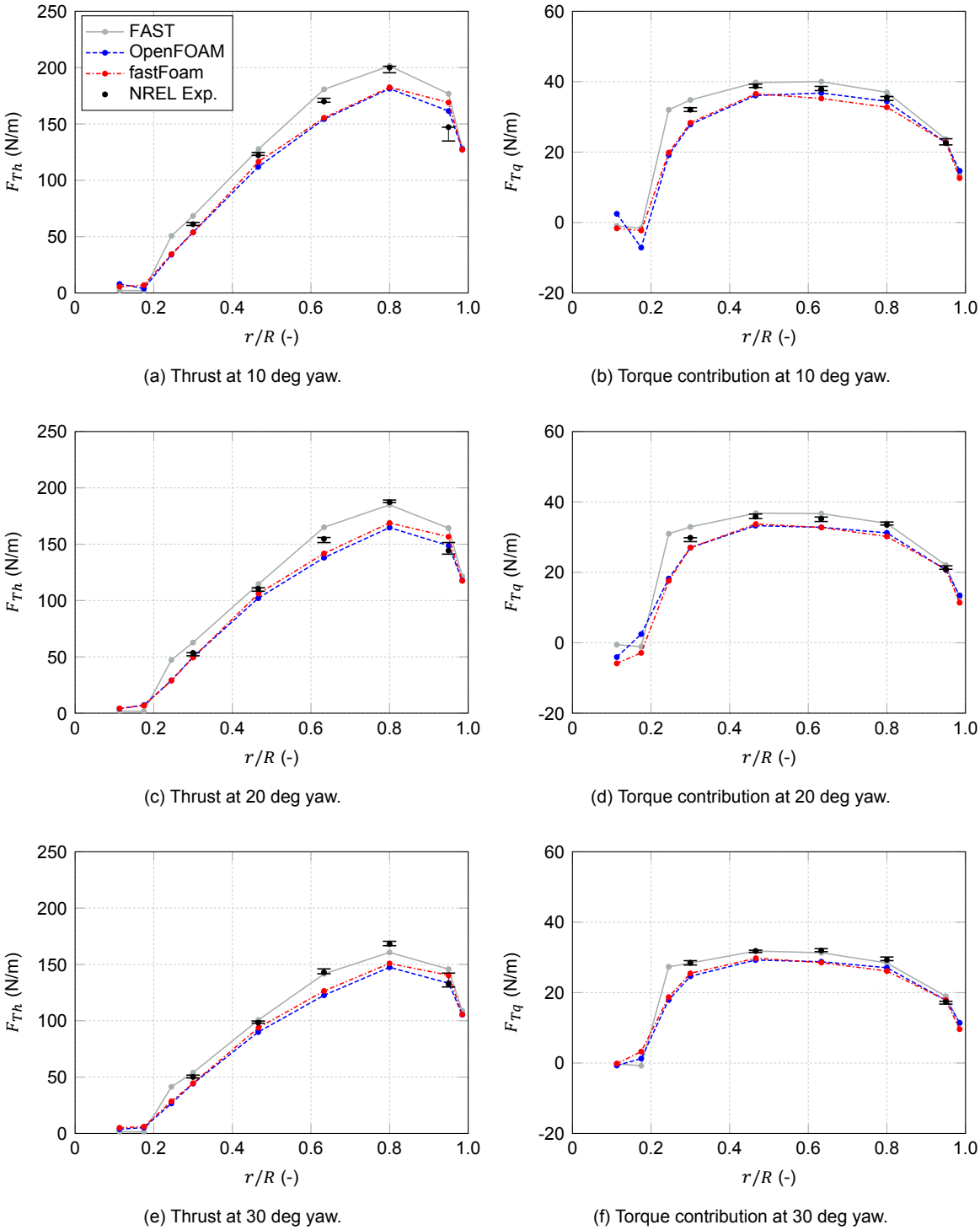
(b) Edgewise direction.

Figure 6.11: Blade 1 tip displacements according to simulations for case 2.

To investigate the effect of the increasing yaw angle, the aerodynamic forces at several sections and different yaw angles are given in Figure 6.12 at an azimuth of 0 deg. For the forces related to the thrust one can observe that FAST shows a better agreement with the experimental results at the outboard sections between 60 and 80 percent of the blade span. However, near the tip (0.95 r/R) and near the root (0.3 r/R) the CFD results are more accurate. Both CFD methods agree well which is expected due to the low magnitude of displacements. A difference is only observable near the tip at at 0.95 r/R, where the OpenFOAM results seem to be closer to the experimental values. Overall it can

be observed that the thrust forces greatly reduce from about 5 to 10 to 20 percent for the respective yaw angles of 10, 20 and 30 deg.

For the forces which contribute to the torque a similar reduction can be observed. These forces are overpredicted slightly by FAST at lower yaw angles, whereas the CFD methods underestimate them. However, at the largest yaw angle of 30 deg the FAST simulations show really good agreement for this selected azimuth angle which was unexpected.



(a) Thrust at 10 deg yaw.

(b) Torque contribution at 10 deg yaw.

(c) Thrust at 20 deg yaw.

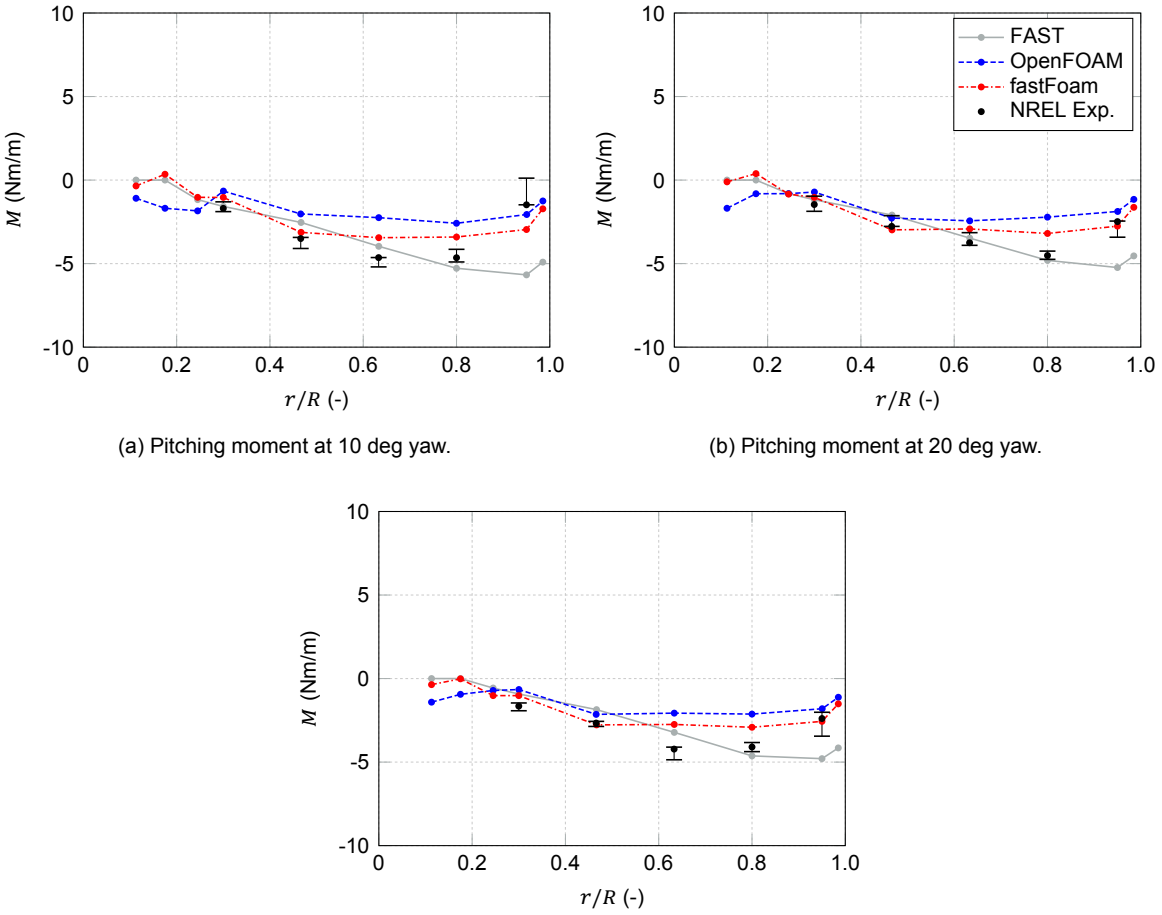
(d) Torque contribution at 20 deg yaw.

(e) Thrust at 30 deg yaw.

(f) Torque contribution at 30 deg yaw.

Figure 6.12: Force distribution at different blade sections and yaw angles for case 2 (0 deg azimuth).

Additionally in Figure 6.13 the pitching moment distribution along the blade at zero azimuth is given for the different yaw angles analyzed. Again the aforementioned difference between both fastFoam and OpenFOAM results is present. The CFD results show better agreement with the experiment at the tip compared to the FAST results. However, the FAST results do agree well near the outboard section between 60 to 80 percent of blade span. The fastFoam results show the most accurate results for the 0.48 r/R section over the whole sets of yaw angles.



(a) Pitching moment at 10 deg yaw.

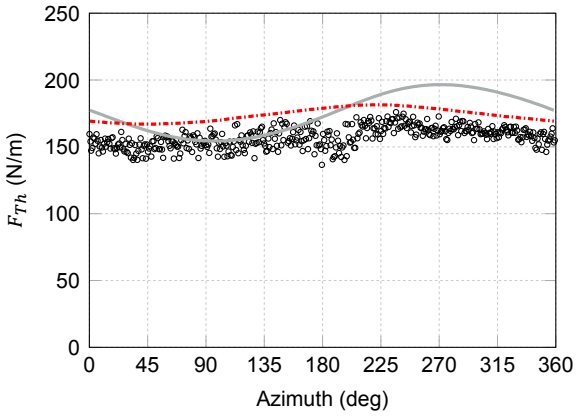
(b) Pitching moment at 20 deg yaw.

(c) Pitching moment at 30 deg yaw.

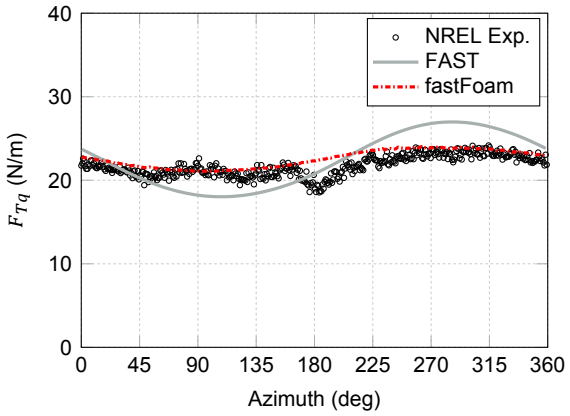
Figure 6.13: Moment distribution at different blade sections and yaw angles for case 2 (0 deg azimuth).

Finally, it is investigated how the loads change over one revolution at unsteady yaw conditions. Therefore, the forces are shown near the tip at the 0.95 r/R section, where most of the power is generated, for varying yaw angles in Figure 6.14. For the thrust forces shown on the left it can be observed that the CFD method is slightly overpredicting at the entire range of azimuths. However, the shape of the fluctuations is accurately predicted with peaks at about 225 deg. In contrast, the FAST simulations, using BEM, fail to predict the shape of the azimuthal variations. Especially at large yaw angles near the 90 and 270 deg azimuths large deviations from the experimental data are present.

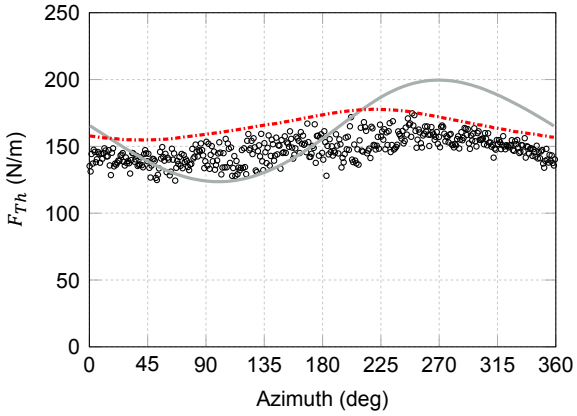
For the torque contributing forces the fastFaom method shows excellent agreement near the tip at all yaw angles. Only at 180 deg azimuth the force reduction due to the tower is not predicted. In contrast, FAST also shows a sinusoidal variation, but is again overestimating its magnitude especially near 90 and 270 deg azimuth. At the largest yaw angle of 30 deg its maximum deviation to the experimental data gets really large with up to 40 percent.



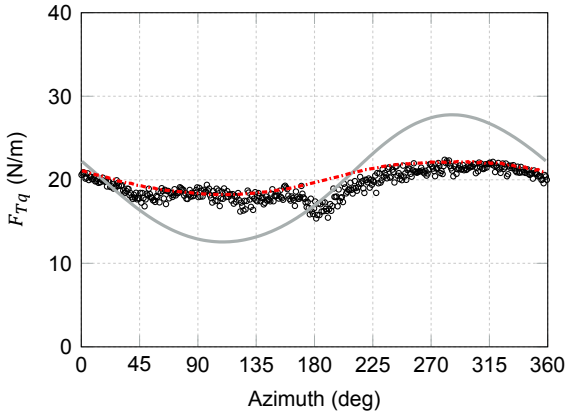
(a) Thrust variation at about 10 deg yaw.



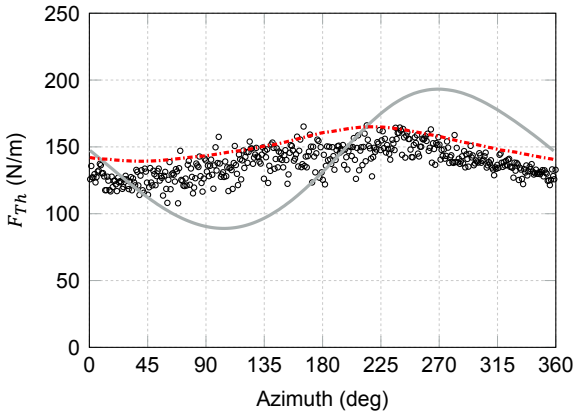
(b) Torque contribution variation at about 10 deg yaw.



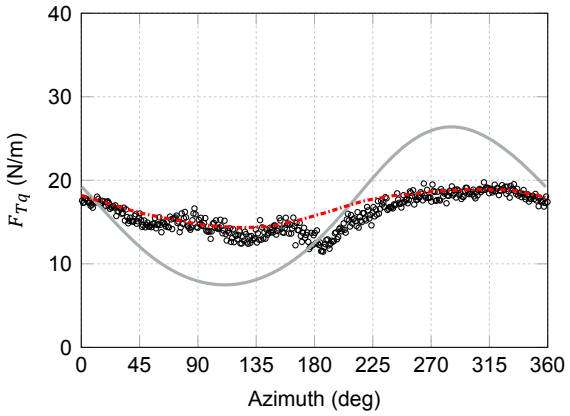
(c) Thrust variation at about 20 deg yaw.



(d) Torque contribution variation at about 20 deg yaw.



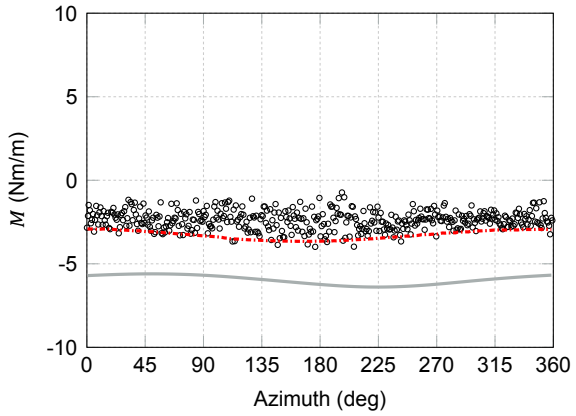
(e) Thrust variation at about 30 deg yaw.



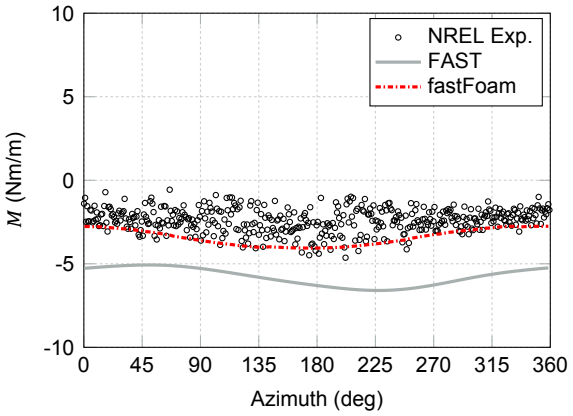
(f) Torque contribution variation at about 30 deg yaw.

Figure 6.14: Force azimuthal variations with approximately 10, 20 and 30 deg yaw angle at outer blade section (0.95 r/R) for case 2.

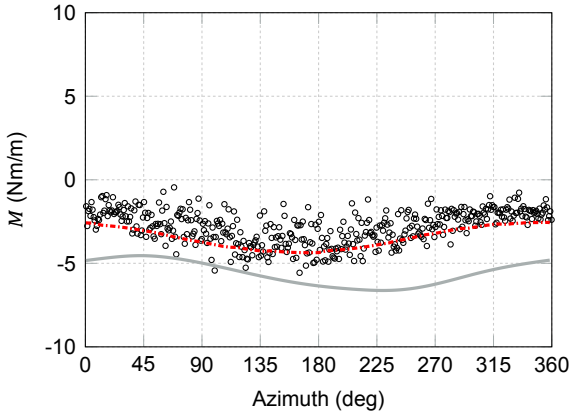
In addition to the forces, also the moment variations near the tip due to the unsteady yaw conditions are shown in Figure 6.15. It can be sen that the fastFoam results show good agreement to the the experimental data with a slight overprediction of the mean. For FAST there is a much larger overestimation and for large yaw angles the shape of the moment variations is not accurately predicted. The minimum value for the moment in the FAST simulations is located at an azimuth of about 225 deg, whereas in the fastFoam and experimental results it is located around the 180 deg azimuth.



(a) Pitching moment variation at about 10 deg yaw.



(b) Pitching moment variation at about 20 deg yaw.

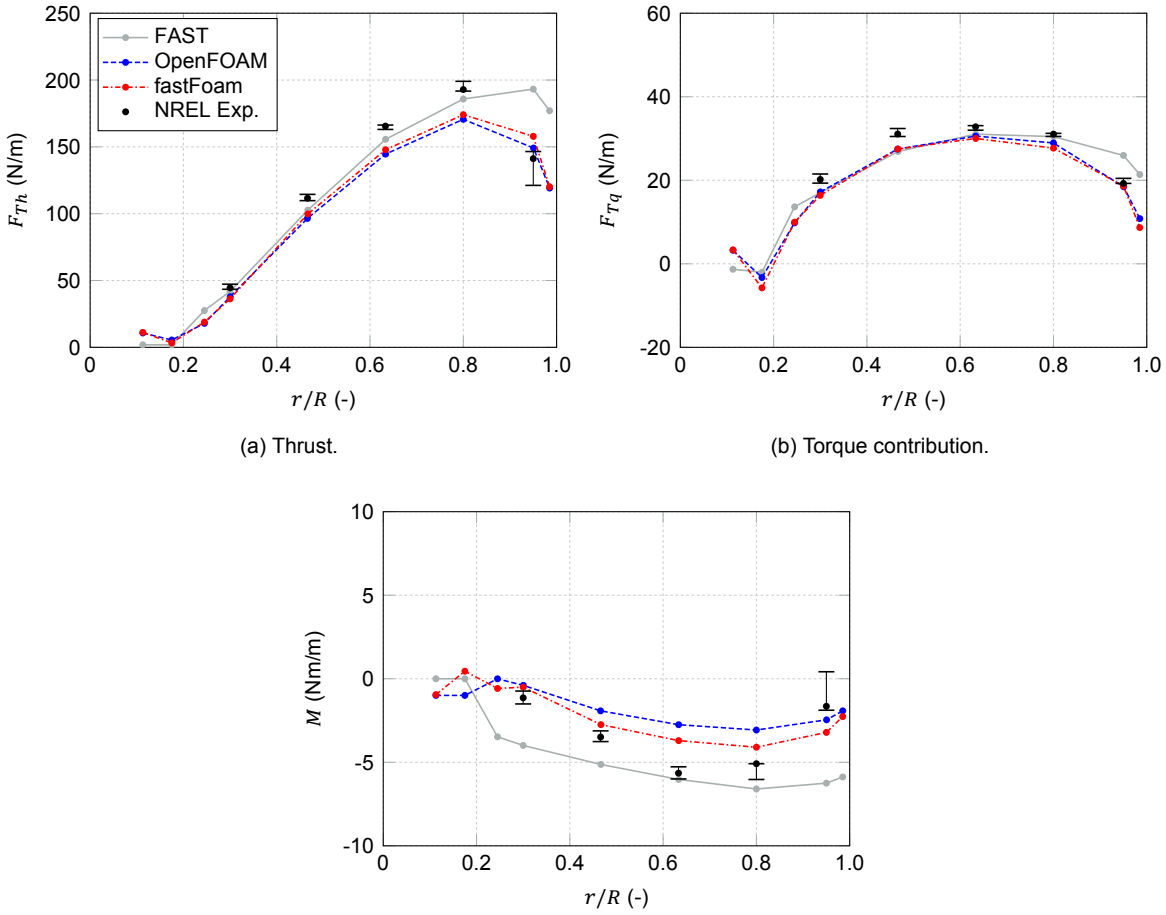


(c) Pitching moment variation at about 30 deg yaw.

Figure 6.15: Moment azimuthal variations with approximately 10, 20 and 30 deg yaw angle at outer blade section (0.95 r/R) for case 2.

Finally, it was decided to plot the force and moment distribution again for an azimuth of 270 deg shown in Figure 6.16. This was done as at this specific azimuth the deviation to the experimental data was the largest for FAST near the tip, see Figure 6.14 and 6.15. As can be seen in Figure 6.16 FAST shows good agreement for the forces up to 80 percent of the blade span. The CFD method results also agree well, but from about 60 percent blade span their deviation to the experimental results gets larger. However, near the tip FAST clearly overestimates the forces up to 40 percent at the 0.95 r/R section. The CFD methods show a better agreement at the tip for the forces.

For the pitching moment at 270 deg azimuth the same deviation between both CFD methods is present. The fastFoam method shows the best agreement at the inner blade part, where FAST clearly overestimates the magnitude of the pitching moment. At the middle section from 60 to 80 percent FAST shows a better agreement, whereas at the tip the same behaviour, such as already shown for the forces, is present in FAST resulting in a clear overprediction.



(a) Thrust.

(b) Torque contribution.

(c) Pitching moment.

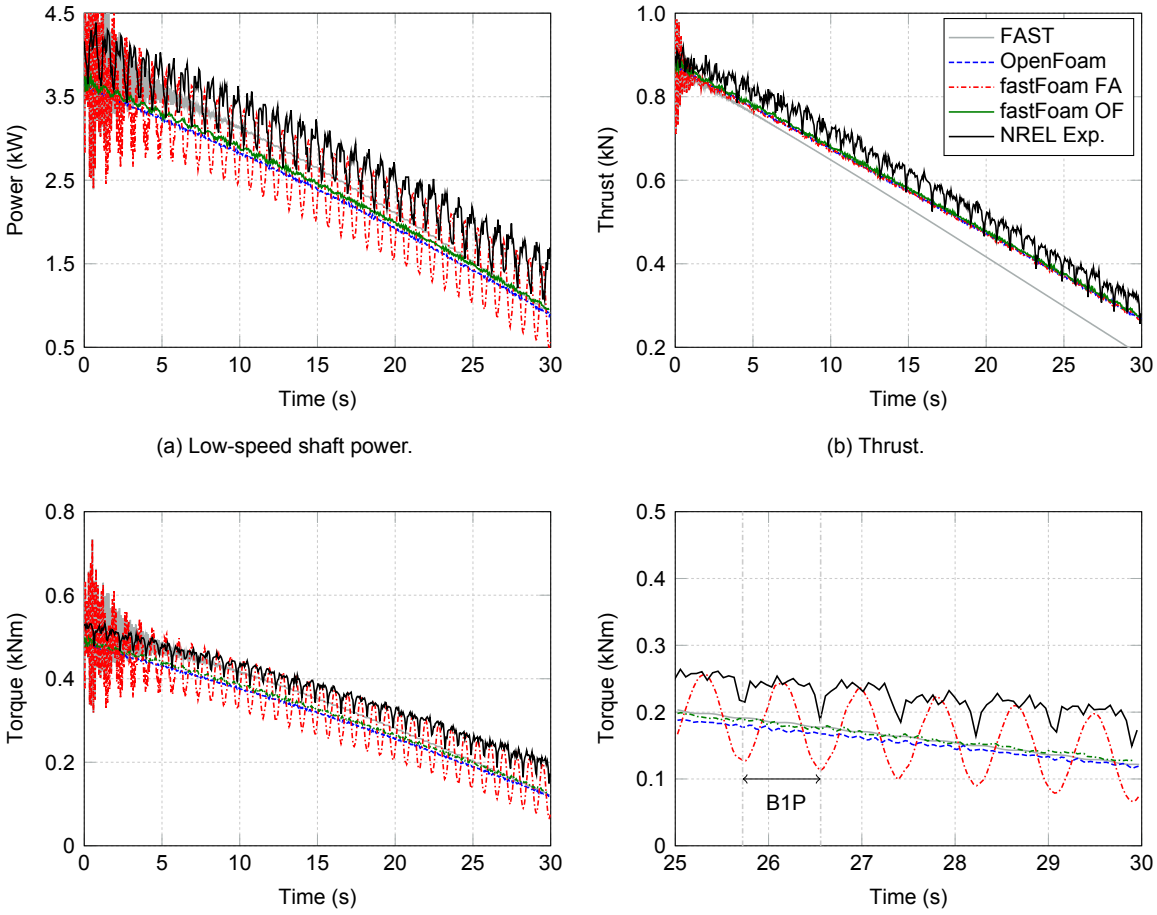
Figure 6.16: Force and moment distribution at different blade sections at approximately 30 deg yaw and 270 deg azimuth for case 2.

6.1.4. Pitch Slope

In addition to the normal operation and yaw cases, a last 30 s case for the phase VI turbine was examined based on a pitching motion, where the pitch angle is linearly increased. The wind speed for this case was approximately 6 m/s, thus lower compared to the previous cases. This was done as no experimental data for an increasing pitch case is available for the 7 m/s wind speed. It also allows to investigate the agreement of the simulations to the experiment at a different wind speed.

The general parameters such as thrust, torque and power are given in Figure 6.17 for the simulations and the experiment. It can be observed that for the thrust a strong linear reduction is present for the different methods. Whereas the CFD methods are able to predict the linear slope of the thrust reduction, FAST clearly underestimates the thrust at higher pitch angles. For the power and torque the CFD methods show an approximately constant underestimation of the mean power values from the experiment. FAST shows good agreement at low pitch angles, but also underestimates these parameters at higher pitch angles. Again the slope of the reduction is not met by FAST and it seems that at even larger pitch angles than the maximum simulated one of 11, the difference to the experiment gets worse.

Again sinusoidal fluctuations in power and torque can be observed for the fastFoam FAST output. The reason for this cannot be immediately explained and would require further investigations, but the mean value agrees well with the fastFoam OpenFOAM postprocessing outputs.



(a) Low-speed shaft power.

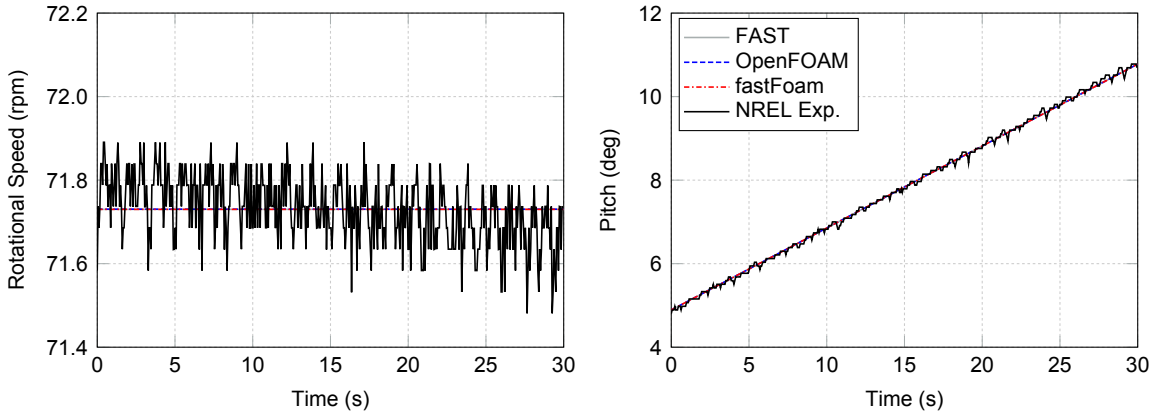
(b) Thrust.

(c) Low-speed shaft torque.

(d) Detailed torque variations at large pitch angles.

Figure 6.17: General turbine parameters according to simulations and experiment for case 3.

Next, the control parameters such as the rotational speed and the pitch angle are taken into account in Figure 6.19. For the rotational speed the mean experimental value is approximately matched by the simulations, where the error increases for large pitch angles. The pitch angle increase also seems to agree well with the experimental slope, but for the experiment some fluctuations can be observed which are not included in the simulations.

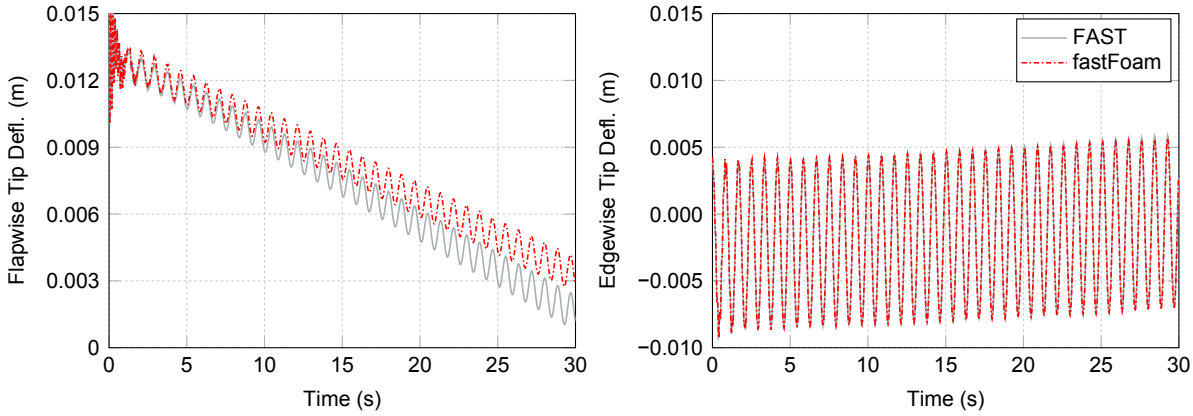


(a) Rotational speed of the rotor.

(b) Pitch angle.

Figure 6.18: Controller parameters according to simulations and experiment for case 3.

Moreover, the tip deflection evolution over time is given in Figure 6.19 for both flap- and edgewise directions. While for the edgewise deflections no difference is observable between FAST and fastFoam, for flapwise deflections the different methods can be clearly distinguished. Compared to FAST, fastFoam results show a small increase in the deflections at higher pitch angles, which may be related to increased loads for fastFoam compared to FAST.



(a) Flapwise direction.

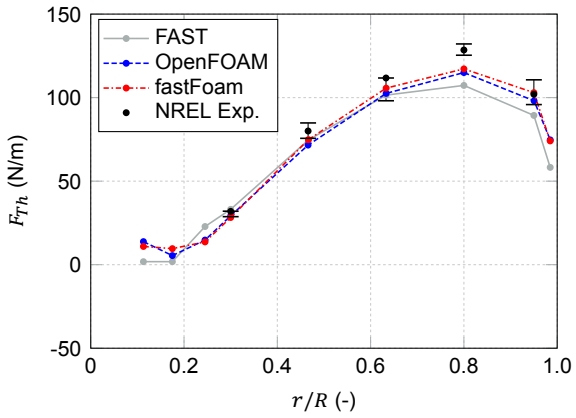
(b) Edgewise direction.

Figure 6.19: Blade 1 tip displacements according to simulations for case 3.

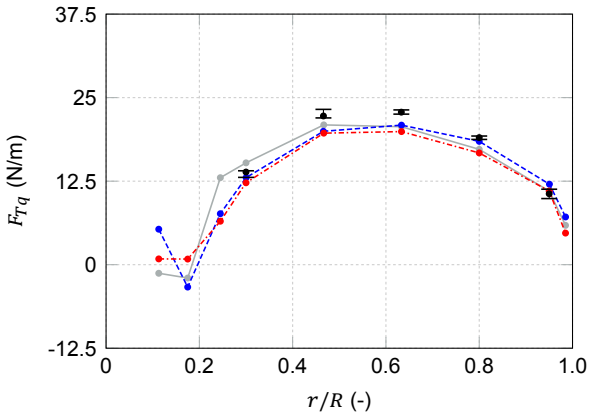
To investigate the loads, the thrust and torque contributing forces are shown in Figure 6.20. It can be seen that for the thrust forces the loads seem to be better predicted by the CFD methods. FAST shows a clear underestimation near the outer part of the blade, which may be attributed to a deficiency of the BEM unsteady model. All methods show relatively good agreement at the inner blade part for the thrust forces. In the outer region, especially at high pitch angles, the agreement gets worse, but the CFD methods are clearly superior in this aspect. Both CFD results agree well with slightly increased results for the fastFoam predicted thrust forces.

For the forces contributing to the torque forces all methods agree relatively well with a tendency to underestimate the measured forces at higher pitch angles. At the inboard part FAST shows again increased forces compared to the CFD results. At the root the result varies quite a bit, which may be attributed to the unsteady flow situation at the cylindrical sections, where aerodynamic phenomena such as vortex shedding are occuring. It is interesting to observe that the disagreement between OpenFOAM and fastFoam results seem to increase at the outboard part of the blade. This could be again a reason of the different postprocessing tools used within OpenFOAM to obtain these results.

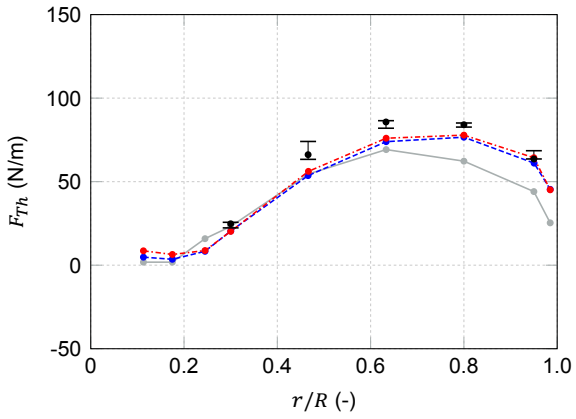
Whereas in OpenFOAM simulations for this turbine an early postprocesing tool was used based on rigid blades, for the fastFoam simulations a runtime processing was utilized taking into account the deflected blades.



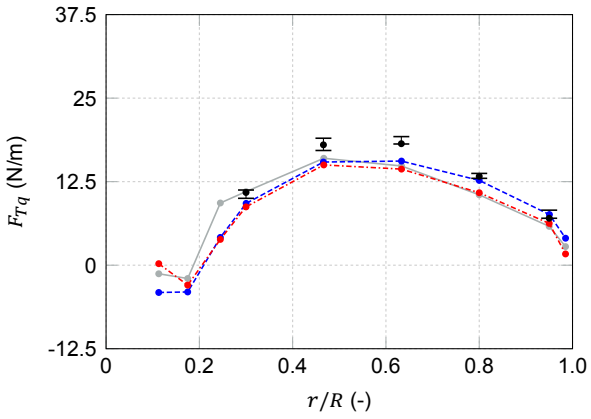
(a) Thrust at 6.9 deg pitch.



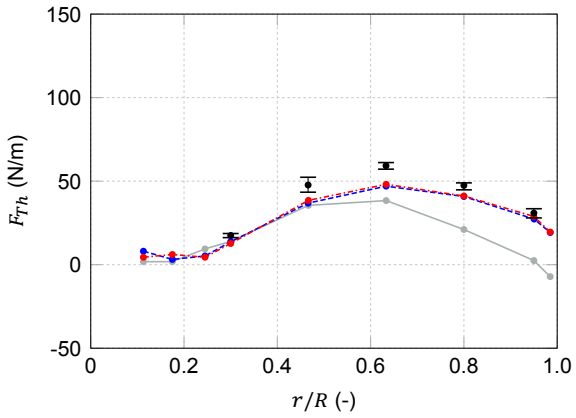
(b) Torque contribution at 6.9 deg pitch.



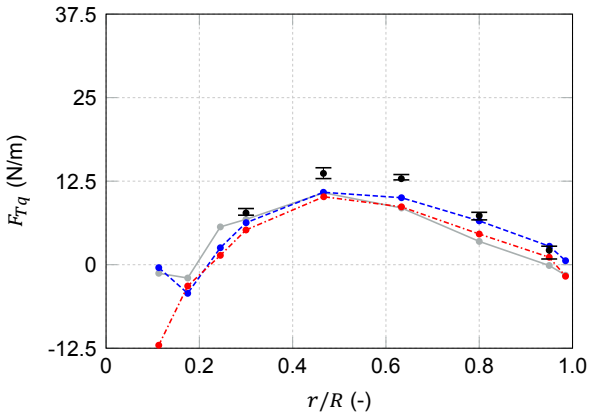
(c) Thrust at 8.9 deg pitch.



(d) Torque contribution at 8.9 deg pitch.



(e) Thrust at 10.7 deg pitch.

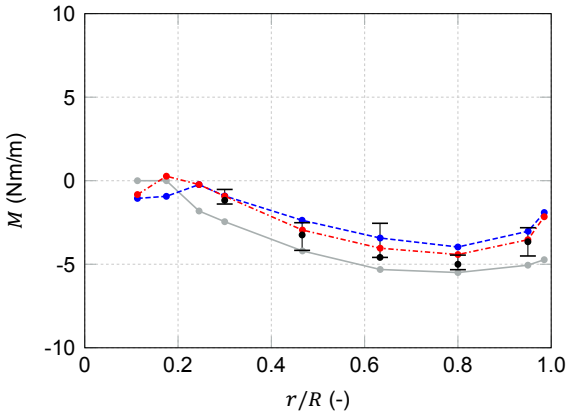


(f) Torque contribution at 10.7 deg pitch.

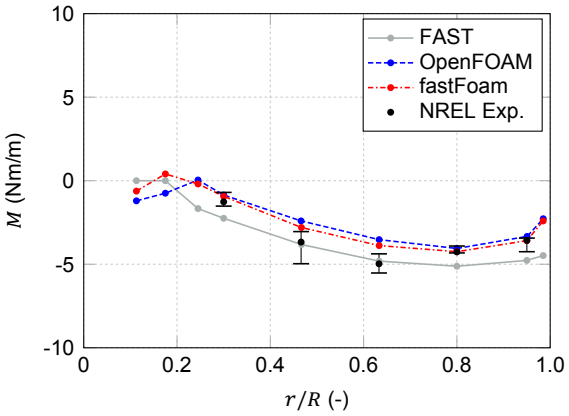
Figure 6.20: Force distribution at different blade sections and pitch angles for case 3 (0 deg azimuth).

In addition, also the aerodynamic pitching moment was considered in Figure 6.21. It can be seen that CFD results and especially fastFoam agrees well in the outer and inner blade regions. Whereas in the central blade region, from 50 to 65 percent blade span, FAST shows an improved result compared

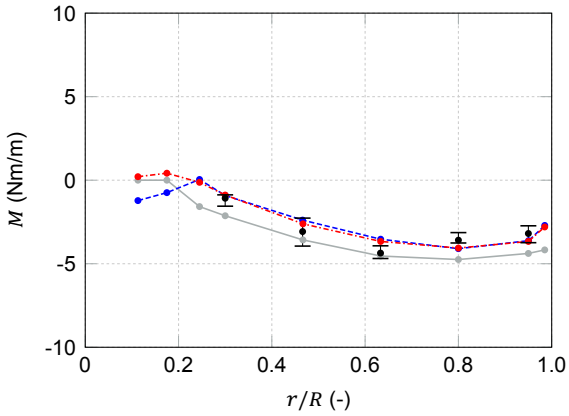
to experimental data at large pitch angles. The magnitude of the pitching moment predicted by FAST is increased compared to CFD methods except at the blade root.



(a) Pitching moment at 6.9 deg pitch.



(b) Pitching moment at 8.9 deg pitch.

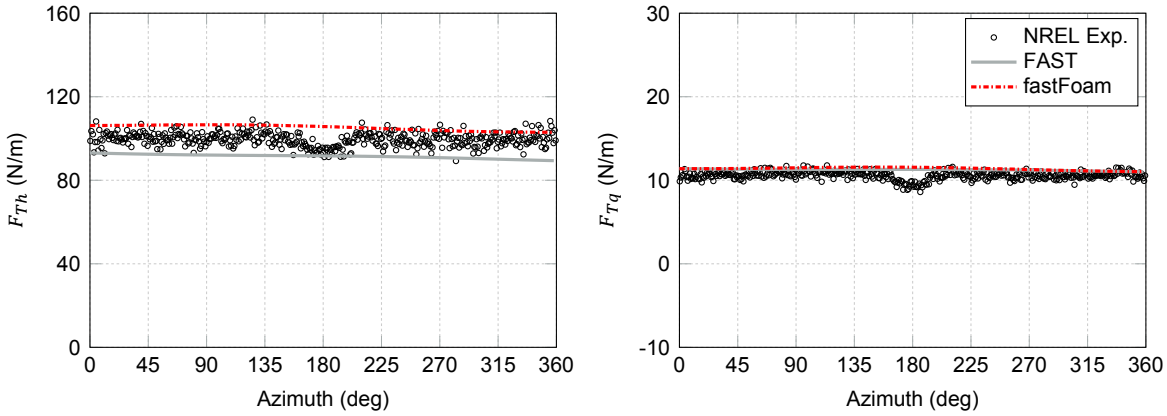


(c) Pitching moment at 10.7 deg pitch.

Figure 6.21: Moment distribution at different blade sections and yaw angles for case 3 (0 deg azimuth).

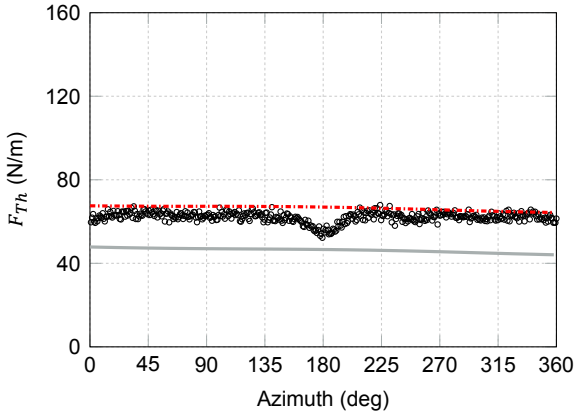
Moreover, the variations in the forces over one rotation while undergoing an increase in pitch angle are investigated in Figure 6.22. It can be sen that for the thrust forces a clear reduction from 0 deg to 360 deg azimuth over one rotation is present, related to a small increase in pitch angle (corresponding to about 0.16 deg difference in pitch angle). The variation is also present in the experiment, but due to the measurement fluctuations not as clear. The tower effect is observable for the experiment, but again not included in the simulations. Near the tip, the deficiency of FAST and thus BEM to predict the thrust force accurately at large pitch angles is obvious.

For the forces related to the torque also a small decrease is present over azimuth, but much smaller in magnitude. Both FAST and fastFoam seem to agree well with the experimental data near the tip.

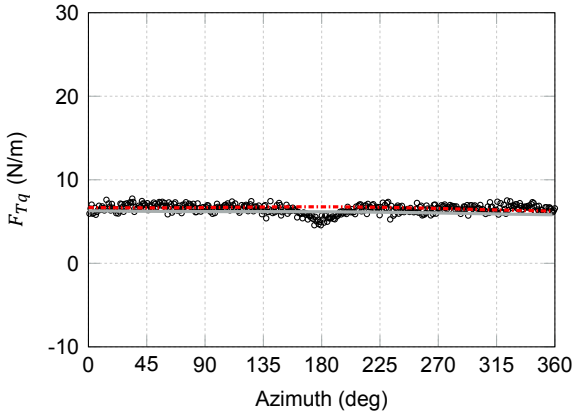


(a) Thrust variation at about 6.9 deg pitch.

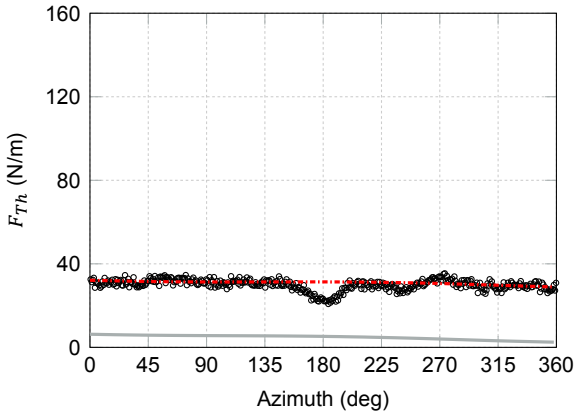
(b) Torque contribution variation at about 6.9 deg pitch.



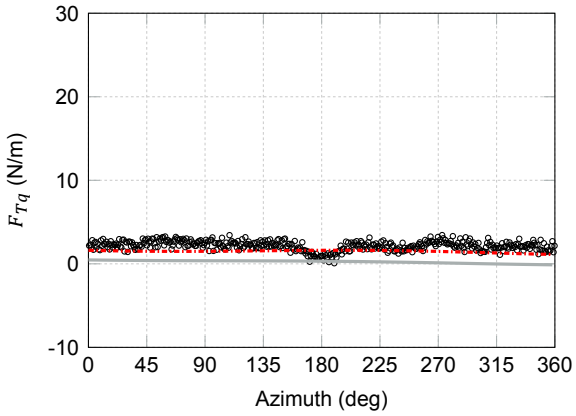
(c) Thrust variation at about 8.9 deg pitch.



(d) Torque contribution variation at about 8.9 deg pitch.



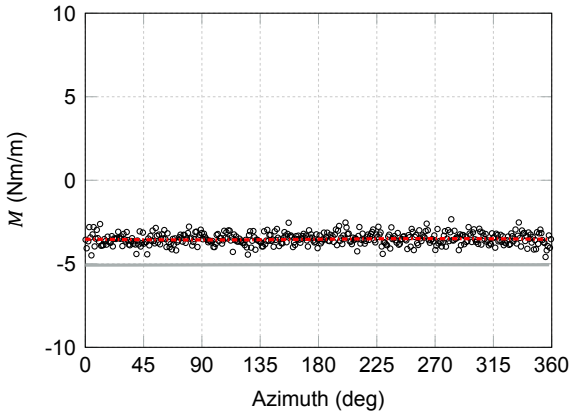
(e) Thrust variation at about 10.7 deg pitch.



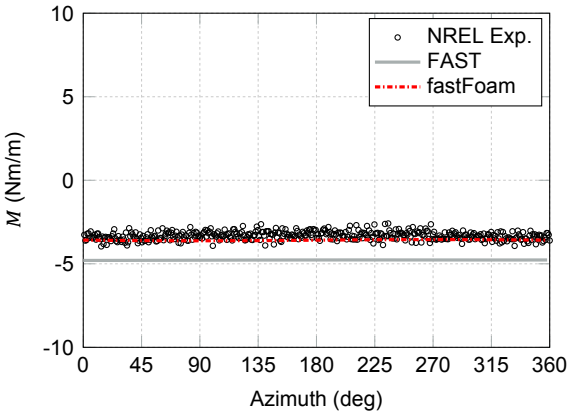
(f) Torque contribution variation at about 10.7 deg pitch.

Figure 6.22: Force azimuthal variations with approximately 6.9, 8.9 and 10.7 deg pitch angle at outer blade section (0.95 r/R) for case 3.

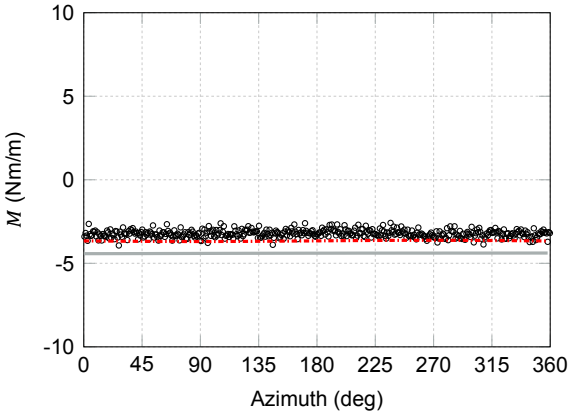
Finally, also the pitching moment azimuth variations near the tip are shown in Figure 6.23c. An overestimation of FAST is clearly present, whereas fastFoam results show excellent agreement at all different pitch angles analyzed. It can be investigated that compared to the forces the pitching moment is not that greatly influenced by the increase in pitch angle.



(a) Pitching moment variation at about 6.9 deg pitch.



(b) Pitching moment variation at about 8.9 deg pitch.



(c) Pitching moment variation at about 10.7 deg pitch.

Figure 6.23: Moment azimuthal variations with approximately 6.9, 8.9 and 10.7 deg pitch angle at outer blade section (0.95 r/R) for case 3.

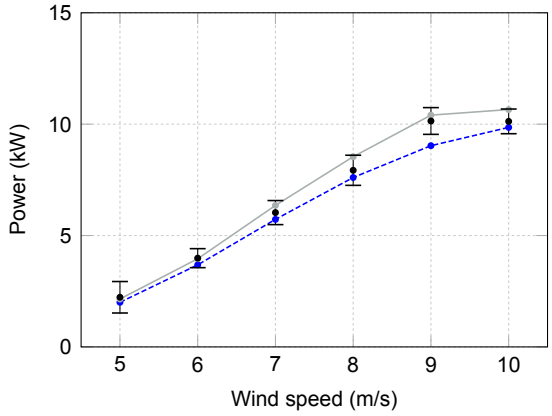
6.1.5. Power Curve

In addition to the aforementioned cases, a partial power curve was simulated. This was done in order to see the agreement at different wind speeds. For the power curve simulations case 4 in the simulation matrix in Table 3.6 is considered. The simulated wind speeds were ranging from 5 to 10 m/s in steps of 1 m/s. Thus, the wind regime below stall for this specific turbine was simulated. The simulations are executed at steady wind conditions as such a total of 6 simulations have been conducted for both the OpenFOAM and FAST simulations. Simulations with fastFoam were not included due to the larger effort in the setup, but the results can be expected to be close to the OpenFOAM simulations as seen in the previous sections.

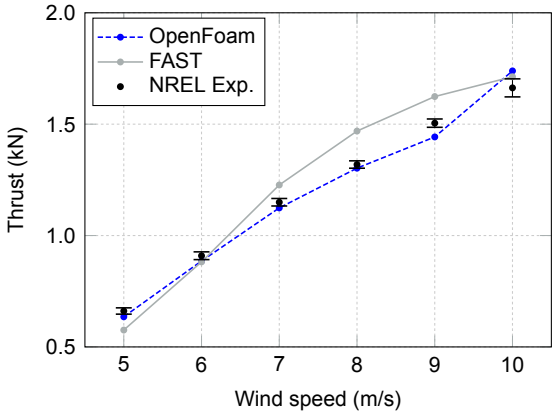
The mean values for the low-speed shaft power, thrust and torque at the different wind speeds are shown in Figure 6.24. For the experiment, error bars are included now corresponding to the standard deviation from the mean value, which is shown by the data point. For the power it can be seen that the CFD method OpenFOAM shows better agreement with the experiment at 8 and 10 m/s wind speed. At 9 m/s the mean power predicted by FAST is closer to the experimental data. For the lower wind speeds the deviations are about the same. For the thrust OpenFOAM clearly shows a better agreement for nearly all wind speeds only at 10 m/s the deviations are larger compared to FAST. Finally, for the torque FAST seems to agree better at 5 and 6 m/s, whereas OpenFOAM results are closer to the experimental results at 7, 8 and 10 m/s.

From these simulations it can be stated regarding the selected wind speeds of 6 and 7 m/s for case 1 to 3, that neither is biased towards clear better agreements of one simulation method. However, one could observe that OpenFOAM simulations show slightly better agreements compared to FAST at

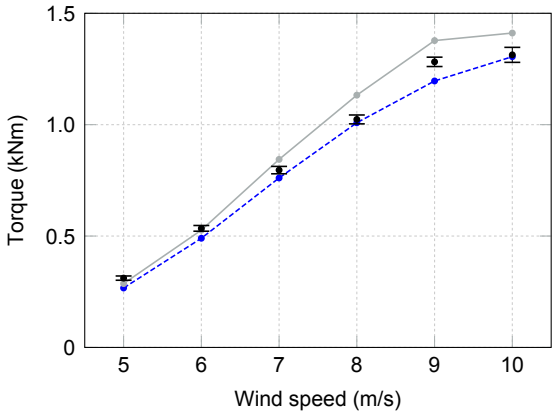
higher wind speeds ranging from 8 to 10 m/s.



(a) Low-speed shaft power.



(b) Thrust.



(c) Low-speed shaft torque.

Figure 6.24: General turbine parameters for different wind speeds obtained through converged simulations at steady wind conditions (case 4).

6.2. NREL 5MW

In addition to the phase VI turbine, a more modern turbine with more flexible blades is included with the NREL 5MW turbine. Results for the NREL 5MW turbine in normal operating conditions are investigated in Section 6.2.1. The effect of a state-of-the-art elastic blade is addressed. Due to the larger blade deflections these cases result in an increased difficulty for the mesh deformation procedure to be demonstrated. Moreover, the NREL 5MW turbine was simulated in yawed conditions at a fixed yaw angle of 30 deg, see Section 6.2.2. The FAST and fastFoam simulations were carried out as fully aero-servo-elastic simulations. Thus the effect of the controller is included for these operational conditions.

6.2.1. Normal Operating Conditions

The first case simulated addresses the normal operating conditions at the rated wind speed of 11.4 m/s. At first both the fastFoam and FAST simulations were carried out with the activated controller in FAST. Due to reasons explained in Section 6.2.1.1, later the yaw, torque and pitch control and thus the entire controller was deactivated. The results for the deactivated controller where shown with more details in Section 6.2.1.2.

6.2.1.1. Activated Controller

First of all, results for the low-speed shaft power, thrust and torque are shown in Figure 6.25. It could be observed that the low-speed shaft power for the OpenFOAM CFD method is quite large with a converged value of about 5.65 MW. Compared to this FAST converges to a much lower value of about

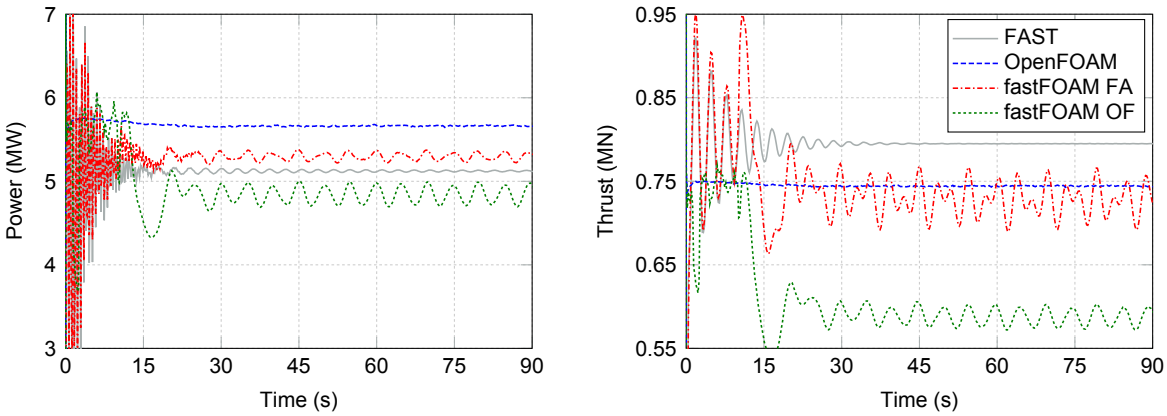
5.1 MW. The fastFoam FAST output also shows an increased value of about 5.3 MW, where again fluctuations of the power are present. The fastFoam OpenFOAM output shows a converged value of only 4.8 MW with strong fluctuations.

Notice that the values shown for FAST include the drivetrain modelling within the ElastoDyn model in FAST, whereas OpenFOAM outputs show the pure aerodynamic torque.

For the thrust one can observe that FAST now predicts the largest value with about 0.8 MN. The CFD methods converge towards a lower value of around 0.75 for OpenFOAM and 0.73 for fastFoam FAST output. The fastFoam OpenFOAM output shows a strong reduction compared to the other methods similar to the power. For fastFoam there are clear fluctuations visible compared to the other methods. The torque shows a similar picture such as for the power. OpenFOAM shows an increased value and fastFoam, especially the OpenFOAM output, as well as FAST show a strong reduction.

In all plots it can be observed that the fastFoam solution first follows the FAST solution until about 10 seconds corresponding to two rotor rotations. At this time the CFD is said to be reasonably converged and the loads from CFD are applied within FAST instead of BEM. From this point in time the values for fastFoam FAST output get closer to the standalone OpenFOAM simulation values for the power, thrust and torque.

From Figure 6.25d it can be observed that the torque for fastFoam varies with the 1P frequency, with the largest values at azimuths of 180 deg, where in principle the tower passage occurs. However, the tower was not included in the fastFoam simulations. Thus, the reason for these specific fluctuations is a bit questionable.



(a) Low-speed shaft power.

(b) Thrust.

(c) Low-speed shaft torque.

(d) Detailed torque variations at higher simulation times.

Figure 6.25: General turbine parameters according to simulations for case 5 with activated controller.

Next, the rotational speed and the pitch angle are shown in Figure 6.26. It can be seen that the

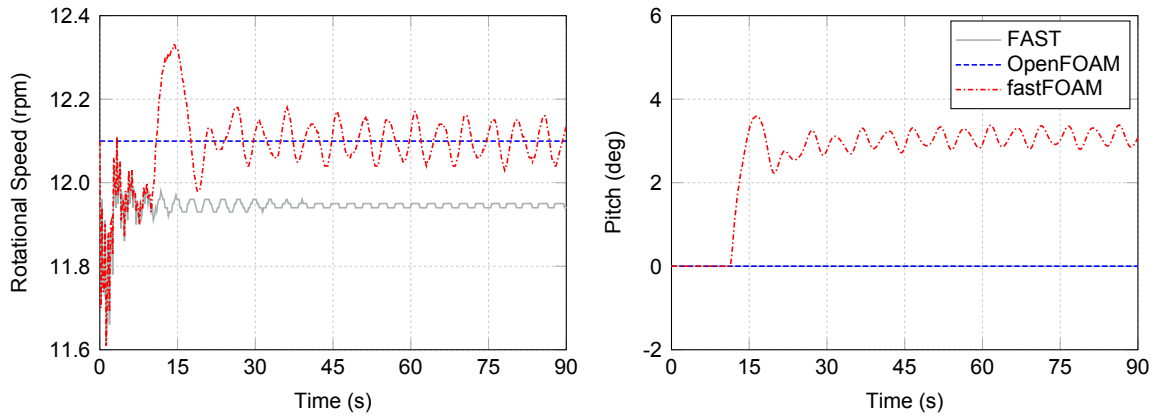
torque controller increases the rotational speed for the fastFoam simulations towards the real rated wind speed of 12.1 rpm. However, the speed shows clear fluctuations of about 1.5 rpm in amplitude. For FAST the controller sets the rotational speed lower to about 11.95 rpm.

Observing the pitch angle, it can be seen that when the loads from CFD are applied at about 10 seconds the pitch is increased for fastFoam by the pitch control in FAST. The pitch is converging but with fluctuations to about 3.5 deg. This can be explained due to the reason that the resulting aerodynamic power from CFD, see the OpenFOAM results in Figure 6.25a, would be too high and thus the pitch controller enables pitching of the blades.

The rated mechanical power of the turbine equals about 5.297 MW, see [22]. With the enabled pitch control this is approximately achieved by the fastFoam simulations. FAST actually underestimates this value slightly and thus may not be able to achieve the 5 MW of rated generator power taking into account the 94.4 percent efficiency according to [22].

That the NREL 5MW turbine results in an increased value for the aerodynamic power, respectively the related torque, is a known issue for some CFD simulations if compared to BEM, see for instance the results of [27]. The results show that the controller can handle this by the activated pitching. However, the resultant fluctuations of the pitch and also the rotational speed show that there could be improvements in terms of the control actions. Finally, it can be statet that the controller is more tuned for FAST and BEM (AeroDyn), see [22]. By retuning of the controller for the CFD for instance by increasing of the required aerodynamic power better results may be achievable. However, such a retuning would require expertise in control theory and of course time. Thus, at the current state of the project the current control behaviour is accepted, as it is still showing that the controller responds to the difference in loads for CFD compared to BEM.

Finally, the resulting 1P fluctuations for fastFoam in the power, thrust and torque could be related to similar fluctuations in the rotational speed and pitch. If the peaks and valleys are counted for a certain time period, for instance from 75 to 90 s, it can be seen that there number is equivalent for the different outputs. However, the exact cause why this occurs has not been found yet and would probably require more simulations with different settings, which due to time limitations cannot be achieved.



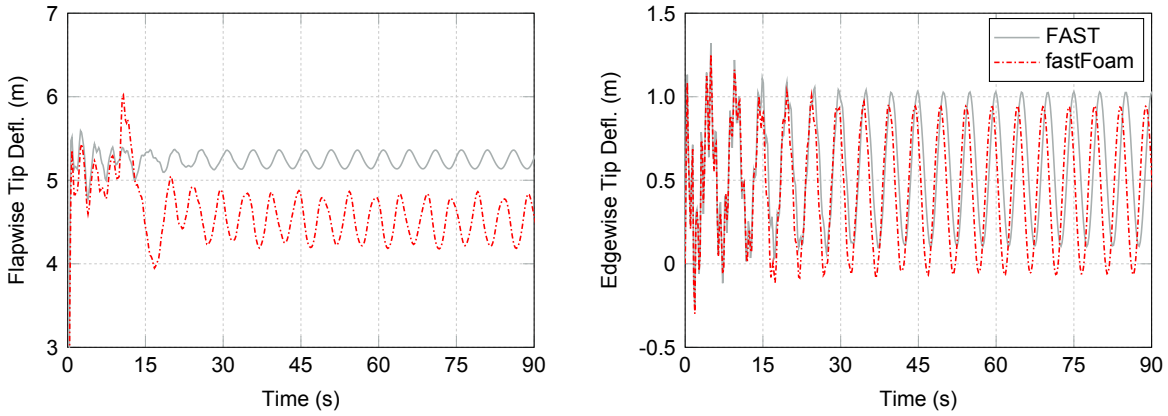
(a) Rotational speed of the rotor.

(b) Blade 2 pitch angle.

Figure 6.26: Controller parameters according to simulations for case 5 with activated controller.

Moreover, also the the displacements at the tip are compared both in edgewise and flapwise directions for the methods, which include a structural solver, see Figure 6.27. For OpenFOAM the rigid blade assumption is used and thus the displacement is zero. It can be seen that due to the pitching the flapwise tip deflection is lower for fastFoam compared to FAST . However, due to the fluctuations the deflections show also variations with larger amplitudes compared to FAST. The mean value is about 5.2 m for FAST while only 4.5 m of deflection for fastFoam are obtained.

For the edgewise displacements the effect of the pitching is reduced and thus the tip deflections are about the same with slightly decreased values for fastFoam. This can be explained as the edgewise deflections are mainly driven by the gravitational loads which, remain unchanged by the pitching motion.

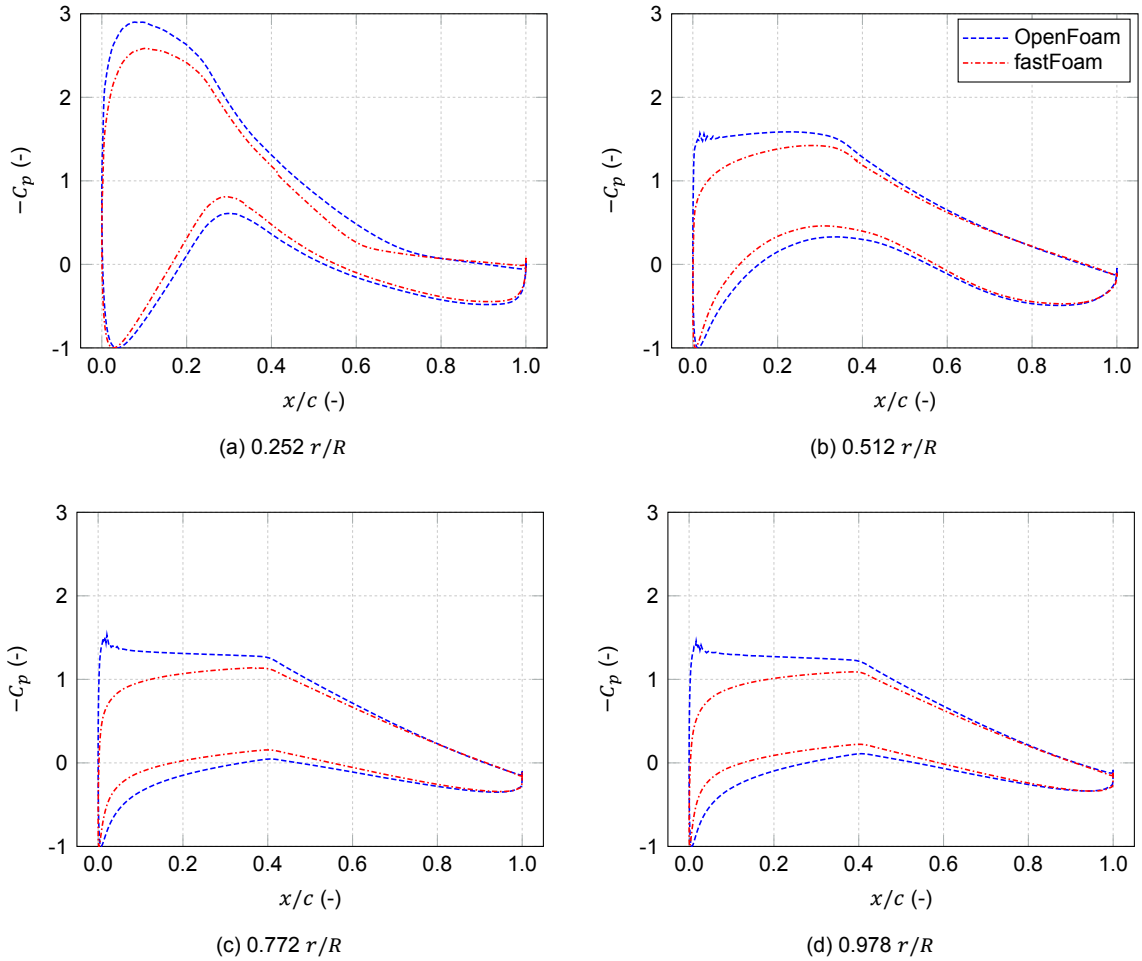


(a) Flapwise direction.

(b) Edgewise direction.

Figure 6.27: Blade 2 tip displacements according to simulations for case 5 with activated controller.

The resulting pressure distributions are shown in Figure 6.28 for both the rigid and elastic blade CFD methods. As can be seen the pressure reduces in magnitude for fastFoam, which can be attributed to the pitching of the blades thereby directly decreasing the angle of attack.



(a) 0.252 r/R

(b) 0.512 r/R

(c) 0.772 r/R

(d) 0.978 r/R

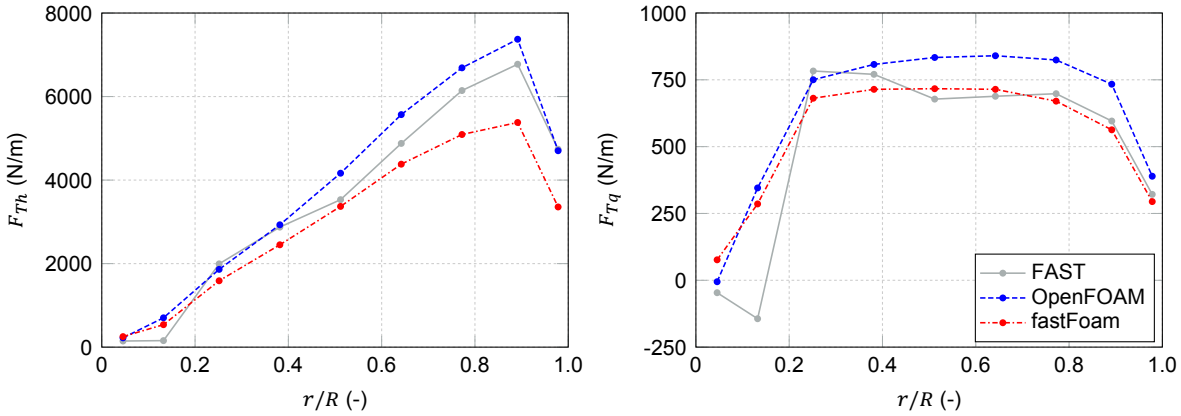
Figure 6.28: Pressure distributions at different blade sections for case 5 with activated controller (0 deg azimuth).

The effect of the pitching according to the controller can be investigated from the loads such as

shown in Figure 6.29. As can be seen, the thrust and torque contributing forces reduce drastically for fastFoam compared to OpenFOAM. The forces related to the torque are kept approximately equivalent to the forces from FAST in magnitude at the outer part of the blade. OpenFOAM results clearly show a strong increase in loads compared to FAST, which may further explain the reason of the pitching motion applied.

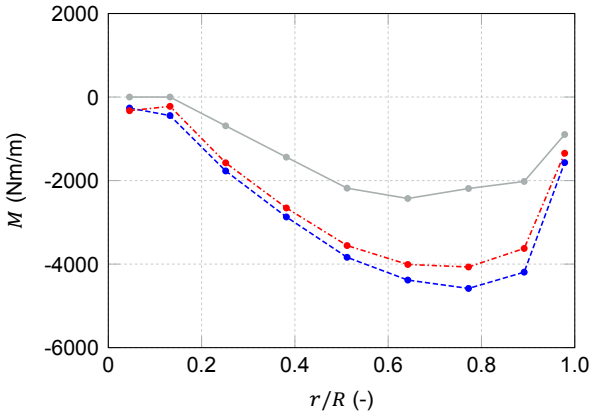
For the pitching moment the CFD results have a similar order of magnitude, whereas FAST estimates the magnitude only about half as large. This is contradicting to the NREL phase VI results where the magnitude was approximately similar. The reason for this is currently unknown and would require more investigations. One explanation, which would be need to be proven, could be due the reason that FAST only considers the lift and drag forces acting at one point in one section with one moment arm. In contrast to this for CFD, the moment is composed of the forces from every cell and their related moment arms assigned to the section. Thus the difference could be resultant from the mesh matching of a three dimensional surface mesh to one beam node.

This is a point of attention as the deviation compared to other comparisons between FAST and the CFD methods is quite large and thus should be investigated further. However, due to lack of time and possibly required simulations this difference is only noted for now. One additional simulation was run were only the forces were communicated and the moments were set to zero in FAST. This was done to investigate the influence of the aerodynamic moments compared to the forces. It was found that the influence is very small and nearly negligible. This should be especially true if no torsion is applied such as for ElastoDyn.



(a) Thrust.

(b) Torque contribution.



(c) Pitching moment.

Figure 6.29: Force and moment distribution at different blade sections for case 5 with activated controller (0 deg azimuth).

To conclude it can be stated that the pitching of the blades in fastFoam reduces the overall loads

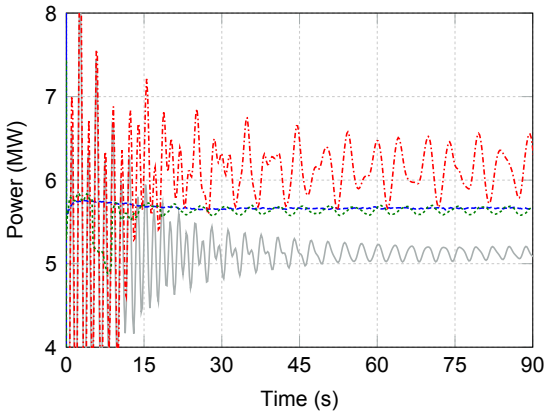
and thus deflections. Thereby, the comparison between FAST and fastFoam as well as OpenFOAM is influenced. Therefore, it was decided to run another simulation where the pitch and also torque controller is deactivated to allow for a better comparison at similar pitch angles and rotational speeds.

6.2.1.2. Deactivated Controller

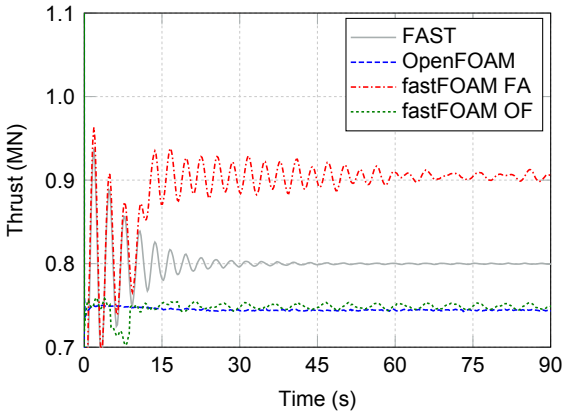
As a next step, the entire controller describing the torque, pitch and yaw control was disabled within FAST and thus also for fastFoam. In addition, the generator degree of freedom was disabled. The result for the low-speed shaft power, thrust and torque is shown in Figure 6.30. It can be observed that due to the lack of the controller clearly more initial fluctuations are present and both FAST and fastFoam take much longer time to converge. Whereas FAST clearly converges towards a constant solution for the power, thrust and torque, fastFoam only seems to converge to a relatively constant value for the thrust considering the FAST output. For the power and torque the fastFoam FAST output shows more pronounced fluctuations, compared to the case where the controller was disabled, see Figure 6.25 for comparison. This could be a hint that the drivetrain modelling in FAST will amplify the fluctuations, especially if no controller is present to damp these.

Considering the fastFoam OpenFOAM output, one can see good agreement with OpenFOAM as now the rotational speed is equal due to the deactivated control. However, at the start of the fastFoam simulation a sudden valley in power, torque and thrust can be seen for the fastFoam OpenFOAM outputs. These can be attributed to the phase where the blades in the CFD mesh are smoothed from rigid to elastic state, which exactly takes place in this period. Thus an effect of blade deflections on power, thrust and torque is proven. Such an effect may also explain why these parameters also show periodic fluctuations for the fastFoam OpenFOAM output in converged state, where also periodic fluctuations of deflections occur which will be investigated later.

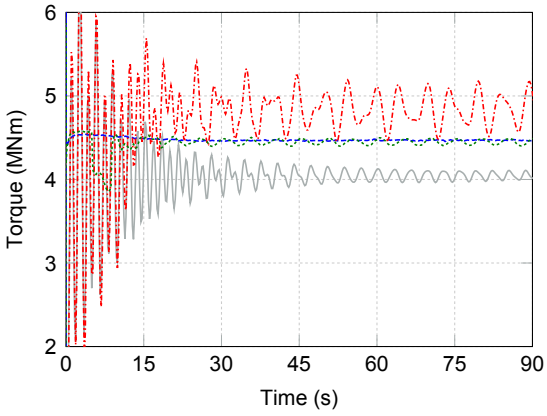
For the converged torque state, such as shown in Figure 6.32 it can be seen that only the methods which include the blade deflections show fluctuations. Whereas the pure aerodynamic torque represented by the fastFoam OpenFOAM output does show an obvious sinusoidal behaviour (with a 1P frequency) a clear sinusoidal wave is not present for fastFoam FAST output. For this output there seems to be periodic behavior, but of more disturbing nature (not strictly related to any 1P, 2P or 3P frequency). An explanation, which again would require more investigations, could be that due to the drivetrain modelling combined with the periodically varying pure aerodynamic torque these more randomly driven fluctuations are obtained. Comparing this to FAST one also observes more randomly slowly decaying fluctuations which could be related to the 2P frequency.



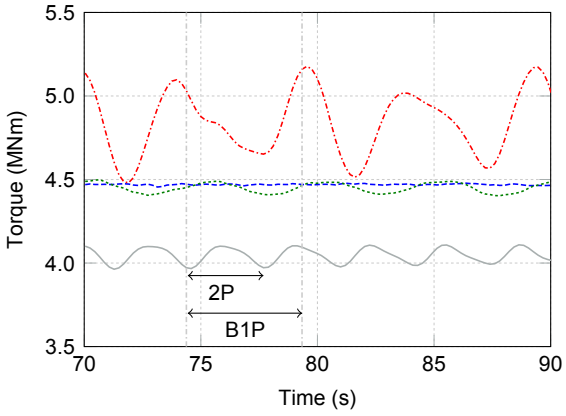
(a) Low-speed shaft power.



(b) Thrust.



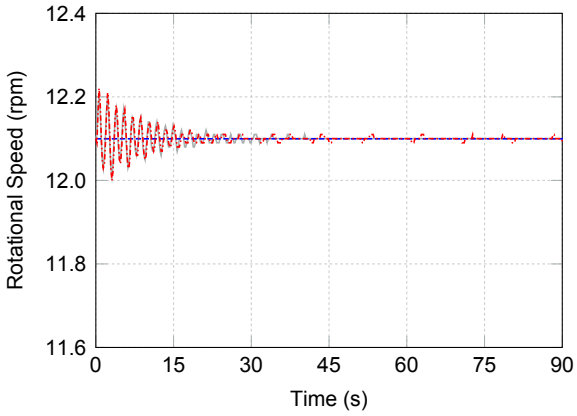
(c) Low-speed shaft torque.



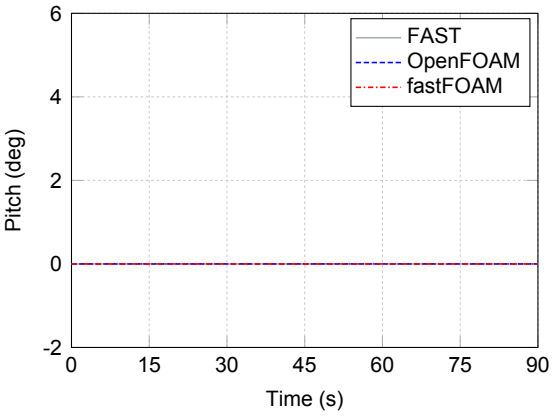
(d) Detailed torque variations at higher simulation times.

Figure 6.30: General turbine parameters according to simulations for case 5 with deactivated controller.

The control variables rotational speed and pitch angles are reported in Figure 6.31 for this simulation. As can be seen, for the FAST and fastFoam simulations some initial fluctuations are present in the rotational speed, but later it is converging to the constant value such as used in the OpenFOAM simulations. For the pitch angle now the zero pitch setting is applied throughout all simulations.



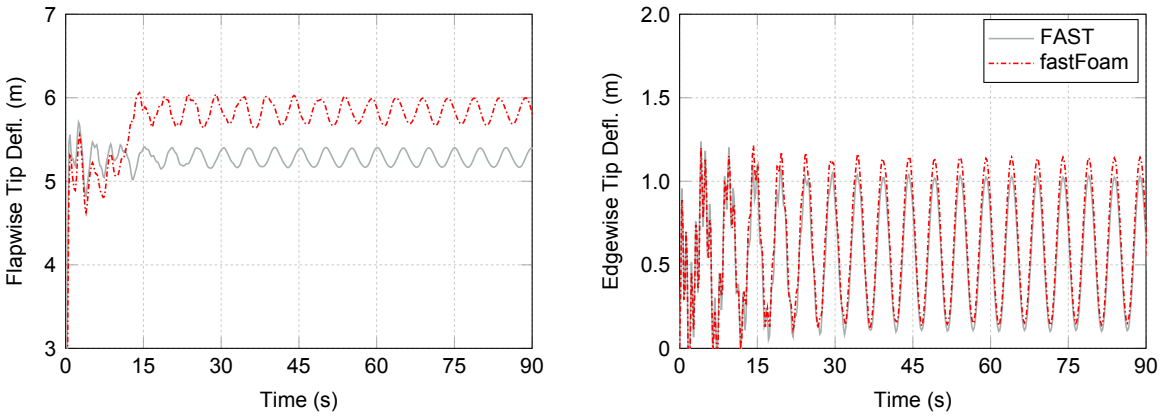
(a) Rotational speed of the rotor.



(b) Blade 2 pitch angle.

Figure 6.31: Controller parameters according to simulations for case 5 with deactivated controller.

Moreover, the flapwise and edgewise tip deflection evolutions are shown in Figure 6.32. It can be noticed, that the flapwise displacements for both fastFoam and FAST show a similar sinusoidal behavior with a 1P frequency. However, the peak values occur at slightly different times, thus an offset of the phase has occurred. The amplitude and mean of the flapwise tip deflections for fastFoam are increased compared to FAST. For the edgewise results good agreement is obtained, which can be expected as these deflections are driven by gravity.

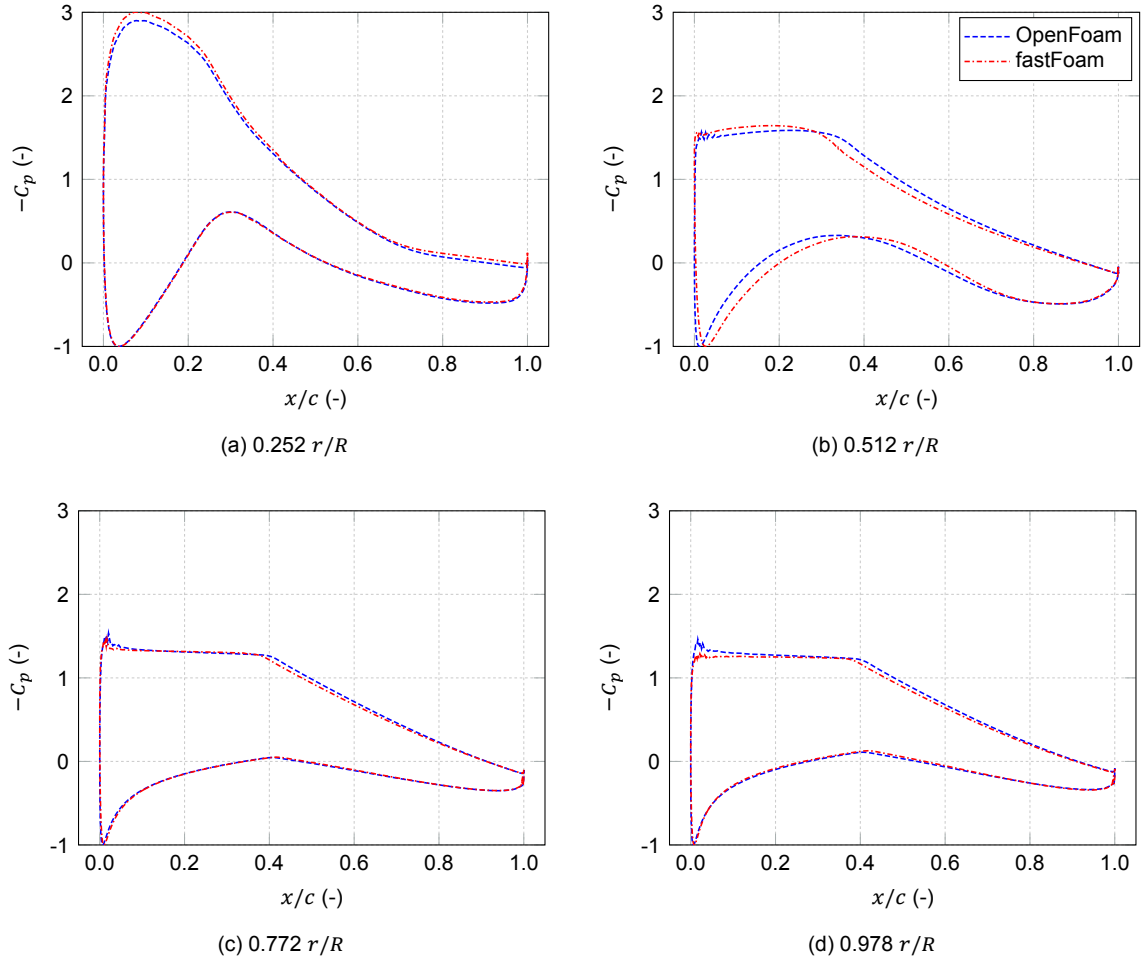


(a) Flapwise direction.

(b) Edgewise direction.

Figure 6.32: Blade 2 tip displacements according to simulations for case 5 with deactivated controller.

Next, also the effect of the elastic blades on the pressure distributions is investigated in Figure 6.33. Comparing the OpenFOAM rigid blade pressure coefficients with fastFoam it can be observed that the pressure coefficient agrees relatively well over the entire span. Only at the mid section at about half span more pronounced deviations are visible. The stagnation point shifts slightly backward to the trailing edge for this section. Whereas at the inner part the upper surface pressure coefficient is increased for the elastic blade, at the outer sections there is a slight reduction. As no torsion is included and thus no direct change of angle of attack, these differences could be attributed to the different positions of the blade if modelled elastic or rigid. In the three dimensional space this could then result for instance in a slight change in the incoming velocity seen by the section, which would influence the pressure distribution.



(a) 0.252 r/R (b) 0.512 r/R

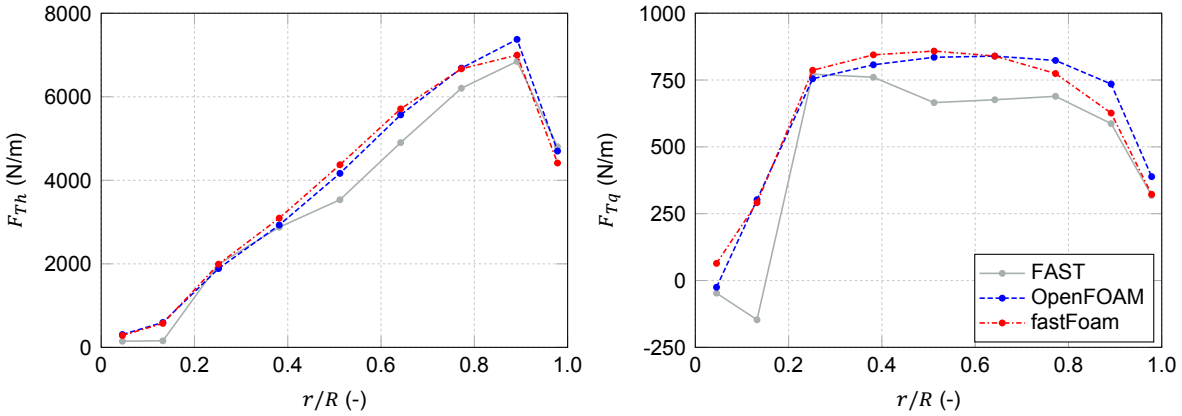
(c) 0.772 r/R (d) 0.978 r/R

Figure 6.33: Pressure distributions at different blade sections for case 5 with deactivated controller (0 deg azimuth).

Next, the aerodynamic forces have been investigated in Figure 6.34. The pitching moment was not included now as the previously investigated strong deviations between the CFD and BEM methods were existent again, disallowing a reasonable comparison, see Figure 6.29c. Therefore, further investigations would be required, which due to time reasons could not be achieved for now.

Considering the aerodynamic thrust forces, good agreement can now be observed between the CFD methods with slightly increased values for fastFoam in the middle sections. Only near the tip the deviation gets larger, where fastFoam predicts reduced loads. The CFD methods show an increase in loads of about 20 percent at the outboard sections from 50 to 90 percent span. At the tip all methods show relatively good agreement again.

For the forces which contribute to the torque, fastFoam shows a constant increase compared to OpenFOAM of approximately 10 percent from 25 to 50 r/R. At the tip fastFoam indicates a decrease in loads compared to OpenFOAM. In the central blade part FAST shows a strong reduction compared to the CFD methods. The load distribution obtained from FAST is not very smooth compared to the one obtained from CFD. For instance near the root the forces show a decrease first at about 0.15 r/R and then a sudden increase. This could again be attributed to the BEM method utilizing different airfoil data and thereby resulting in sudden jumps.



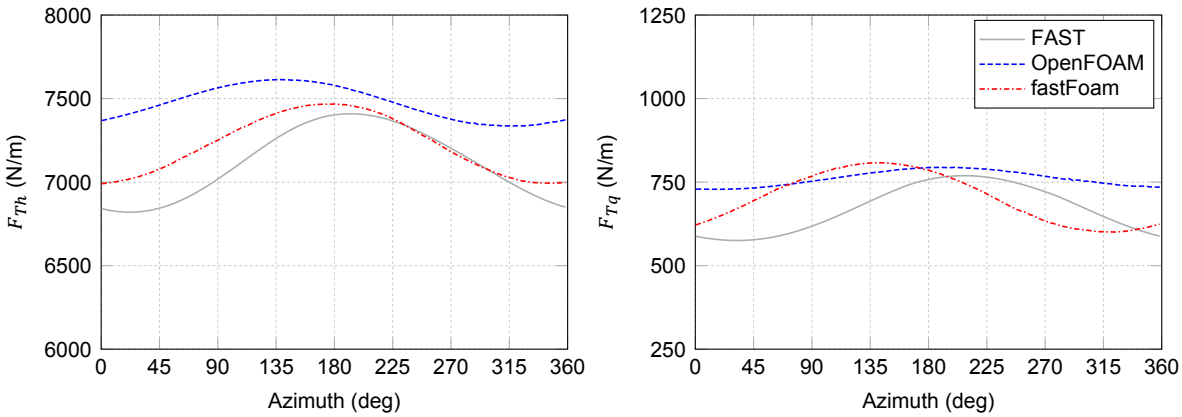
(a) Thrust.

(b) Torque contribution.

Figure 6.34: Force distribution at different blade sections for case 5 with deactivated controller (0 deg azimuth).

Next, the variations of the aerodynamic forces over one rotation are considered in Figure 6.35 at an outer blade section. For the thrust force it can be observed that both fastFoam and FAST, which include blade elasticity, show a peak at about 180 deg azimuth. For FAST this peak is slightly shifted to higher azimuths. This shift also explains the phase difference for the flapwise deflections, see Figure 6.32a. In contrast to this, OpenFOAM, predicts largest forces at azimuth angels of about 135 deg. Due to the rotor tilt of 5 deg, the development of the wake and the possible blade deformations such variations occur. The deformed blades seem to shift the occurrence of maximum thrust forces to higher azimuths shown by FAST and fastFoam.

For the forces contributing to the torque a different picture is observed. Whereas OpenFOAM shows a peak at about 180 deg, fastFoam and FAST both deviate by about 45 deg. As such FAST predicts maximum forces at 225 degree, whereas fastFoam results show a peak at 135 deg. Thus there is a significant difference between FAST and fastFoam in the prediction when the maximum torque occurs.

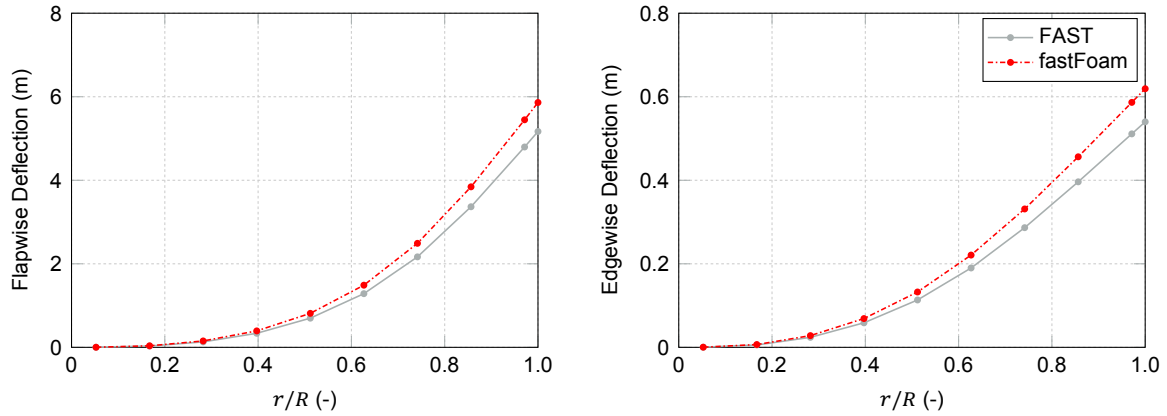


(a) Thrust variation.

(b) Torque contribution variation.

Figure 6.35: Force azimuthal variations at outer blade section (0.892 r/R) for case 5 with deactivated controller.

In addition, the deflections over the blade span are shown in Figure 6.36. It can be concluded that due to the increased loads, fastFoam also shows increased deflections in both flap- and edgewise direction compared to FAST.

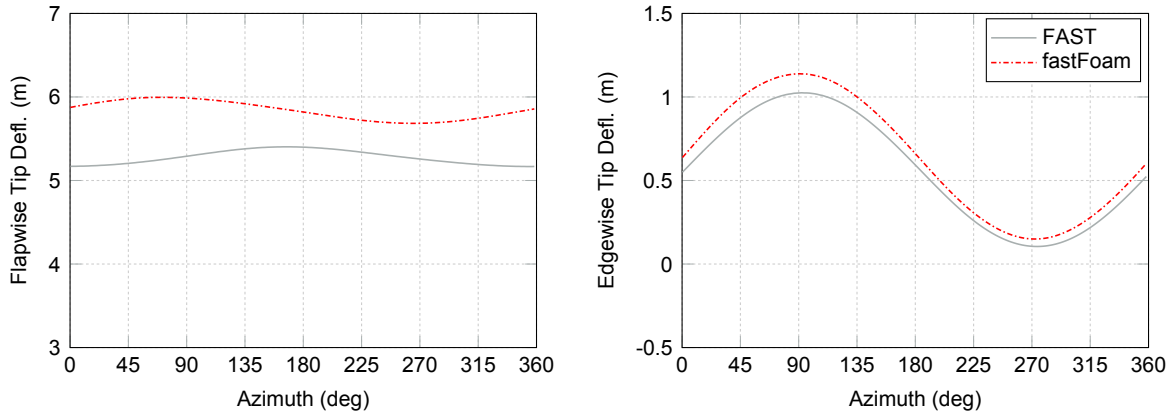


(a) Flapwise tip deflection variation.

(b) Edgewise tip deflection variation.

Figure 6.36: Displacements at different blade sections for case 5 with deactivated controller (0 deg azimuth).

Finally, the variation of the blade tip deflections is reported in Figure 6.37 in both directions. For the flapwise direction the phase shift of about 110 deg can be seen again. This is an interesting observation, but for an explanation further investigations would be required. In contrast to the flapwise tip deflections, the edgewise ones agree well in phase and magnitude with a slight increase for fastFoam compared to FAST.



(a) Flapwise tip deflection variation.

(b) Edgewise tip deflection variation.

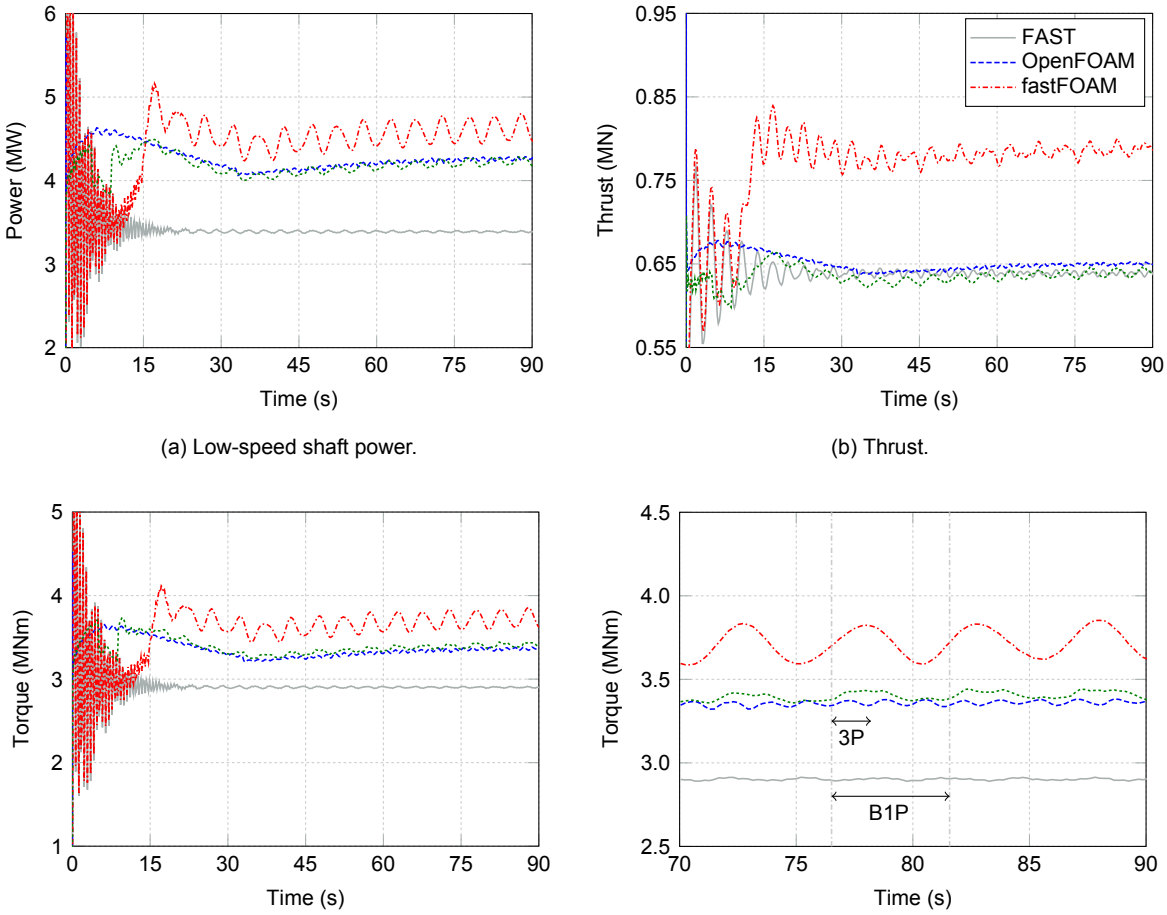
Figure 6.37: Azimuthal variation of displacements at blade tip for case 5 with deactivated controller.

6.2.2. Fixed Yaw Error with Activated Controller

For the unsteady yaw case with an activated torque and pitch controller the results for the Low-speed shaft power, thrust and torque are given in Figure 6.38 for the different simulation methods. It can be observed that the fastFoam method first follows FAST for approximately the first 10 seconds. After this time the loads from CFD are again applied instead of the BEM loads. Therefore, mainly the power and torque get closer to the OpenFOAM results. The CFD methods take about 75 seconds until they converge, whereas FAST shows much faster convergence from about 30 seconds. For fastFoam FAST output significant periodic fluctuations are present in the power and torque, whereas for the thrust more random variations can be observed. If the OpenFOAM processing is run on the fastFoam simulations the results agree well with OpenFOAM with rigid blades, only the fluctuations increase. Both CFD methods show an increase in power and torque, while only for fastFoam FAST output a significant increase in thrust can be observed compared to FAST and OpenFOAM.

Whereas FAST shows a relatively converged solution towards one value for the torque, the CFD methods show periodic fluctuations, see Figure 6.38d. It is interesting to observe that the fastFoam

simulation describes a periodic behaviour with a 1P frequency (about 5 s period), while the OpenFOAM simulations result in a 3P periodic variation ($1.\overline{66}$ s period), driven by the blade passing frequency (3P). This reason for this different behaviour needs to be investigated further to be fully understood.



(a) Low-speed shaft power.

(b) Thrust.

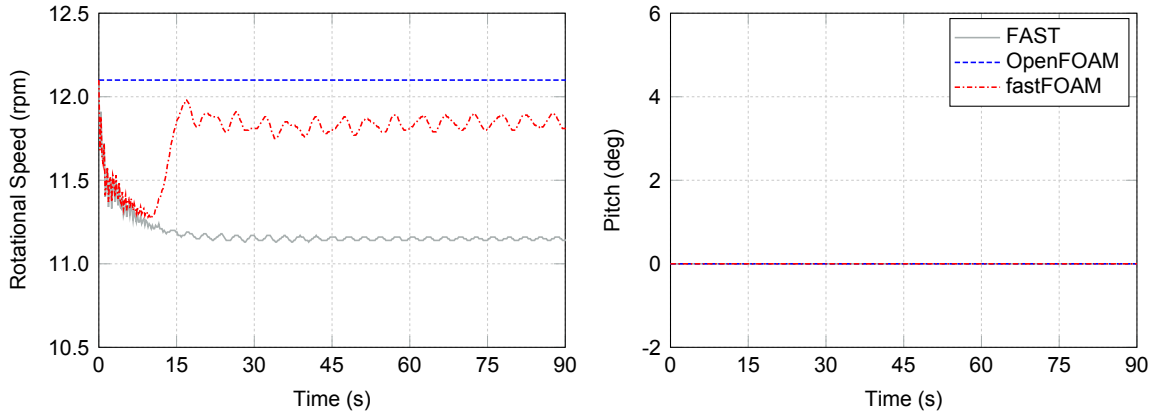
(c) Low-speed shaft torque.

(d) Detailed torque variations at higher simulation times.

Figure 6.38: General turbine parameters according to simulations for case 6.

To see how the torque and pitch control perform, the rotational speed and the pitch angle are given in Figure 6.39. It can be seen that for the FAST simulations the torque controller immediately reduces the rotational speed of the rotor. For the fastFoam simulations up to 10 seconds the controller follows the FAST simulation, but when the CFD loads are applied it increases the rotational speed closer to the rated rotational speed of 12.1 RPM. However, the torque controller fails to settle a constant speed. It converges to 1P periodic fluctations of the rotational speed. These periodic fluctations may be the driver of the power, thrust and torque periodic fluctations as seen in Figure 6.38.

The reason why the controller acts like this would require further investigations and probably simulations. However, a main reason could be again that the torque controller is tuned for FAST and thus BEM and a retuning could most likely reduce these fluctutations. In contrast to the torque controller the pitch control is passive and does not initiate any pitching motion.



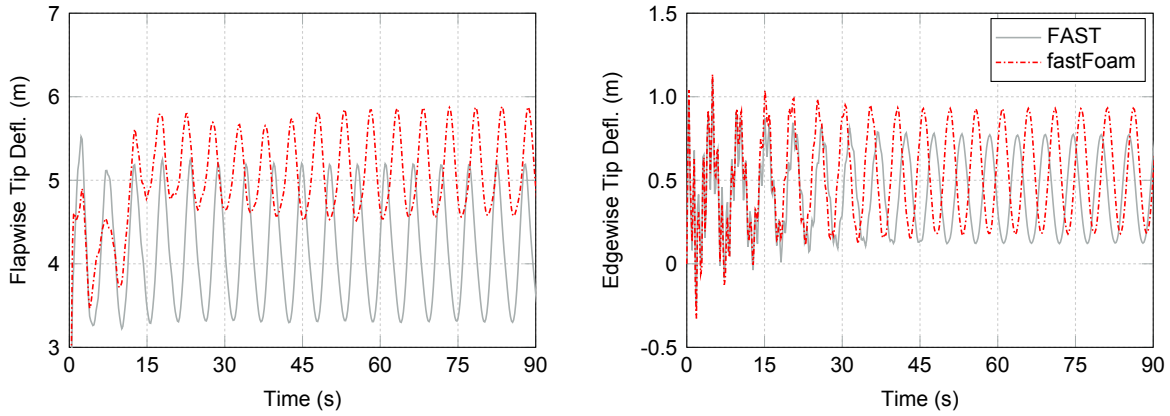
(a) Rotational speed of the rotor.

(b) Blade 2 pitch angle.

Figure 6.39: Controller parameters according to simulations for case 6.

Moreover, also the flapwise and edgewise displacements at the tip are reported in Figure 6.40. For OpenFOAM no value is shown as it is zero again, due to the assumption of rigid blades. It can be seen that fastFoam predicts an increase in the mean value of the flapwise tip deflection. However, the amplitude of the flapwise variations ins greatly reduced. The periodic fluctations are not in phase, which can be explained due to the different rotational speeds such as previously reported in Figure 6.39.

For the edewise tip deflections one can observe that now fastFoam shows a larger amplitude and mean value compared to the FAST simulations. However, the diference is much smaller compared to the flapwise tip deflections.



(a) Flapwise direction.

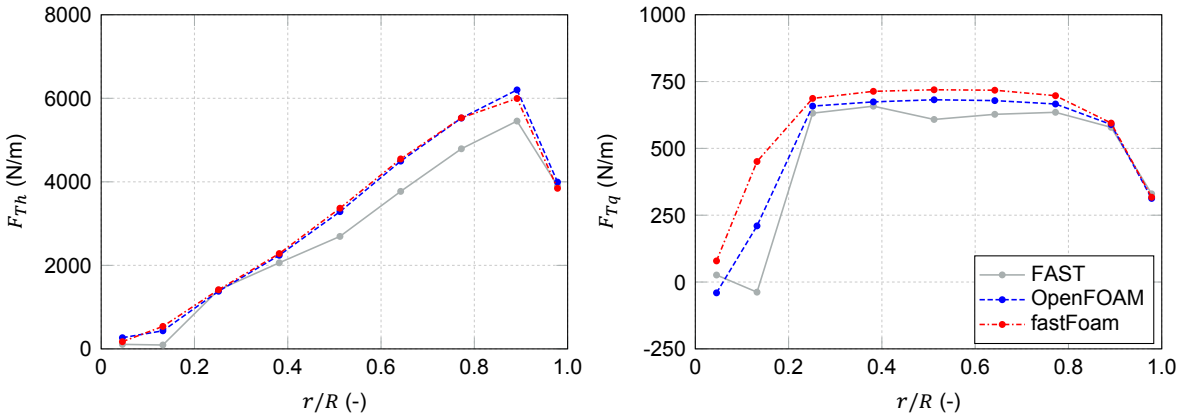
(b) Edgewise direction.

Figure 6.40: Blade 2 tip displacements according to simulations for case 6.

In addition, the load distributions at several sections are shown in Figure 6.41. It can be seen that the thrust forces for both CFD methods agree well and show a smooth increase towards the outer blade sections. In contrast, FAST shows a not so smooth behaviour with a sudden increase near the 0.3 r/R section. One reason for this could be due to the sudden jumps in airfoil polar data used within the BEM method. Near the outer blade regions the CFD methods show an increase of about 15 percent in forces compared to FAST. Near the tip at 0.98 r/R all three methods show a good agreement at an azimuth angle of 0 deg.

For the forces contributing to the torque a similar picture can be observed. The FAST results seem not as smooth and show a decrease towards the middle blade section. At the root a sharp decrease occurs again near the 0.15 r/R section, which may be due to the sudden jump from circular to airfoil

shaped sections. Different to the thrust, the fastFoam results show a significant increase also compared to the OpenFOAM results.

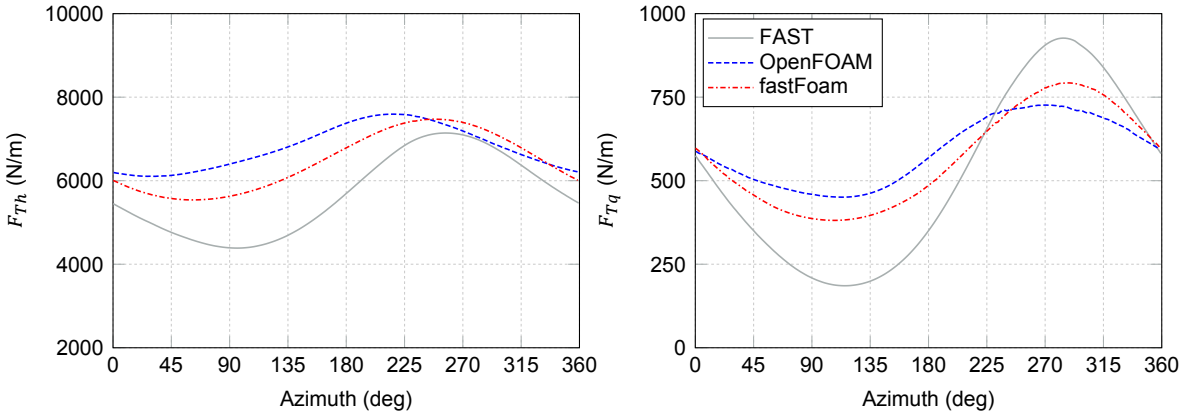


(a) Thrust. (b) Torque contribution.

Figure 6.41: Force distribution at different blade sections for case 6 (0 deg azimuth).

Finally, for the loads also the azimuthal variations are given in Figure 6.42. It can be seen that at the outer blade section a relatively good agreement is obtained for the forces. For the thrust the peak shifts to higher azimuths (about 270 deg) for the FAST and fastFoam method compared to the OpenFOAM approach. This might be due the modelling of elastic blades compare to the rigid blade approach. Whereas OpenFOAM and fastFoam agree well in the mean values and the amplitude, FAST shows a reduction in mean values but an increased amplitude of the fluctuations. Such an increase in amplitude was already present for the higher yaw angles for case 2 of the NREL phase VI results, see Figure 6.14. It can be attributed to the deficiency of the skewed-wake correction model in FAST and generally in BEM.

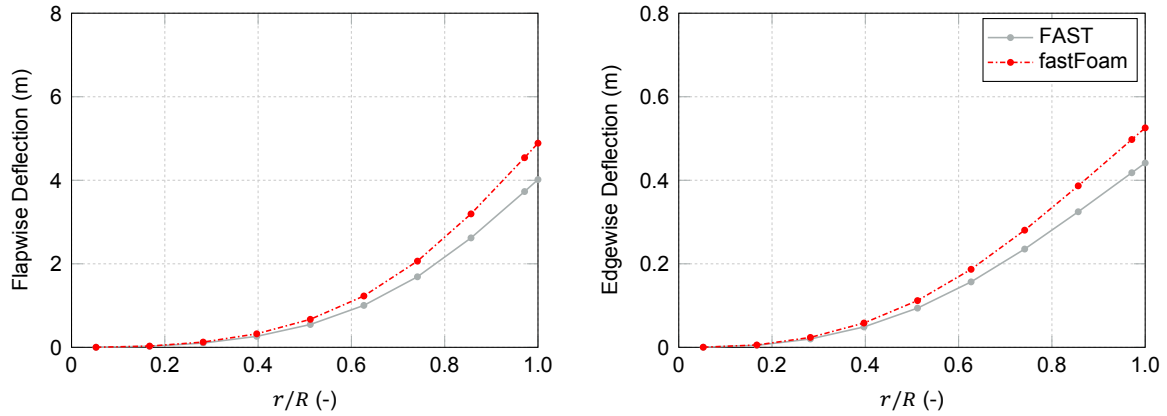
The same effect is observable in the torque contributing forces. It can be seen that the amplitude greatly reduces for FAST compared to the CFD methods. However, for fastFoam also an increase is observed but with much lower magnitude.



(a) Thrust variation. (b) Torque contribution variation.

Figure 6.42: Force azimuthal variations at outer blade section (0.89 r/R) for case 6.

Additionally, the distribution of displacements over the span is given in Figure 6.43. It can be seen, that both the flapwise and edgewise blade deflections increase if the loads from CFD are applied. This is reasonable as the previous load comparison showed an increase in the loads such as reported in Figure 6.41.

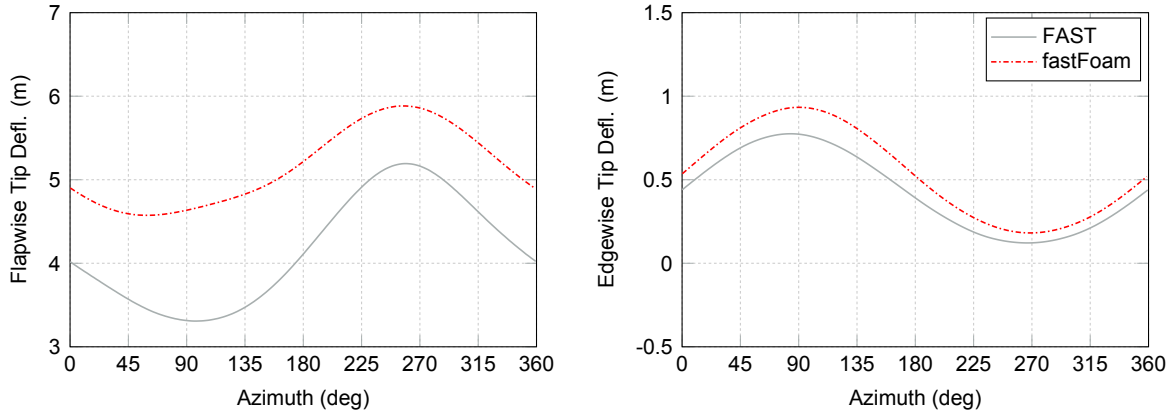


(a) Flapwise tip deflection variation.

(b) Edgewise tip deflection variation.

Figure 6.43: Displacements at different blade sections for case 6.

Similarly to the load azimuthal fluctions also the fluctions for the flapwise and edgewise tip deflections are given, see Figure 6.44. It can be stated that the flapwise deflections follow the azimuthal variation of the thrust forces such as shown in Figure 6.42a. In contrast, the edgewise deflections are not mainly related to the torque contributing force variations, see Figure 6.42b. This is due to the reason that for the edgewise deflections the main contributor is the gravitational force. Noting that a positive edgewise deflection is towards the leading edge, the largest occurs for an azimuth of about 90 degree for a clockwise rotating rotor.



(a) Flapwise tip deflection variation.

(b) Edgewise tip deflection variation.

Figure 6.44: Azimuthal variation of displacements at blade tip for case 6.

6.3. Computation Time

Finally, a comparison of the computation time is given in Table 6.2 for the different methods. It can be seen that a FAST BEM-based simulation requires a neglectable amount of computation time compared to the CFD methods. This is a result of the simplified engineering approach of BEM versus the highly iteratively CFD methods, where the Navier-Stokes equations are solved on a large computational mesh. This is the biggest disadvantage of the CFD, which limits it despite its good accuracy to only specific tasks such as detailed design simulations for instance.

For the computations the cluster Eddy of the University of Oldenburg was mainly used, see [24]. Before it was available the cluster Carl, here referred as old cluster, was utilized. Eddy is composed of 241 Intel Xeon CPU E5-2650 with 2.2 GHz and 24 cores each. With the CFD methods using about 144 to 384 cores only about two to three rotor revolutions could be achieved per day.

Next, comparing the CFD methods OpenFOAM and fastFoam it can be seen that the OpenFOAM method is generally faster. This is mainly due to the reason that whereas the OpenFOAM simulations only require rigid mesh movements and the PIMPLE loop in `pimpleDyMFoam`, fastFoam uses in addition the elastic mesh movement, the load calculation at the beam nodes as well as the coupling itself. Overall, the computation time is thereby significantly increased.

Table 6.2: Computation time of the cases simulated by different methods.

Case	Iterations	FAST	OpenFOAM	fastFoam
Case 1	17,248	1 min	17.99 d (168 cores, old cluster)	9.67 d (144 cores)
Case 2	25,866	1 min	14.74 d (288 cores, old cluster)	11.83 d (144 cores)
Case 3	25,822	1 min	13.21 d (144 cores)	15.75 d (144 cores)
Case 5	13,068	1 min	9.48 d (360 cores)	14.53 d (360 cores, control) 14.31 d (384 cores, no control)
Case 6	13,068	1 min	9.55 d (360 cores)	12.81 d (192 cores)

To have a better understanding of what is causing this huge computational effort, a breakdown for one time step is given in Table 6.3 for both NREL phase VI and 5MW cases. As shown previously in Table 4.2 the number of cells for the 5MW turbine is about three times as large compared to the phase VI, resulting in clear differences for the computational time. It can be seen that the PIMPLE loop clearly dominates for the phase VI simulation in case 3 with about 40 s of time spent. However, also the rigid mesh motion is computationally expensive due to the AMI interpolation, which is required at the AMI patches used for the pitch and azimuth motions. The share of the elasticity related motion via the beams is nearly neglectable with about 1 s. The coupling itself has the same order of magnitude in computation time compared with the AMI interpolation. However, this also includes other processes such as the runtime processing, which may also take its share. For a better overview of how long only the coupling takes this should be distinguished in an updated version.

The computation time for the OpenFOAM computations is only composed of the rigid mesh motion as well as the main contributor the PIMPLE loop. For the phase VI then an increase of about 15 percent is obtained for fastFoam compared to OpenFOAM, due to the additional components such as the coupling and the mesh motions accounting for the elastic blades. Conducting Table 6.2 an overall increase of about 20 percent can be seen for case 3 for the fastFoam simulations compared to OpenFOAM. The difference can be attributed to the increased convergence difficulty in the beginning of the simulations as a result of the unsteady rigid motions and the blade deformations used in fastFoam.

Considering the NREL 5MW simulations it was first decided for case 5 to increase the number of cores significantly, due to the increase by a factor of three in the number of cells compared to the phase VI. However, it can be seen that the AMI interpolation for the rigid mesh motion and also the coupling process do not scale very well with an enormous amount of cores. This may be due to the reason that the communication time between the processors gets really large, which should be avoided. However, the computational time for the PIMPLE loop can then be lowered drastically.

Therefore, for the last executed simulation of case 6 the number of cores was lowered again. Then, the required time for the AMI interpolation and the coupling is reduced, but the time share for the PIMPLE loop is significantly increased (51 s). The overall time of one time step for case 5 and case 6 with 360 vs 192 cores for the converged state is quite large for fastFoam with both 83 s. An increase of 32 percent compared to OpenFOAM seems huge. It could be that there is an optimal amount of cores in between the used 192 and 360, which reduces the computational time and results in an increase more similar in magnitude such as for the phase VI. It would thus be advisable for further simulations to test for instance 288 cores to see if the computation time of one time step reduces.

Considering again Table 6.2 it can be seen that although the time share for case 5 and case 6 is equal in converged state, the overall time is nearly 2 days lower for the same number of iterations in case 6. This could indicate that in the non-converged state in the beginning of the simulation, the higher number of cores seem to have a negative effect due to larger communication times between processors.

Table 6.3: Breakdown of computational time for one iteration for the CFD methods at converged state.

Parameter	Case 3	Case 5	Case 6
Rigid mesh motion via AMI	5 s	25 s	14 s
Elasticity related mesh motion	1 s	1 s	1 s
Coupling process and others	6 s	27 s	17 s
PIMPLE loop	40 s	30 s	51 s
Total time of one iteration for fastFoam	52 s	83 s	83 s
Total time of one iteration for OpenFOAM	45 s	63 s	63 s
Increase fastFoam versus OpenFOAM	15 %	32 %	32 %

Also it needs to be noted that there are certain disturbances of random nature, which may lead to a slowdown of the simulations such as an increased RAM usage on the cluster for instance. The Eddy cluster uses computing nodes both with 64 and 128 GB of RAM, these were selected depending on availability, see [24]. Therefore, for some simulations slower nodes were used whereas other simulations were performed using the fastest computing nodes.

7

Conclusions and Recommendations

In this Chapter conclusions and recommendations are given based on the previously obtained results in view of the research objective. Therefore, first a detailed overview of the conclusions is given followed by the recommendations which the project resulted in.

7.1. Conclusions

Finally, key conclusions from the implemented method and the calculated results are obtained and related to the research questions to be answered.

First of all, it can be stated that the project objective is achieved by implementing the fastFoam solver in OpenFOAM, which is coupled to FAST via the MpCCI coupling environment. The newly obtained fastFoam method allows for aero-servo-elastic wind turbine simulations using FAST, but now fully based on CFD. The selected coupling follows a loose coupling procedure, where the data between OpenFOAM and FAST is only exchanged once per time step. The exchanged data are the blade positions and orientations together with the global angles (yaw, azimuth and pitch) from FAST as well as the loads from CFD. The coupling is intended to be minimally intrusive, thus no radical changes were done in the FAST source code. In addition, for OpenFOAM the existing `pimpleDyMFoam` solver was mainly extended by mesh movements based on beams (similar to FAST) and load mappings from blade surface to beam nodes, see Chapter 5.

Simulations based on the three different methods FAST, OpenFOAM and fastFoam were executed for two turbines the NREL phase VI and 5MW in order to validate the developed method and answer the main research questions.

NREL phase VI simulations:

- From the NREL phase VI results in normal operating conditions, see Section 6.1.2, it can be concluded that all simulation methods show relatively good agreement. For such a rotor, which is equipped with very stiff blades, no significant differences between the developed aero-servo-elastic fastFoam method and the OpenFOAM CFD methods are observable due to the low magnitude of blade deflections. However, more pronounced differences are shown between the BEM-based FAST method and the CFD methods due to the different aerodynamic modelling approach.

- In extreme operational conditions such as large yaw angles of 20 to 30 degree or increased pitch angles, see Sections 6.1.3 and 6.1.4, it is shown that the BEM method results in large deviations from the experimental data, whereas the CFD methods still agree relatively well. The reason for this can be found in the engineering approach within BEM, where yaw models and unsteady aerodynamic models are implemented. For the yawing it was clearly observable in Figure 6.14 that the implemented Pitt/Peters skewed-wake correction model does not predict the load variations due to the unsteady inflow as accurately as the CFD. In addition, the unsteady aerodynamics modelling fails to resolve the aerodynamic loads precisely if the angle of attack is reduced due to pitching, see Figure 6.20. The disagreement for the BEM method with the

experiment was found to be especially existent near the tip, where large loads act and most of the energy is generated.

- The comparison with the experimental data from the phase VI experiment leads to the conclusion that the implemented fastFoam method has been validated. However, one point of attention was found to be fluctuations in the power, torque and thrust of the FAST output for the fastFoam simulations, which cannot be explained from theory and thus could be the resultant of an introduced error within FAST only present for the coupled approach.

NREL 5MW simulations:

- For the NREL 5MW it was found that the torque and pitch controller within FAST has a significant influence on the fastFoam simulations in the normal operating conditions case, see Section 6.2.1.1. The underlying reason could be attributed to the fact that the controller is tuned for BEM and the CFD simulations predict a larger magnitude of power (torque) to which the controller reacts. This reaction resulted in an unwanted pitching of the blades, which disturbed the comparison and thus it was decided to run the simulations again with the controller deactivated.
- In contrast to the phase VI, the 5MW represents a relatively modern turbine with more flexible blades. This is also represented in the simulations as the influence of the blade elasticity leads to increased deviations between the rigid blade assumption method OpenFOAM and the fastFoam method. This shows that for more flexible blades a rigid modelling approach may result in increased errors.
- With deactivated control good agreement between the different methods was observed in normal operating conditions, see Section 6.2.1.2. As only the aerodynamic modelling is replaced for fastFoam compared to FAST while the structural modelling is the same, this leads to relatively similar deflections along the blade span. For the aerodynamic loads it was found that the CFD methods resulted in a much smoother load distribution compared to FAST, which especially shows some irregularities of non-physical nature near the root for edgewise loading, see Figure 6.34b. Such an irregularity was also present in Heinz et al. [14] and could be attributed to a sudden change in airfoil data.
- Two points of attention were found. First of all, the pitching moment magnitude was about twice as large for the CFD methods. This could be an error or a difference in the load mappings from blade surface in CFD to beam nodes compared to FAST. In addition, power, torque and thrust fluctuations of unresolved origin were again present for the fastFoam approach.
- Finally, the 5MW simulations in extreme operational conditions such as heavy yaw (Section 6.2.2) showed that reasonable aero-servo-elastic simulations with activated controller can be achieved for the coupled method if the overall loads due to yawing are reduced. This confirmed the initial reasoning that the controller reacts due to an overprediction of the CFD loads compared to BEM loads in FAST in the normal operating case. A trend that the BEM in FAST underestimates the aerodynamic loads compared to the higher fidelity CFD is existent. The CFD methods show relatively good agreement if only one azimuth is observed, but if a full rotor rotation is considered clear differences are present attributed to the different blade representations (rigid versus elastic). In addition, an overall reduction in loads is present similar to the phase VI such as given by theory.

Finally, regarding the main research questions it can be stated that it was found that for improved accuracy a CFD method may deliver better results compared to BEM especially in extreme operational conditions such as the simulated heavy yaw or unsteady pitching motions. Moreover, for more modern turbines, such as the NREL 5MW, the modelling of flexible blades is inevitable and must be included. This holds especially true if an unsteady inflow is regarded. Addressing futuristic turbines with highly flexible blades, which may include exotic configurations such as flaps and slats, this may get even more important if a high accuracy is desirable. For BEM such a modelling would possibly require the addition of new engineering add-ons to include such configurations and thereby an additional factor of uncertainty.

However, for normal operating conditions it was found that the BEM based FAST method showed a relatively good result in comparison to the CFD and especially also the experimental data for the

phase VI. In addition, the greatest advantage of BEM is its low computational time compared to the CFD methods, see Section 6.3. The large computational effort for the CFD methods really prohibit its usage for a large number of simulations such as given by the IEC requirements. Therefore, the CFD methods are restricted for specific applications such as the detailed design phase or research activities. It can also be stated that BEM can benefit from these high fidelity simulations and especially also measurement campaigns, such as the phase VI, in order to improve its accuracy within the underlying engineering add-ons.

The implemented aero-servo-elastic wind turbine simulation method fastFoam, which is based on CFD, is a first step towards a method combining all the three aspects including highest fidelity aerodynamic modelling. The influence of the controller, which is often neglected together with the elasticity in CFD only simulations, was found to be significant for the 5MW turbine. Therefore, it can be underlined that for an approach where accuracy is most desired and computational effort is not valued highly an aero-servo-elastic method with the highest fidelity models may deliver best results.

7.2. Recommendations

Next, a few recommendations are formulated based on the implemented method and the resulting simulations. First of all, it can be stated that all simulations were based on the ElastoDyn structural model in FAST. However, the existent higher fidelity BeamDyn structural model, which includes the torsional DOF, may further improve the results. For the NREL 5MW BeamDyn input files exist including full 6x6 stiffness and mass matrices. In the coupling the possibiliy to include BeamDyn, which has different beam node positions compared to ElastoDyn, was added including load mappings tailored for BeamDyn.

However, due to the reason that the time step of BeamDyn is approximately one fifth of the ElastoDyn and CFD time step it was not possible to run simulations based on BeamDyn. This can be explained as the CFD time step would need to be decreased as well and the simulations would take about five times as long, thus an enormous amount of computational time would be required. Due to this it was impractical and discarded within this first version. However, there may be solutions to this problem which would allow to use the higher fidelity BeamDyn structural model. A solution would be subcycling, which is both supported by MpCCI and FAST.

A subcycling within MpCCI indicates that the codes to be coupled run with different time steps. Thus, the CFD time step would be five times larger compared to the FAST (BeamDyn) time step. The loads from CFD would then be applied as a constant over five time steps in FAST. In addition, also subcycling in FAST itself could be a possibility, where the time step of FAST equals the initial time step, but only the BeamDyn time step is reduced by a factor of five. This is also in principle supported by FAST, but with an initial test no convergence was obtained following this approach. Thus further work would be required to use BeamDyn within the coupled fastFoam simulations.

In addition, some unexplained discrepancies were found for the pitching moment especially for the NREL 5MW simulations. Thus it is advised to further investigate why the pitching moment between the BEM and CFD methods deviates that much. An initial explanation was given by the reasoning that FAST uses only one moment arm where the resultant aerodynamic loads are acting compared to the CFD, which includes several moment contributions from each faces on the blade surface related to one beam node. Thus, there could be a discrepancy in the load mappings from two dimensional surfaces to one dimensional beam nodes. By running an additional simulation, where only the forces are exchanged within the coupling and the moments are set to zero it was found that the moment contributions have nearly no effect on the result. However, why the pitching moments deviate needs to be addressed in more detail.

Moreover, for all simulations with fastFoam fluctuations were found in the torque, pitch and thrust outputs given by FAST. The origin of these fluctuations could not be fully explained. Including the output from OpenFOAM for the coupled fastFoam simulations these fluctuations were either not present or much smaller in magnitude. Therefore, it is advised that for an improved version of the coupling the origin of these fluctuations is researched. An influence could be the complex drivetrain modelling within ElastoDyn, but it is questionable why only the coupled version with the loads from CFD shows this behaviour, whereas the BEM based FAST does not. However, the mean values still show reasonable results, which indicate that the exchanged load magnitudes from the CFD to FAST are acceptable. A comparison between the mapped loads in ElastoDyn both from AeroDyn and CFD also confirms

this. This could still be a programmatic fault, which only occurs for the coupled version. Due to the reason that it is very difficult to find the exact origin of these fluctuations, as a converged solution of the coupled simulation would have to be much further investigated, this is accepted at the current state of the coupled method.

Besides the previously mentioned fluctuations, there were problems with the controller for the 5MW turbine in normal operating conditions due to an increased value in the power (torque) for the CFD methods compared to BEM. Such an increase was not present for the phase VI, where an unsteady RANS model with $k - \omega$ SST modelling was applied compared to the DES approach with Spalart-Allmaras for the 5MW. Thus it would be advisable to also test a RANS only model for the 5MW turbine to see if the torque and power reduce and obtain a magnitude similar to the BEM results and if this has an effect on the control reaction. However, this difference could also be a result of the different aerodynamic modelling approaches and thus a mitigation may be difficult to achieve. A retuning of certain controller variables such as the required aerodynamic power may also be possible.

Further improvements may also be achievable through optimisation of the convergence methods, by selecting different times for the specific methods which were applied to improve the convergence such as shown in Figure 5.9 and Table 5.1. Thereby a better overall convergence may be obtainable. Also the computational time may be further reduced for fastFoam, especially in comparison to an OpenFOAM simulation by better scaling of some components with updated number of cores, see Section 6.3. For the 5MW simulations it could be argued that the optimal number of cores was probably not found for the three simulations done, thus some computational time may be saved.

In addition to the simulated NREL phase VI and 5MW turbine, it would be interesting to simulate a futuristic turbine such as the 10MW AdVanced Aerodynamic Tools of lArge Rotors (AVATAR) turbine for instance, see [36]. This could then show some additional major improvements of the fastFoam approach or possible new limitations. Due to the limited project time this could not be achieved, but would be an interesting task for the future.

Finally, the modelling of flaps and slats could be included into the fastFoam method. This was already achieved by Fraunhofer IWES in a different project, where the motion of the flaps or slats is represented in the CFD mesh. For a BEM method an entire new engineering model would be required for such a modelling, thus for this specific approach a CFD method seems to be more feasible. Then the modelling accuracy for these exotic configurations could be addressed.

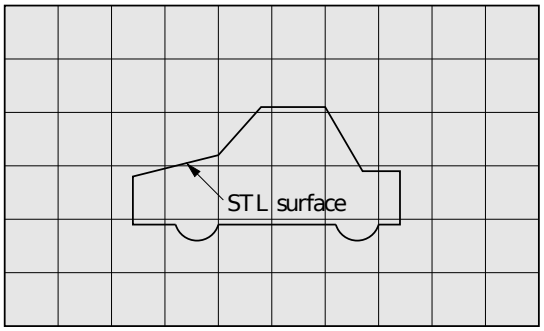
A

Meshing in OpenFOAM

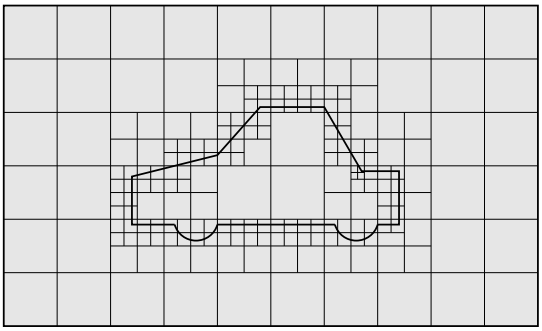
This Appendix contains an example for a blockMeshDict (Figure A.1) and an overview of the snappyHexMesh meshing procedure, see Figure A.1.

Listing A.1: Example of a blockMeshDict for generating a single block with blockMesh [41].

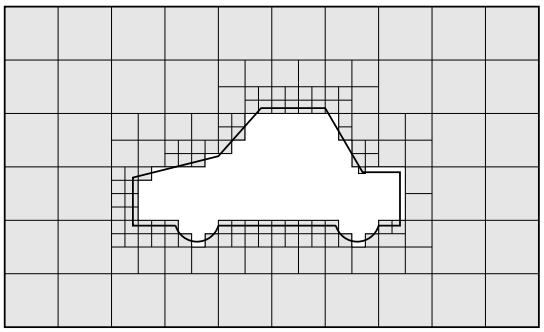
```
vertices
(
    ( 0    0    0   )    // vertex number 0
    ( 1    0    0.1)    // vertex number 1
    ( 1.1  1    0.1)    // vertex number 2
    ( 0    1    0.1)    // vertex number 3
    (-0.1 -0.1  1   )    // vertex number 4
    ( 1.3  0    1.2)    // vertex number 5
    ( 1.4  1.1  1.3)    // vertex number 6
    ( 0    1    1.1)    // vertex number 7
);

blocks
(
    hex (0 1 2 3 4 5 6 7)    // vertex numbers
    (10 10 10)               // numbers of cells in each direction
    simpleGrading (1 2 3)    // cell expansion ratios
);

edges
(
    arc 1 5 (1.1 0.0 0.5)
);

boundary                // keyword
(
    inlet               // patch name
    {
        type patch;     // patch type for patch 0
        faces
        (
            (0 4 7 3)  // block face in this patch
        );
    }                   // end of 0th patch definition
    outlet              // patch name
    {
        type patch;     // patch type for patch 1
        faces
        (
            (1 2 6 5)
        );
    }
    walls
    {
        type wall;
        faces
        (
            (0 1 5 4)
            (0 3 2 1)
            (3 7 6 2)
            (4 5 6 7)
        );
    }
);
```



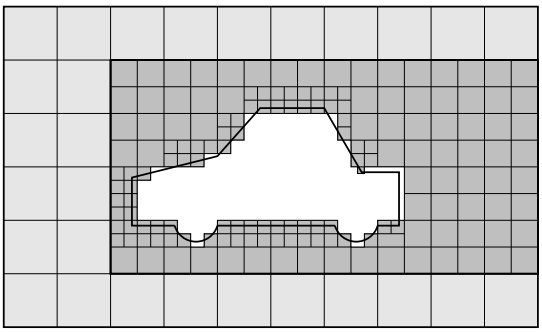
(a) The starting mesh and the surface geometry.



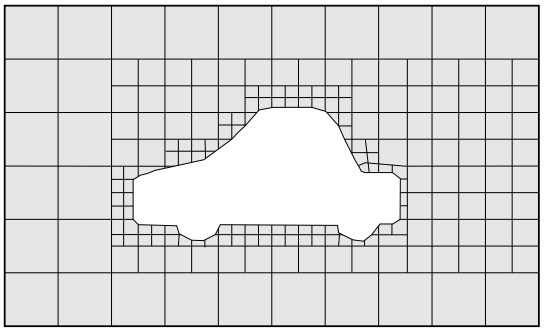
(b) Cell splitting near the surface.



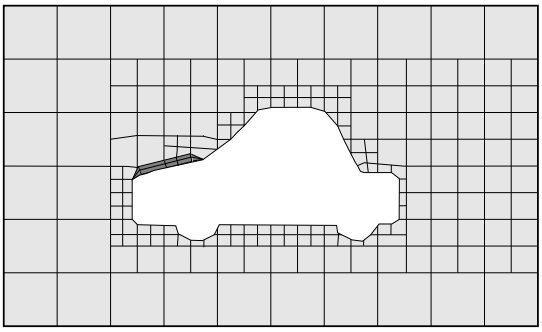
(c) Removal of cells.



(d) Additional Cell splitting by region.



(e) Snapping of near surface cells to surface.



(f) Adding of mesh layers

Figure A.1: The snappyHexMesh meshing process (Figures taken from [41], Figure (a) modified)

B

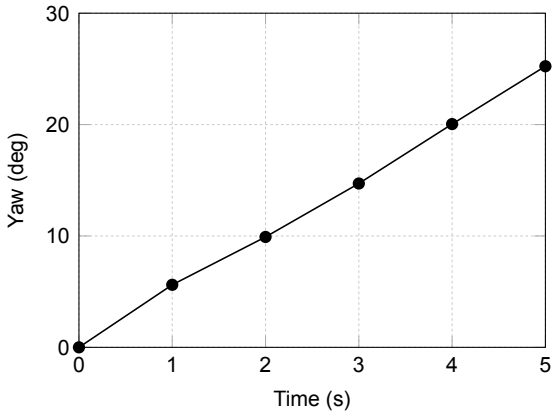
Solid Body Mesh Motion

This Appendix includes an example for the dynamicMeshDict (Figure B.1) as well as the turbine angles shown in Figure B.1, which correspond to the mesh rotations in Figure 4.16.

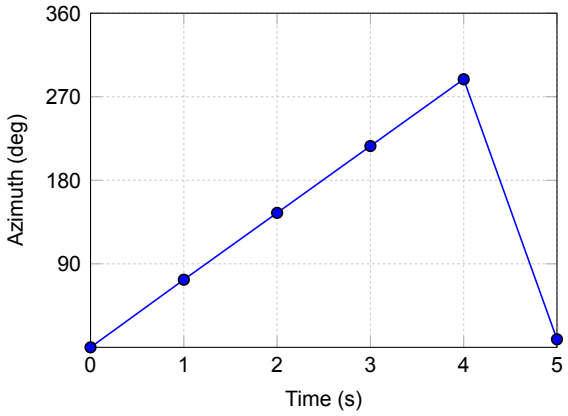
Listing B.1: Example of a dynamicMeshDict used for yaw, torque and pitch motions.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dynamicFvMesh   turbineSolidBodyMotionFvMesh;

motionSolverLibs ( "libfvMotionSolvers.so" );

turbineSolidBodyMotionFvMeshCoeffs
{
    // hierarchy of the motion to be applied. The first one is the master
    // other are considered as slave motion applied afterwards.
    hierarchy   (0 1 2 3);

    startTime       0.0;

        FIELD
    {
            turbineSolidBodyMotionFunction  turbineControlMotion;
        turbineControlMotionCoeffs
        {
                origin        (0 0 0);
                axis          (0 0 1);

            mode         yaw;

            modeCoeffs
            {

                    alignmentZero   0.0;
                actionCurve table ((0 0)(30 20));
                // Entries have format: (time yaw_angle)
            }
        }
    }

    ROTOR
    {
            turbineSolidBodyMotionFunction  turbineControlMotion;
        turbineControlMotionCoeffs
        {
                origin        (0 0 0);
                axis          (1 0 0);

            mode         torque;

            modeCoeffs
```

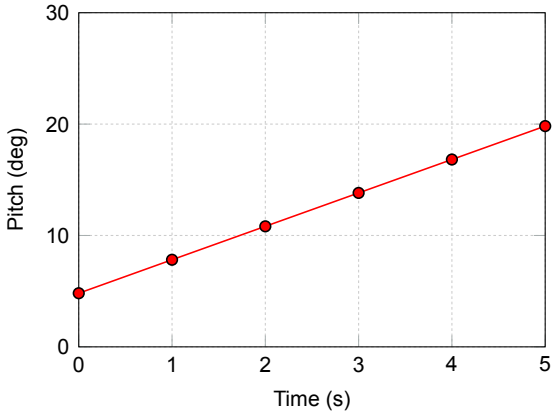
```
            {
                azimuthZero 0.0;
                actionCurve table ((0 7.54)(30 7.54);
                // Entries have format: (time rotational_speed)
            }
        }
    }

    BLADE0
        {
                turbineSolidBodyMotionFunction  turbineControlMotion;
                turbineControlMotionCoeffs
                {
                        origin          (0 0 0);
                        axis            (0 0 -1);

            mode            pitch;

            modeCoeffs
                        {
                pitchZero   0.0;
                actionCurve     table ((0 0)(30 0));
                // Entries have format: (time pitch_angle)
                lastPitchAction 0;
                        }
        }
        }

    BLADE1
        {
                turbineSolidBodyMotionFunction  turbineControlMotion;
                turbineControlMotionCoeffs
                {
                        origin          (0 0 0);
                        axis            (0 0 1);

            mode            pitch;

            modeCoeffs
                        {

                pitchZero   0.0;
                actionCurve     table ((0 0)(30 0));
                lastPitchAction 0;
                        }
        }
        }
}
```



(a) Yaw angle.



(b) Azimuth angle.



(c) Pitch angle.

Figure B.1: The yaw, azimuth and pitch angles corresponding to the mesh rotations in Figure 4.16.

C

Turbine Coordinate Systems

This Appendix includes the coordinate systems used throughout the simulations shown in Figure C.1.

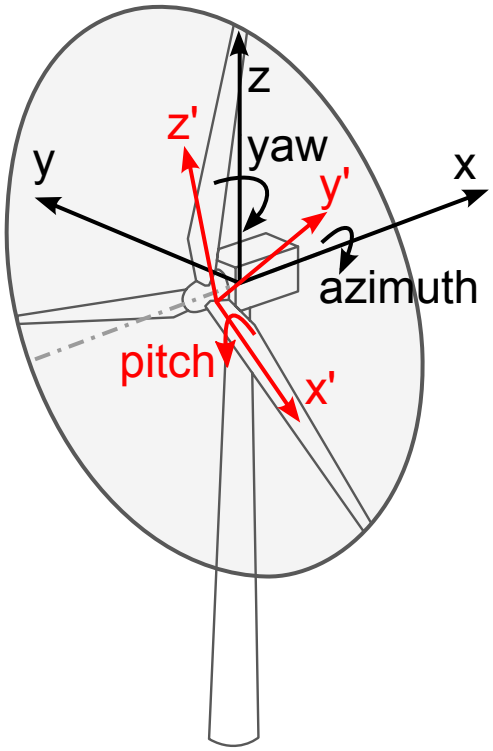


Figure C.1: The global coordinate system (black) and the blade coordinate system (red).

D

MpCCI Workflow

This Appendix contains an overview of the MpCCI workflow, see Figure D.1.

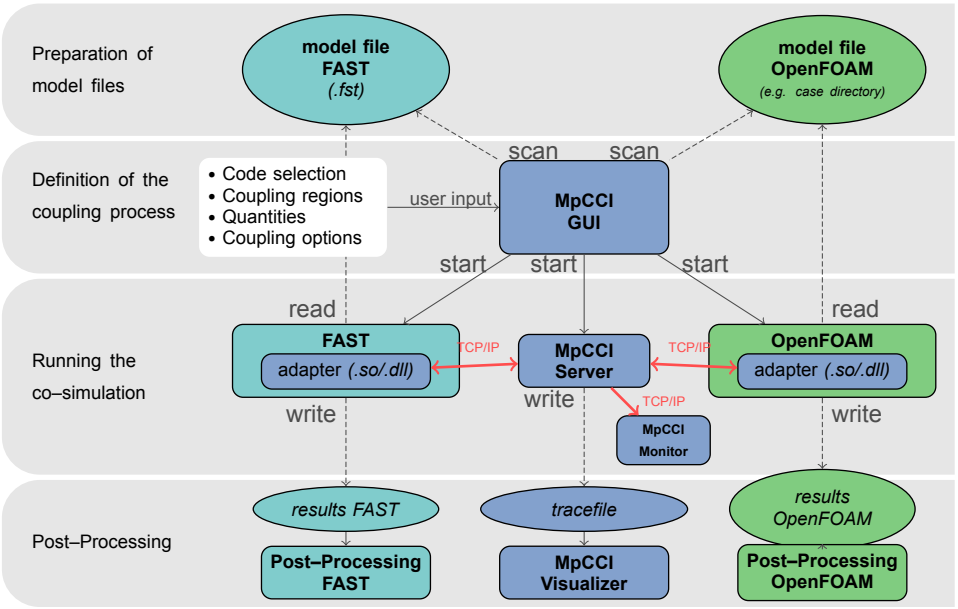


Figure D.1: Overview of the MpCCI workflow.

E

The fastFoam Solver

This Appendix contains the beamMeshDict (Listing E.1) and the beamElementControlDict (E.2) used for the mesh movements and the main part of the fastFoam solver, see Listing E.3.

Listing E.1: Example of the beamMeshDict used for case 6 (c).

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      beamMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//- General mesh settings

AnalysisType    NonLinear;    // Linear NonLinear

ElementType    GEBT;    // GEBT

NumberOfNodes    2;    // per element: Isoparametric/GEBT element: 2,3,4

ReferenceOrientation    // reference orientation of the beam elements
(
            1 0 0
            0 1 0
            0 0 1
);


//- Mesh element settings

RotorCenter    (0 0 0);
RotationAxis    (1 0 0);

ApplyTwist      true;
ApplyPitch      false;
ApplyTilt       true;
ApplyCone       false;
ApplyYaw        true;

TiltDict
{

    //- Tilt Angle [°]
    TiltAngle       5;

    //- Tilt Center for Rotation
    TiltCenter      (0 0 0);

}

ConeDict
{

    //- Cone Angle [°]
    ConeAngle       -2.5;
```

```
}

YawDict
{

    //- Yaw Angle [°]
    YawAngle          30;

}


//- MeshPoints
Points
(
(0 0 1.5)
(0 0 3.3088)
(0 0 6.9265)
(0 0 10.544)
(0 0 14.162)
(0 0 17.779)
(0 0 21.397)
(0 0 25.015)
(0 0 28.632)
(0 0 32.25)
(0 0 35.868)
(0 0 39.485)
(0 0 43.103)
(0 0 46.721)
(0 0 50.338)
(0 0 53.956)
(0 0 57.574)
(0 0 61.191)
(0 0 63)
);


//- TwistAngles
TwistAngles
(
-13.3080907075
-13.3080907075
-13.3080907075
-13.3080907075
-12.5454838822
-10.7698876751
-9.7568982933
-8.73130384
-7.6472676916
-6.5489075983
-5.4978451711
-4.462378655
-3.4818817097
-2.692523485
-1.9929821242
-1.2923521435
-0.5975147662
-0.1406634305
0
);
```

Listing E.2: Example of the beamElementControlDict used for case 6 (c).

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      beamElementControlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Basic settings

NumberOfBlades      3;

// Patch settings

BladeSettings
{
    Blade_0
    {
        Name        Blade0;
        SurfacePatches  (BLADE0);   // Can also be more than one
        AMIPatches          (PITCH_AMI_OUT_0 PITCH_AMI_IN_0);
        RotationalOffset    0;      // Rotational offset to given beam mesh
        CorrectAMI          false;

    }

    Blade_1
    {
        Name        Blade1;
        SurfacePatches  (BLADE1);   // Can also be more than one
        AMIPatches          (PITCH_AMI_OUT_1 PITCH_AMI_IN_1);
        RotationalOffset    120;        // Rotational offset to given beam mesh
        CorrectAMI          false;
    }

    Blade_2
    {
        Name        Blade2;
        SurfacePatches  (BLADE2);   // Can also be more than one
        AMIPatches          (PITCH_AMI_OUT_2 PITCH_AMI_IN_2);
        RotationalOffset    240;        // Rotational offset to given beam mesh
        CorrectAMI          false;
    }
}

WriteSets       false;

// Mesh motion settings

MeshMotionMethod        MasterSlaveCubic;
```

```
RestrictMotionToZone     false;

MasterSlaveCoeffs
{
InnerCellDistance        0.2;

OuterCellDistance        15;

MinimalCellDistance      1;
}


//- Output settings

writeMeshVTK             true;

writeBeamOutput          true;

writeEndDisplacementList     true;

writeNodeForces          true;

writeRootBendingMoments  true;

writeRuntimeBeamVTKs     true;

EmergencyMeshWrite   true;

IncludeGravity           false;

GravitySettings
{
    GravityConstant          9.81;
    GravityDirection         (0 0 -1);
}


// Rotation settings

IncludeRotation          false;

RotationAxis             (0.8627299156628210 0.4980973490458728 -0.08715574274765818);

Omega        1.2671090355;        // rad/s

// Solver settings

BeamAnalysisType     Dynamic;     // Static, Dynamic

Solver               Alpha;       // NewtonRaphson, Alpha

NewtonRaphsonSettings
{
    NoLS         8;  // number of load steps
}


InitializationTime   0.05;

LinearSmoothing          true;
```

```
AzimuthalCorrection      false;

DisplacementTolerance        1e-8;

DampingSettings          // Rayleigh Damping
{
   alpha         0.001;
   beta          0.001;
}

// Coupling settings

CouplingType         FixedCoupling;   // ResidualCoupling

FixedCouplingCoeffs
{
   CouplingInterval      1;
}

ForceProjectionMethod        NearestNeighbor;     // InterpolatedProjection

// Relaxation Settings

RelaxationType       FixedRelaxation;     // LinearRelaxation

FixedRelaxationCoeffs
{
   RelaxationFactor      1;
}


// Force calculation settings

ForceFactor      1;

RhoFluid         1.225;

ForceTolerance       1e-12;

HubAttachmentPoint       (0 0 1.656);   // Hub (patch) attachment point on beam axis
```

Listing E.3: The main part of the implemented solver fastFoam.C.

```
1
2  #include "argList.H"
3  #include "Time.H"
4
5  #include "fvCFD.H"
6  #include "turbineSolidBodyMotionFvMesh.H"
7  #include "singlePhaseTransportModel.H"
8  #include "turbulentTransportModel.H"
9
10
11 #include "fvMesh.H"
12 #include "pimpleControl.H"
13 #include "CorrectPhi.H"
14 #include "fvIOoptionList.H"
15 #include "fixedFluxPressureFvPatchScalarField.H"
16
17 #include "FASTBeamAnalysis.H"
18 #include "TurbinePost.H"
19 #include <fstream>
20
21 #include "mpcciFunctionObject.H"
22
23 using namespace Foam;
24
25 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
26 int main(int argc, char *argv[])
27 {
28     #include "setRootCase.H"
29     #include "createTime.H"
30     #include "createNamedDynamicFvMesh.H"
31     #include "initContinuityErrs.H"
32
33     pimpleControl pimple(mesh);
34
35     #include "createFields.H"
36     #include "createUf.H"
37     #include "createMRF.H"
38     #include "createFvOptions.H"
39     #include "createControls.H"
40     #include "CourantNo.H"
41     #include "setInitialDeltaT.H"
42
43     double lastTimeStep=0;
44     int    currentStartUpStep=0;
45     int    startIterationCounter=0;
46     bool   startUpActive=false;
47
48     //turbulence->validate(); only OF4
49
50     #include "createBeam.H"
51     #include "createInputs.H"
52
53     // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
54
55     Info<< "Starting FAST Beam coupling" << endl;
56
57     turbineSolidBodyMotionFvMesh* meshp=static_cast<turbineSolidBodyMotionFvMesh*>(&mesh);
58
59     initRot=meshp->getInitRotations();
60     Info<<"initial mesh Yaw, Azimuth, Pitch are ("<<initRot[0]<<", "<<initRot[1]<<", "<<initRot[2]<<",
61     "<<initRot[3]<<") "<< endl;
62
63     oldAzimuth=initRot[1];
64
65     // get pointer to the mpcci function object
66     runTime.functionObjects().start();
67     int i = runTime.functionObjects().findObjectID("MpCCI_functionObject_adapter_for_OpenFOAM");
68     mpcciFunctionObject* mpcci = static_cast<mpcciFunctionObject*>(&(runTime.functionObjects()[i]));
69     mpcci->setBladePosPtr      (&blade_position[0]);
70     mpcci->setBladeOriPtr      (&blade_orientation[0]);
71     mpcci->setBladeVelPtr      (&blade_velocity[0]);
72     mpcci->setBladeAvelPtr     (&blade_angular_velocity[0]);
73     mpcci->setBladeForcePtr    (&blade_forces[0]);
74     mpcci->setBladeMomentPtr   (&blade_moments[0]);
75     mpcci->setBladeRootPosPtr  (&bladeRoot_position[0]);
76     mpcci->setBladeRootOriPtr  (&bladeRoot_orientation[0]);
77     mpcci->setBladeRootVelPtr  (&bladeRoot_velocity[0]);
78     mpcci->setBladeRootAvelPtr(&bladeRoot_angular_velocity[0]);
79     mpcci->setRotorPosPtr      (&rotor_position);
80     mpcci->setRotorOriPtr      (&rotor_orientation);
81     mpcci->setRotorVelPtr      (&rotor_velocity);
82     mpcci->setRotorAvelPtr     (&rotor_angular_velocity);
```

```
83      mpcci->setAzimuthPtr       (&azimuth);
84      mpcci->setYawPtr           (&yaw);
85      mpcci->setPitchPtr         (&pitch[0]);
86
87      while (runTime.run())
88      {
89          #include "readControls.H"
90          #include "CourantNo.H"
91          #include "setDeltaT.H"
92          #include "startUpTimeStepControl.H"
93
94          runTime++;
95
96          Info<< "Time = " << runTime.timeName() << " sec; Time step size: "<< runTime.deltaTValue()<<"
97          sec!"<< nl << endl;
98
99          #include "modifyInputs.H"
100
101         // Set Yaw, Azimuth and Pitch angles
102         meshp->setrot(&angles);
103
104         scalar T0=runTime.elapsedCpuTime();
105         // Update mesh to given Yaw, Azimuth and Pitch angles
106         mesh.update();
107         scalar T1=runTime.elapsedCpuTime();
108         Info<<"Mesh update to yaw azimuth and pitch finished within: " << T1-T0 << "sec" <<  nl << endl;
109
110         //runTime.writeNow(); // uncomment to write the rigid body motion mesh
111
112         // Get current mesh points
113         pointField updatedMeshPoints=mesh.points();
114
115         // Quaternion describing yaw rotation
116         vector yawRotation;
117         yawRotation.x() = 0;
118         yawRotation.y() = 0;
119         yawRotation.z() = angles[0];
120         Quaternion yawQuat(Quaternion::rotationSequence(Quaternion::XYZ), yawRotation);
121         yawQuat.normalize();
122
123         rotationAxis=Beam[0]->getRotationAxis();
124         Info<<"Original RotationAxis is  =("<<rotationAxis<<") "<< endl;
125
126         rotationAxisUpdated = yawQuat.R()&rotationAxis;
127         //- Normalize rotation axis
128         rotationAxisUpdated=rotationAxisUpdated/(1.0*mag(rotationAxisUpdated));
129         Info<<"Updated RotationAxis is  =("<<rotationAxisUpdated<<") "<< endl;
130
131
132         scalar tT0=runTime.elapsedCpuTime();
133
134         total_disp.zeros();
135
136         // Start Elasticity related mesh motion by looping over blades
137         for (int i=0; i< numberOfBlades ; i++)
138         {
139             List <point> initPoints=Beam[i]->getMeshPointsInitial();
140             double bladeRot=Beam[i]->getRotationalOffset();
141             Info<<"Blade "<<i<<" bladeRot is  =("<<bladeRot<<") "<< endl;
142
143             // Rotation of mesh points due to yaw, torque, pitch
144             // TODO: fix this for different reference systems
145
146             // Quaternion describing azimuthal rotation (from blade 0 position)
147             Quaternion azimuthQuat(rotationAxis, FASTAngles[1]+bladeRot);
148             azimuthQuat.normalize();
149
150             // Quaternion describing pitch rotation
151             vector pitchAxis (frenetFramesInitial[0].xx(),frenetFramesInitial[0].yx(),
152             frenetFramesInitial[0].zx());
153
154             Quaternion pitchQuat(pitchAxis, -FASTAngles[2+i]);
155             pitchQuat.normalize();
156
157             // Quaternion describing rigid rotation
158             Quaternion rigidQuat = yawQuat*azimuthQuat*pitchQuat;
159             rigidQuat.normalize();
160
161             Field <point> rigidPoints=rigidQuat.R() & initPointsBlade0;
162
163             vector rigidRotation=rigidQuat.eulerAngles(Quaternion::rotationSequence(Quaternion::XYZ));
164
165             Quaternion bladeRotQuat(rotationAxis, bladeRot);
166             bladeRotQuat.normalize();
```

```
167             Quaternion bladeRotQuatNeg=inv(bladeRotQuat);
168             bladeRotQuatNeg.normalize();
169             vector bladeRotationNeg=bladeRotQuatNeg.eulerAngles(Quaternion::rotationSequence(
170             Quaternion::XYZ));
171
172             Info<<"Blade "<<i<<" rotations to zero (x,y,z) of node =("<<bladeRotationNeg.x()<<",
173             "<<bladeRotationNeg.y()<<", "<<bladeRotationNeg.z()<<") "<< endl;
174
175             for (int j=0; j< size/6; j++)
176                 {
177                     int k=6*j;
178
179                     // Rigid body motion part
180
181                     // Calculate displacements to move initial blade i position to initial blade 0 position
182                     disp_bladeRot(k)=positions_init(k)-initPoints[j].x();
183                     disp_bladeRot(k+1)=positions_init(k+1)-initPoints[j].y();
184                     disp_bladeRot(k+2)=positions_init(k+2)-initPoints[j].z();
185                     disp_bladeRot(k+3)=bladeRotationNeg.x();
186                     disp_bladeRot(k+4)=bladeRotationNeg.y();
187                     disp_bladeRot(k+5)=bladeRotationNeg.z();
188
189
190                     // Calculate displacements to move initial blade i to its rigid body position from
191                     // initial blade 0 position
192                     disp_rigid(k)=rigidPoints[j].x()-positions_init(k);
193                     disp_rigid(k+1)=rigidPoints[j].y()-positions_init(k+1);
194                     disp_rigid(k+2)=rigidPoints[j].z()-positions_init(k+2);
195                     disp_rigid(k+3)=rigidRotation.x();
196                     disp_rigid(k+4)=rigidRotation.y();
197                     disp_rigid(k+5)=rigidRotation.z();
198
199                     // Elasticity related motion part
200
201                     if ((runTime.timeOutputValue() > InitializationTime && linSmoothing==false)
202                      || (runTime.timeOutputValue() > startSmooth && linSmoothing==true))
203                     {
204                         // Quaternion describing the rotation due to twist prebend etc.
205                         vector twistRotation;
206                         twistRotation.x() = positions_init(k+3);
207                         twistRotation.y() = positions_init(k+4);
208                         twistRotation.z() = positions_init(k+5);
209                         Quaternion twistQuat(Quaternion::rotationSequence(Quaternion::XYZ), twistRotation);
210                         twistQuat.normalize();
211
212                         // Quaternion describing the total rotation to rigid body position
213                         Quaternion totQuat = rigidQuat*twistQuat;
214                         totQuat.normalize();
215
216                         // Quaternion describing the total rotation in actual position according to FAST
217                         vector actualRotation;
218                         actualRotation.x() = positions(k+3,i);
219                         actualRotation.y() = positions(k+4,i);
220                         actualRotation.z() = positions(k+5,i);
221                         Quaternion actualQuat(Quaternion::rotationSequence(Quaternion::XYZ), actualRotation);
222                         // actual rotation from FAST
223                         actualQuat.normalize();
224
225                         // Quaternion describing the rotation from rigid body position to actual position
226                         Quaternion elasticQuat=actualQuat*inv(totQuat);
227                         elasticQuat.normalize();
228                         vector elasticRotation=elasticQuat.eulerAngles(Quaternion::rotationSequence(
229                         Quaternion::XYZ));
230
231                         // Calculate displacements to move blade i from its rigid body position
232                         // to its actual position
233                         disp_elastic(k)=positions(k,i)-rigidPoints[j].x();
234                         disp_elastic(k+1)=positions(k+1,i)-rigidPoints[j].y();
235                         disp_elastic(k+2)=positions(k+2,i)-rigidPoints[j].z();
236                         disp_elastic(k+3)=elasticRotation.x();
237                         disp_elastic(k+4)=elasticRotation.y();
238                         disp_elastic(k+5)=elasticRotation.z();
239                     }
240                 }
241
242             // Update total displacements
243             total_disp.col(i) +=  disp_bladeRot+disp_rigid;
244
245             #include "printPositions.H"
246
247             // Move initial blade i beam mesh to initial blade 0 position
248             Beam[i]->updateBeam( disp_bladeRot );
249             // Move blade i beam mesh to its rigid body position from initial blade 0 position
250             Beam[i]->updateBeam( disp_rigid );
```

```
251             // Write rigid body beam mesh vtk
252             if (runTime.write()==true)
253             {
254             Beam[i]->writeRigidOutput(runTime);
255             }
256
257             if (runTime.timeOutputValue() > InitializationTime )
258             {
259                 if (linSmoothing==true)
260                 {
261                     if (runTime.timeOutputValue()>=startSmooth && runTime.timeOutputValue()<=endSmooth)
262                     {
263                         smoothVal = smoothSlope*(runTime.timeOutputValue()-startSmooth);
264                         disp_elastic = smoothVal*disp_elastic;
265                         Info << "Smoothing elastic deformations is activated with a smoothing factor
266                         of "<<smoothVal<<nl<<endl;
267                     }
268                     if (runTime.timeOutputValue()>=startSmooth)
269                     {
270                         // Move the fluid mesh from its rigid body position to its actual position
271                         // (elasticity)
272                         Beam[i]->moveMesh( runTime, mesh, disp_elastic, updatedMeshPoints );
273                         // Move blade i beam mesh to its actual position from its rigid body position
274                         Beam[i]->updateBeam( disp_elastic );
275                         // Update total displacements
276                         total_disp.col(i) +=  disp_elastic;
277
278                         if (runTime.write()==true)
279                         {
280                             for (int i=0; i< numberOfBlades ; i++)
281                             {
282                                 Beam[i]->writeOutput(runTime);
283                             }
284                         }
285                     }
286                 }
287                 else
288                 {
289                     Info << "Fluid-structure coupling activated! It was started at
290                     "<<InitializationTime<<" sec"<<nl<<endl;
291                     // Move the fluid mesh from its rigid body position to its actual position (elasticity)
292                     Beam[i]->moveMesh( runTime, mesh, disp_elastic, updatedMeshPoints );
293                     // Move blade i beam mesh to its actual position from its rigid body position
294                     Beam[i]->updateBeam( disp_elastic );
295                     // Update total displacements
296                     total_disp.col(i) +=  disp_elastic;
297
298                     if (runTime.write()==true)
299                     {
300                         for (int i=0; i< numberOfBlades ; i++)
301                         {
302                             Beam[i]->writeOutput(runTime);
303                         }
304                     }
305                 }
306             }
307
308         }
309
310         scalar tT1=runTime.elapsedCpuTime();
311         Info<<"Mesh update to elastic state finished within: " << tT1-tT0 << "sec" <<  nl << endl;
312
313         if ((runTime.timeOutputValue() > InitializationTime && linSmoothing==false)
314           || (runTime.timeOutputValue() > startSmooth && linSmoothing==true))
315         {
316             // Update devRhoReff
317             devRhoReff = turbulenceModel.devReff();
318             // Here the actual moving of the points obtained through moveMesh takes place
319             mesh.movePoints(updatedMeshPoints);
320         }
321
322         scalar t0=runTime.elapsedCpuTime();
323         Info<<"FSI and coupling finished within: " << t0-t1 << "sec" <<  nl << endl;
324
325         runTime.write();
326
327         // Calculate absolute flux from the mapped surface velocity
328         phi = mesh.Sf() & Uf;
329
330         if (mesh.changing() && correctPhi)
331         {
332             #include "correctPhi.H"
333         }
334
```

```
335         // Make the flux relative to the mesh motion
336         fvc::makeRelative(phi, U);
337
338         if (mesh.changing() && checkMeshCourantNo)
339         {
340             #include "meshCourantNo.H"
341         }
342
343         // --- Pressure-velocity PIMPLE corrector loop
344         while (pimple.loop())
345         {
346             #include "UEqn.H"
347
348             // --- Pressure corrector loop
349             while (pimple.correct())
350             {
351                 #include "pEqn.H"
352             }
353
354             if (pimple.turbCorr())
355             {
356                 laminarTransport.correct();
357                 turbulence->correct();
358             }
359         }
360
361         t1=runTime.elapsedCpuTime();
362
363         Info<<"Pimple loop finished within: " << t1-t0 << "sec" <<  nl << endl;
364
365         normDistances=Beam[0]->getNormDistances();
366
367         // Write forces into matrix
368         for (int i=0; i< numberOfBlades ; i++)
369         {
370
371             Beam[i]->calcForces(mesh, p, devRhoReff);
372
373             forces=Beam[i]->getForces();
374
375             for (int j=0; j< size/6; j++)
376             {
377                if (Pstream::myProcNo()==0 && runTime.timeOutputValue() > InitializationTime)
378                {
379
380                     int k=6*j;
381                     normForces[k] = forces[k]/normDistances[j];
382                     normForces[k+1] = forces[k+1]/normDistances[j];
383                     normForces[k+2] = forces[k+2]/normDistances[j];
384                     normForces[k+3] = forces[k+3]/normDistances[j];
385                     normForces[k+4] = forces[k+4]/normDistances[j];
386                     normForces[k+5] = forces[k+5]/normDistances[j];
387                     normForces[k+6] = forces[k+6]/normDistances[j];
388
389                     blade_forces[i][j].x() = normForces[k];
390                     blade_forces[i][j].y() = normForces[k+1];
391                     blade_forces[i][j].z() = normForces[k+2];
392                     blade_moments[i][j].x() = normForces[k+3];
393                     blade_moments[i][j].y() = normForces[k+4];
394                     blade_moments[i][j].z() = normForces[k+5];
395                     Info<<"Wrote forces and moments to the mpcci buffer"<< endl;
396                }
397                else
398                {
399                      blade_forces[i][j].x() = 0.0;
400                      blade_forces[i][j].y() = 0.0;
401                      blade_forces[i][j].z() = 0.0;
402                      blade_moments[i][j].x() = 0.0;
403                      blade_moments[i][j].y() = 0.0;
404                      blade_moments[i][j].z() = 0.0;
405                //}
406             }
407
408             #include "printForces.H"
409
410             // Reset beam mesh for next timestep (fluid mesh automatically resets)
411             Beam[i]->resetMesh();
412
413         }
414
415         Info<<"Force calculation finished"<< endl;
416
417         if (runtimeProcessing)
418         {
```

```
419             for (int i=0; i< numberOfBlades ; i++)
420             {
421                 PostProcess.updateBeamMesh( i, total_disp.col(i) );
422                 PostProcess.updateOmegaRot( i, omega, rotationAxisUpdated, FASTAngles[i+2]);
423
424                 #include "printPostDisps.H"
425
426             }
427
428             devRhoReff = turbulenceModel.devReff();
429
430             PostProcess.update( mesh, p, U, devRhoReff, runTime);
431
432             for (int i=0; i< numberOfBlades ; i++)
433             {
434                 PostProcess.updateBeamMesh( i, -total_disp.col(i) );
435             }
436
437         }
438
439
440         Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
441         << "  ClockTime = " << runTime.elapsedClockTime() << " s"
442         << "  LastStepTime = " << runTime.elapsedClockTime()-lastTimeStep<< " s"
443         << nl << endl;
444
445         lastTimeStep=runTime.elapsedClockTime();
446
447     }
448
449     Info<< "End\n" << endl;
450
451     return 0;
452 }
453
454
455 // ************************************************************************* //
```

F

MpCCI GUI

This Appendix includes several screenshots, which describe how to setup the coupling for fastFoam within the MpCCI GUI.

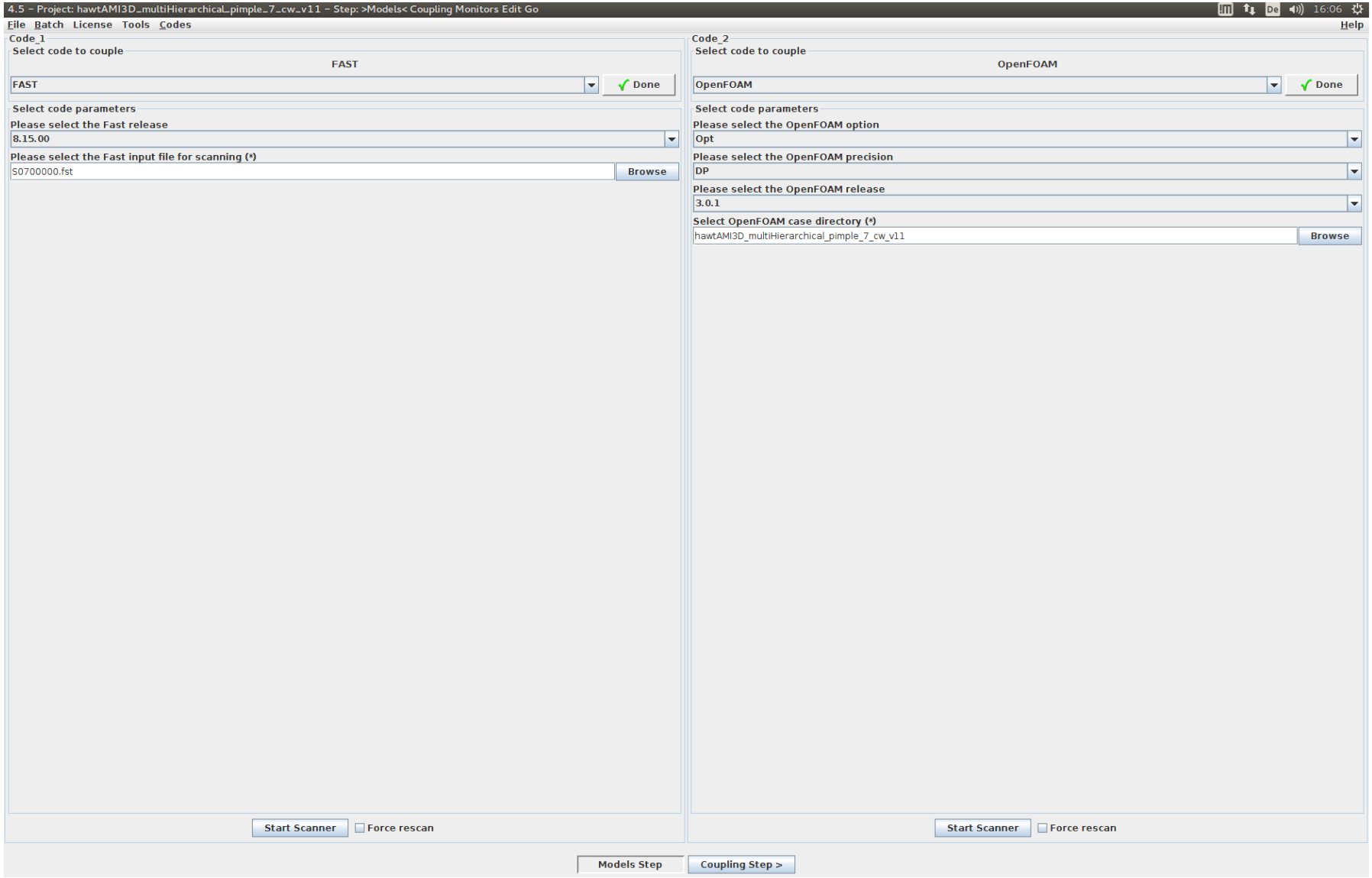


Figure F.1: The models step in the MpCCI GUI.

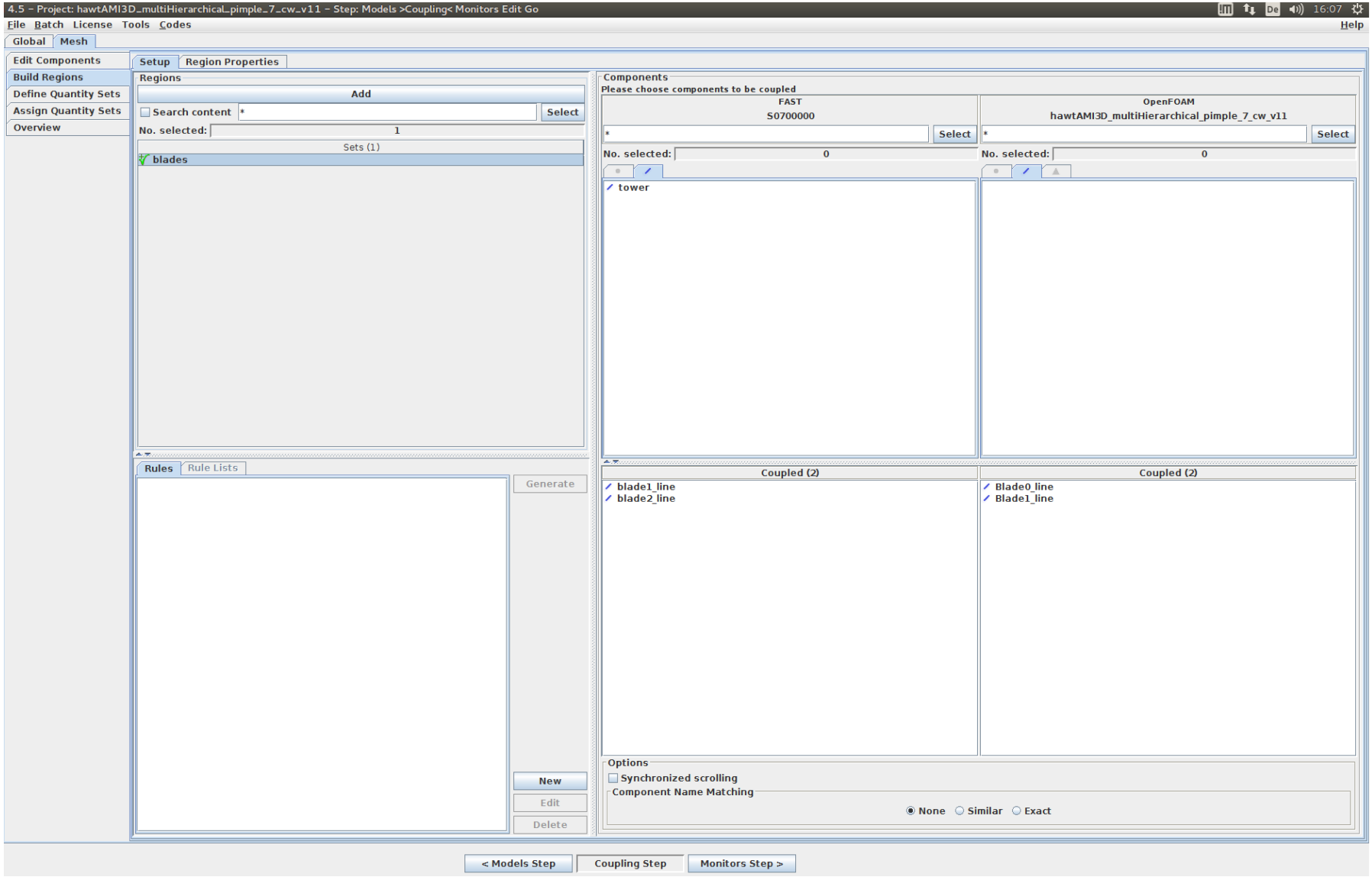


Figure F.2: The coupling step for the mesh variables in the MpCCI GUI.

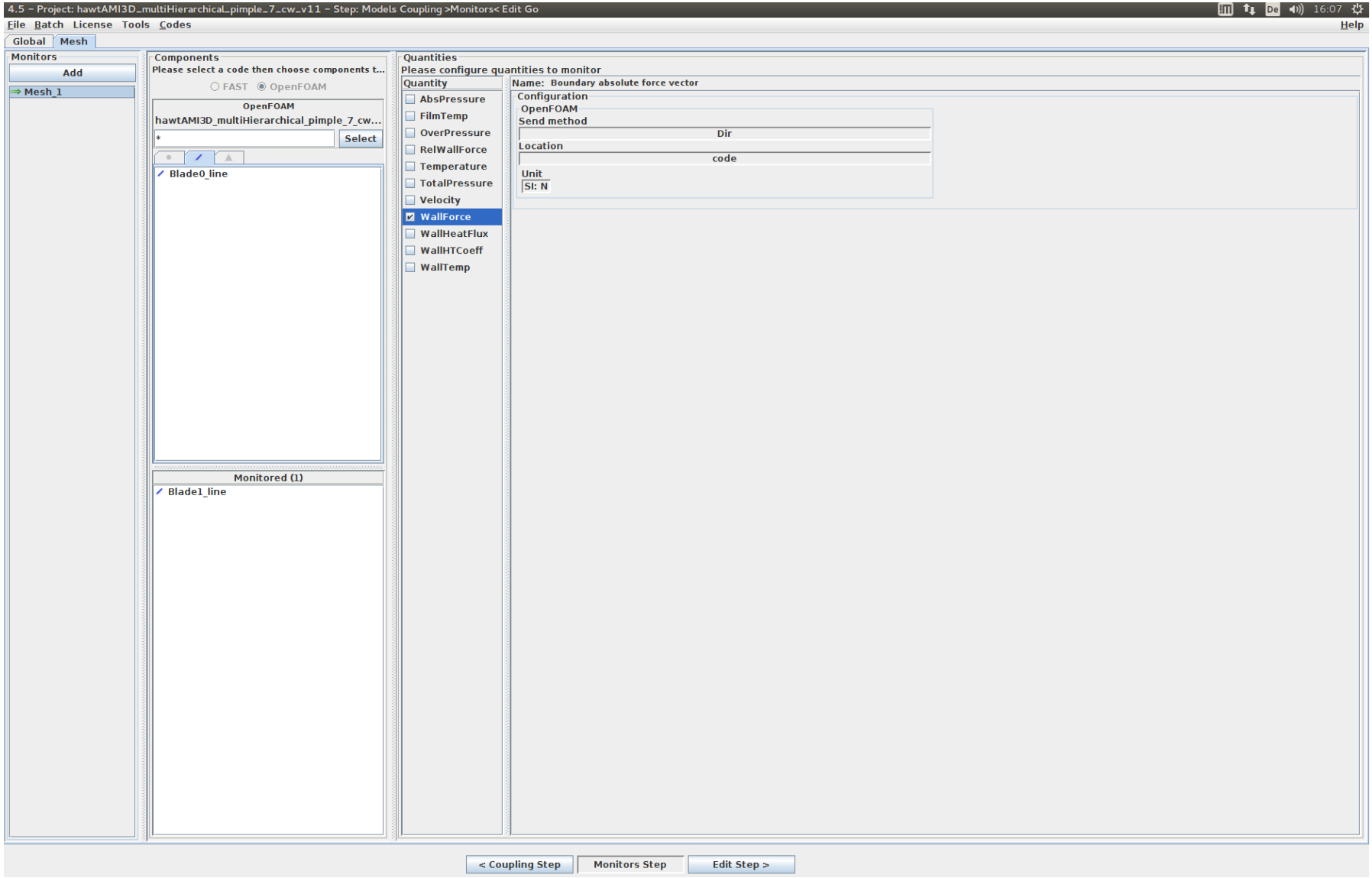


Figure F.3: The monitors step in the MpCCI GUI.

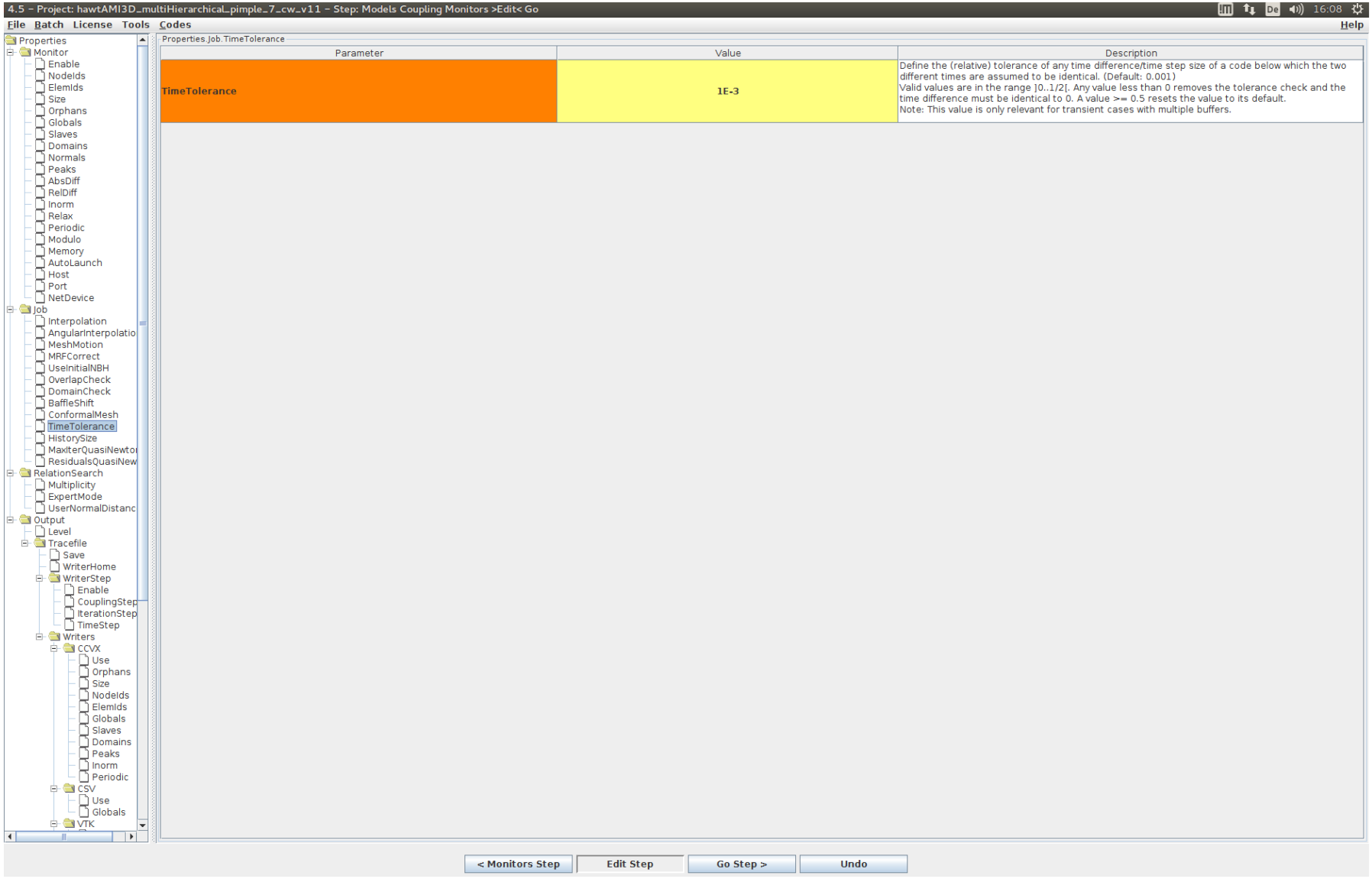


Figure F.4: The edit step in the MpCCI GUI.

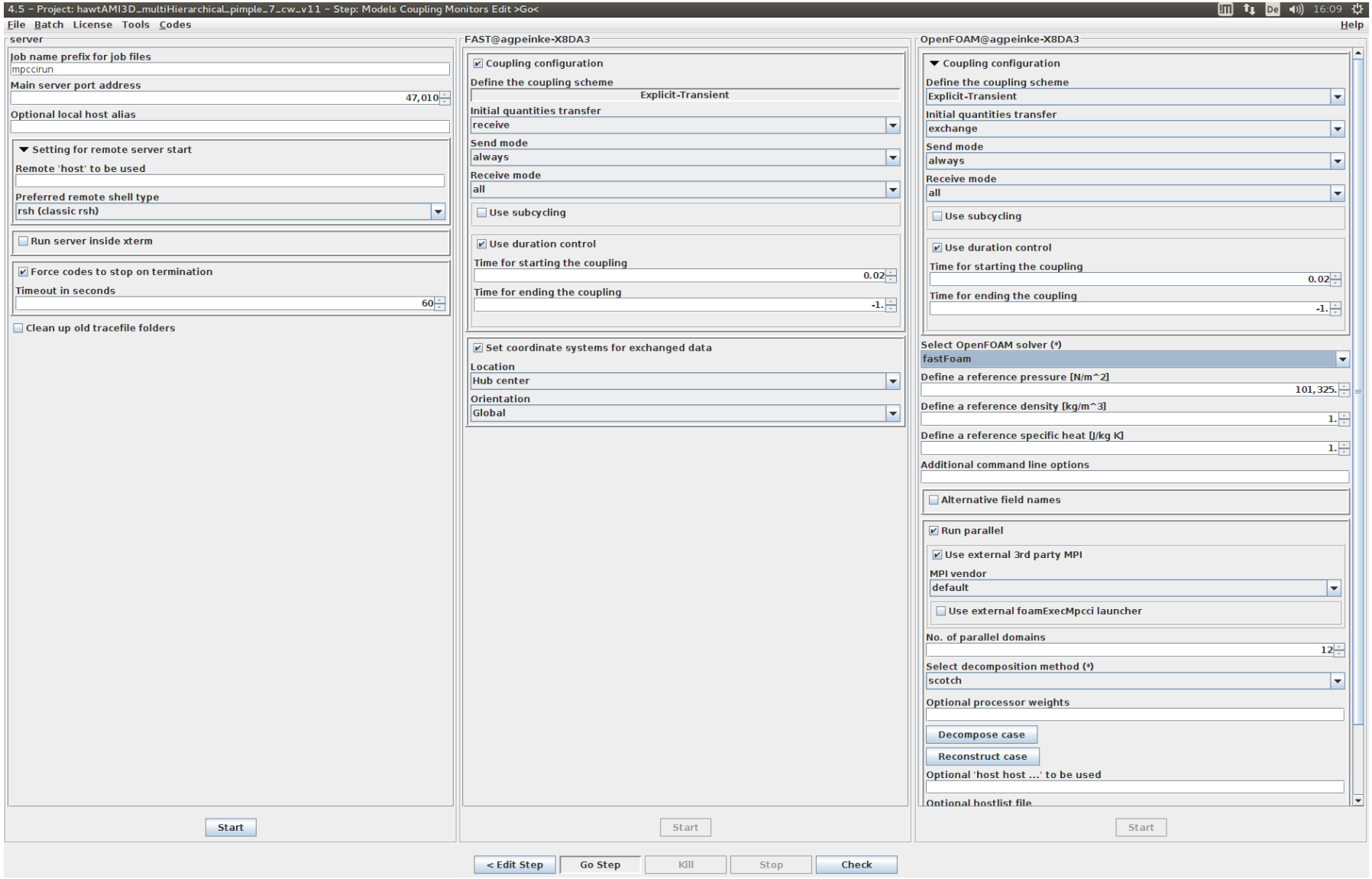


Figure F.5: The go step in the MpCCI GUI.

Bibliography

[1] T. Burton, N. Jenkins, D. Sharpe, and E. Bossanyi. *Wind Energy Handbook*. John Wiley & Sons, Chichester, West Sussex, 2nd edition, 2011. ISBN 978-0-470-69975-1.

[2] M. Carrion, R. Steijl, M. Woodgate, G. N. Barakos, X. Munduate, and S. Gomez-Iradi. Aeroelastic analysis of wind turbines using a tightly coupled CFD–CSD method. *Journal of Fluids and Structures*, 50:392–415, October 2014. ISSN 0889-9746. doi: 10.1016/j.jfluidstructs.2014.06.029. URL http://www.sciencedirect.com/science/article/pii/S0889974614001546. [Accessed on: 2016-07-15].

[3] D. Chandar and H. Gopalan. Comparative analysis of the arbitrary mesh interface(AMI), generalized grid interface (GGI) and overset methods for dynamic body motions in openFOAM. In *46th AIAA Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, June 2016. ISBN 978-1-62410-436-7. doi: 10.2514/6.2016-3324. URL http://arc.aiaa.org/doi/10.2514/6.2016-3324. [Accessed on: 2016-09-10].

[4] M. Churchfield and S. Lee. SOWFA | NWTC Information Portal, March 2015. URL https://nwtc.nrel.gov/SOWFA. [Accessed on: 2016-09-01].

[5] M. Churchfield, S. Lee, J. Michalakes, and P. J. Moriarty. A numerical study of the effects of atmospheric and wake turbulence on wind turbine dynamics. *Journal of Turbulence*, 13:N14, January 2012. ISSN 1468-5248. doi: 10.1080/14685248.2012.668191. URL http://www.tandfonline.com/doi/abs/10.1080/14685248.2012.668191. [Accessed on: 2016-09-02].

[6] D. Corson, D. Griffith, T. Ashwill, and F. Shakib. Investigating Aeroelastic Performance of Multi-Mega Watt Wind Turbine Rotors Using CFD. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. American Institute of Aeronautics and Astronautics, Honolulu, April 2012. ISBN 978-1-60086-937-2. URL http://arc.aiaa.org/doi/abs/10.2514/6.2012-1827. [Accessed on: 2016-09-01].

[7] I. Czajka, K. Suder-Dębska, and K. Jarosz. Modelling of an aerodynamic noise of a horizontal axis wind turbine using ansys/fluent and OpenFOAM packages. In *Proceedings of Forum Acusticum*, volume 2014-January, Krakow, 2014. ISBN 978-83-61402-28-2.

[8] E. Daniele. Wind turbine control in computational fluid dynamics with OpenFOAM. *Wind Engineering*, 41(4):213–225, July 2017. ISSN 0309-524X, 2048-402X. doi: 10.1177/0309524X17709724. URL http://journals.sagepub.com/doi/10.1177/0309524X17709724. [Accessed on: 2017-08-01].

[9] B. Dose, B. Stoevesandt, and J. Peinke. Studying the effect of blade deflections on the aerodynamic performance of wind turbine blades using OpenFOAM. In *Proceedings of DEWEK 2015*, Bremen, May 2015. DEWI. [Accessed on: 2016-09-28].

[10] B. Dose, H. Rahimi, I. Herráez, B. Stoevesandt, and J. Peinke. Fluid-structure coupled computations of the NREL 5mw wind turbine blade during standstill. *Journal of Physics: Conference Series*, 753(2):022034, 2016. ISSN 1742-6596. doi: 10.1088/1742-6596/753/2/022034. URL http://stacks.iop.org/1742-6596/753/i=2/a=022034. [Accessed on: 2017-11-14].

[11] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. ISBN 978-3-540-42074-3 978-3-642-56026-2. doi: 10.1007/978-3-642-56026-2. URL http://link.springer.com/10.1007/978-3-642-56026-2. [Accessed on: 2017-08-02].

[12] M. O. L. Hansen. *Aerodynamics of Wind Turbines*. Earthscan Ltd, London, 2nd edition, February 2008. ISBN 978-1-84407-438-9.

[13] M. O. L. Hansen and H. Aagaard Madsen. Review Paper on Wind Turbine Aerodynamics. *Journal of Fluids Engineering*, 133(11):114001–114001, October 2011. ISSN 0098-2202. doi: 10.1115/1.4005031. URL http://dx.doi.org/10.1115/1.4005031. [Accessed on: 2016-07-04].

[14] J. C. Heinz, N. N. Sørensen, and F. Zahle. Fluid–structure interaction computations for geometrically resolved rotor simulations using CFD. *Wind Energy*, April 2016. ISSN 1099-1824. doi: 10.1002/we.1976. URL http://onlinelibrary.wiley.com/doi/10.1002/we.1976/abstract. [Accessed on: 2016-07-15].

[15] S. G. Horcas, F. Debrandere, B. Tartinville, C. Hirsch, and G. Coussement. A new, high fidelity offshore wind turbines aeroelasticity prediction method with significant CPU time reduction. In *Proceedings of EWEA Offshore 2015*. European Wind Energy Association (EWEA), Copenhagen, March 2015.

[16] M. C. Hsu and Y. Bazilevs. Fluid–structure interaction modeling of wind turbines: simulating the full machine. *Computational Mechanics*, 50(6):821–833, August 2012. ISSN 0178-7675, 1432-0924. doi: 10.1007/s00466-012-0772-0. URL http://link.springer.com/article/10.1007/s00466-012-0772-0. [Accessed on: 2016-08-25].

[17] L. Ibing. Postgraduate programme renewable energy: Launch of the High Performance Computing Cluster Eddy, August 2017. URL https://www.uni-oldenburg.de/en/physics/studies/courseofstudies/ppres/ppre/news/newsletter/2017/volume-36/launch-of-the-high-performance-computing-cluster-eddy/. [Accessed on: 2017-8-10].

[18] J. M. Jonkman. Overview of the ElastoDyn Structural-Dynamics Module, November 2013.

[19] J. M. Jonkman. Development, Verification, & Validation of New Aero-elastic Capability Within FAST v8, February 2016.

[20] J. M. Jonkman, G.J. Hayman, B.J. Jonkman, and R.R. Damiani. Aero Dyn v15 User's Guide and Theory Manual. Technical Report, NREL, 2015.

[21] H. Jasak, A. Jemcov, and Z. Tukovic. OpenFOAM: A C++ Library for Complex Physics Simulations. In *Proceedings of International Workshop on Coupled Methods in Numerical Dynamics - CMND 2007*, Dubrovnik, Croatia, September 2007. URL https://www.researchgate.net/profile/Zeljko_Tukovic/publication/228879492_OpenFOAM_A_c_library_for_complex_physics_simulations/links/00463528c618e93598000000.pdf. [Accessed on: 2016-09-02].

[22] J. Jonkman, S. Butterfield, W. Musial, and G. Scott. Definition of a 5-MW Reference Wind Turbine for Offshore System Development. Technical Report NREL/TP-500-38060, NREL, Golden, Colorado, February 2009. URL http://www.nrel.gov/docs/fy09osti/38060.pdf. [Accessed on: 2016-09-05].

[23] J. M. Jonkman. The New Modularization Framework for the FAST Wind Turbine CAE Tool. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, January 2013. URL http://arc.aiaa.org/doi/abs/10.2514/6.2013-202. [Accessed on: 2016-07-21].

[24] Jurado Garcia, X. EDDY - Fakultät V - Carl von Ossietzky Universität Oldenburg, August 2017. URL https://www.uni-oldenburg.de/fk5/wr/hochleistungsrechnen/hpc-facilities/eddy/. [Accessed on: 2017-09-26].

[25] N. Kirrkamm, B. Stoevesandt, B. Gollnick, and J. Peinke. Simulation of a multi mega watt wind turbine with opensouce code openfoam. In *European Wind Energy Conference and Exhibition 2010, EWEC 2010*, volume 6, page 4620. EWEA, Warsaw, April 2010. ISBN 978-1-61782-310-7.

[26] Y. Li. *Coupled computational fluid dynamics/multibody dynamics method with application to wind turbine simulations*. Ph.D. thesis, University of Iowa, Iowa City, Iowa, May 2014. URL http://ir.uiowa.edu/etd/4681.

[27] Y. Li, A. M. Castro, T. Sinokrot, W. Prescott, and P. M. Carrica. Coupled multi-body dynamics and CFD for wind turbine simulation including explicit wind turbulence. *Renewable Energy*, 76: 338–361, April 2015. ISSN 0960-1481. doi: 10.1016/j.renene.2014.11.014. URL http://www.sciencedirect.com/science/article/pii/S0960148114007290. [Accessed on: 2016-07-15].

[28] Y. Liu, Q. Xiao, A. Incecik, and D.-C. Wan. Investigation of the effects of platform motion on the aerodynamics of a floating offshore wind turbine. *Journal of Hydrodynamics*, 28(1):95–101, February 2016. ISSN 1001-6058. doi: 10.1016/S1001-6058(16)60611-X.

[29] M. A. Sprague, J. M. Jonkman, and B. J. Jonkman. FAST Modular Framework for Wind Turbine Simulation: New Algorithms and Numerical Examples. In *33rd Wind Energy Symposium*, NREL/CP-2C00-63203, Kissimmee, December 2015. NREL. URL https://arc-aiaa-org.tudelft.idm.oclc.org/doi/abs/10.2514/6.2015-1461. [Accessed on: 2017-02-19].

[30] M. M. Hand, D. A. Simms, L. J. Fingersh, D. W. Jager,, J. R. Cotrell, S. Schreck, and S. M. Larwood. Unsteady Aerodynamics Experiment Phase VI: Wind Tunnel Test Configurations and Available Data Campaigns. Technical Report NREL/TP-500-29955, NREL, Golden, Colorado, December 2001. URL http://www.nrel.gov/docs/fy02osti/29955.pdf. [Accessed on: 2016-08-01].

[31] J. F. Manwell, J. G. McGowan, and A. L. Rogers. *Wind Energy Explained: Theory, Design and Application*. John Wiley & Sons, Chichester, U.K, 2 edition, 2009. ISBN 978-0-470-01500-1.

[32] J. O. Mo and Y. H. Lee. CFD Investigation on the aerodynamic characteristics of a small-sized wind turbine of NREL PHASE VI operating with a stall-regulated method. *Journal of Mechanical Science and Technology*, 26(1):81–92, January 2012. ISSN 1738-494X, 1976-3824. doi: 10.1007/s12206-011-1014-7. URL http://link.springer.com/article/10.1007/s12206-011-1014-7. [Accessed on: 2017-02-20].

[33] H. Nilsson. Rotating machinery training at ofw10 (openfoam workshop 10) using foam-extend-3.1, June 2015. URL http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2015/HakanNilssonRotatingMachineryTrainingOFW10.pdf. [Accessed on: 2017-08-01].

[34] H. Rahimi, E. Daniele, B. Stoevesandt, and J. Peinke. Development and application of a grid generation tool for aerodynamic simulations of wind turbines. *Wind Engineering*, 40(2):148–172, April 2016. ISSN 0309-524X, 2048-402X. doi: 10.1177/0309524X16636318. URL http://wie.sagepub.com/lookup/doi/10.1177/0309524X16636318. [Accessed on: 2016-08-03].

[35] B. Sanderse, S. P. van der Pijl, and B. Koren. Review of computational fluid dynamics for wind turbine wake aerodynamics. *Wind Energy*, 14(7):799–819, October 2011. ISSN 1099-1824. doi: 10.1002/we.458. URL http://onlinelibrary.wiley.com.tudelft.idm.oclc.org/doi/10.1002/we.458/abstract. [Accessed on: 2016-07-04].

[36] G. Schepers. AVATAR: AdVanced Aerodynamic Tools of lArge Rotors. In *33rd Wind Energy Symposium*. American Institute of Aeronautics and Astronautics, January 2015. ISBN 978-1-62410-344-5. doi: 10.2514/6.2015-0497. URL http://arc.aiaa.org/doi/10.2514/6.2015-0497. [Accessed on: 2017-08-21].

[37] J. G. Schepers. *Engineering models in wind energy aerodynamics: Development, implementation and analysis using dedicated aerodynamic measurements*. Ph. D. thesis, TU Delft, 2012. URL http://dx.doi.org/10.4233/uuid:92123c07-cc12-4945-973f-103bd744ec87. [Accessed on: 2016-07-20].

[38] M. Schramm, B. Stoevesandt, and J. Peinke. Adjoint optimization of 2d-airfoils in incompressible flows. In *Proceedings of the 11th World Congress on Computational Mechanics*, pages 6200–6211, 2014. ISBN 978-84-942844-7-2.

[39] Y. Song and J. Perot. CFD Simulation of the NREL Phase VI Rotor. *Wind Engineering*, 39(3): 299–310, June 2015. ISSN 0309-524X. doi: 10.1260/0309-524X.39.3.299. URL http://wie.sagepub.com/lookup/doi/10.1260/0309-524X.39.3.299. [Accessed on: 2016-09-02].

[40] T. Stovall, G. Pawlas, and P. Moriarty. Wind Farm Wake Simulations in OpenFOAM. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, Orlando, Florida, January 2010. URL http://arc.aiaa.org/doi/abs/10.2514/6.2010-825. [Accessed on: 2016-09-02].

[41] The OpenFOAM Foundation. OpenFOAM User Guide Ver 4.0, June 2016. URL http://foam.sourceforge.net/docs/Guides-a4/OpenFOAMUserGuide-A4.pdf.

[42] Q. Wang, N. Johnson, M. A. Sprague, and J. M. Jonkman. BeamDyn: A High-Fidelity Wind Turbine Blade Solver in the FAST Modular Framework. In *33rd Wind Energy Symposium*. American Institute of Aeronautics and Astronautics, January 2015. ISBN 978-1-62410-344-5. doi: 10.2514/6.2015-1465. URL http://arc.aiaa.org/doi/10.2514/6.2015-1465. [Accessed on: 2017-08-21].

[43] K. Wolf. MPCCI – The General Code Coupling Interface. In *Proceedings of German LS-DYNA Forum 2007*, Frankenthal, October 2007. DYNAmore GmbH. URL https://www.dynamore.de/de/download/papers/forum07/it-cae-processes/mpcci-the-general-code-coupling-interface/view. [Accessed on: 2016-01-09].

[44] D. O. Yu and O. J. Kwon. Time-accurate aeroelastic simulations of a wind turbine in yaw and shear using a coupled CFD-CSD method. *Journal of Physics: Conference Series*, 524(1):012046, 2014. ISSN 1742-6596. doi: 10.1088/1742-6596/524/1/012046. URL http://stacks.iop.org/1742-6596/524/i=1/a=012046. [Accessed on: 2016-07-15].

[45] M. Zamani, M. Maghrebi, and S. Varedi. Starting torque improvement using J-shaped straight-bladed Darrieus vertical axis wind turbine by means of numerical simulation. *Renewable Energy*, 95:109–126, September 2016. ISSN 0960-1481. doi: 10.1016/j.renene.2016.03.069. URL http://www.sciencedirect.com/science/article/pii/S0960148116302531. [Accessed on: 2016-09-05].