Computational Modelling of the Oryon Watermill

A Nested Fluid-Structure Interaction Problem with Non-Smooth Dynamics

Vineeth S. Maniyara

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

Computational Modelling of the Oryon Watermill

A Nested Fluid-Structure Interaction Problem with Non-Smooth Dynamics

by

Vineeth S. Maniyara

in partial fulfillment of the requirements for the degree of

Master of Science

in Applied Mathematics

at the Delft University of Technology,

to be defended publicly on Wednesday September 27, 2017 at 10:00 AM.

Student number:	4616944	
Project duration:	November 1, 2016 – September 27, 2017	
Thesis committee:	Prof. dr. ir. C. Vuik,	TU Delft, Professor - Numerical Analysis
	Dr. ir. D. R. van der Heul,	TU Delft, Academic Supervisor
	Dr. J. L. A. Dubbeldam,	TU Delft, Asst. Professor – Mathematical Physics
	Ir. H. Barneveld,	HKV, Industry Supervisor

This thesis is confidential and cannot be made public

Executive Summary

This study sets out to develop a solver to simulate the functional behaviour of the Oryon Watermill, the accuracy of which is established by a comparison with the turbine test results extracted by MARIN. The solver is to evaluate the turbine at its cross-sectional slice and then extrapolate the two dimensional results to a three dimensional turbine torque approximation.

The solver is developed within OpenFOAM's C++ architecture and boils down to writing a new structural solver class that works in tandem with the pre-existing dynamic domain fluid solver. The simulation ideology is based on determining the torque generated by the turbine for different turbine rotational speeds and building the turbine performance curve from results of a number of such simulations.

The turbine performance curve generated by the developed solver is found to be an overprediction as compared to the turbine test results from MARIN. Although overpredicted, the curve is found to show a good capture of trend and a good approximation of the turbine's optimal rotational speed range. The overprediction is hypothesised to be a byproduct of not capturing the observed flow leakage within the turbine casing. This characteristic of the turbine, exists in the unaccounted for direction perpendicular to the considered two dimensional turbine cross section.

The Oryon Watermill, in all its glory, is presented in **Fig 1**

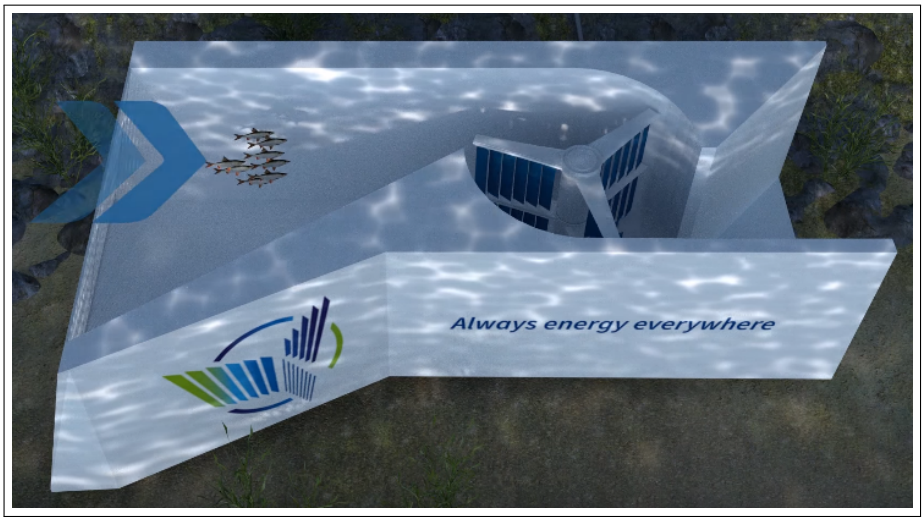


Figure 1: The Oryon Watermill by Deep Water Energy

Abstract

The current research aims to numerically predict the performance characteristics of the Oryon Watermill. The simulation mechanism to be developed, begins at the structural solver part of a partitioned FSI solver, upon which the unique functional characteristics of the turbine are embedded. The modelling is carried out to account for the turbine's functional characteristics like, multiple nested rigid body FSI dynamics, non-smooth dynamics and lamella partial torque contributions. The solver is built to resolve the turbine dynamics in two dimensions and re-scale the final result with respect to the turbine core height.

The results of the simulation show the predicted torque to be of similar trend as that of the experimental data collected by MARIN, while being an over-prediction by a factor of two. The optimal performance range of the turbine is predicted within an error bound of 0.1RPS. The overprediction of the torque is considered to be a result of the 2D solution not accounting for the flow leakage in the third and unaccounted for direction. The time variant fluid solution is found to be unstable due to the spikes observed in the torque time signal and is a consequence of the perfect momentum sink assumption employed in the modelling of the lamella's non-smooth dynamics. The lamella dynamics are observed to not convergence to fully periodic behaviour and the sensitivity of the lamella's dynamics is considered to be the root cause. The velocity Verlet based structural solver does not provide fluid-structure coupling iterations, the inclusion of which, could improve the stability of the fluid solution.

This endeavour establishes a foundation for the predictive numerical simulation of the OWM in the form of baseline numerical simulation results and a developed simulation framework.

To a higher power,

with whose grace,

life is possible.

Acknowledgements

"No duty is more urgent than that of returning thanks".

James Allen

I extend my gratitude to the COSSE consortium, in particular, Prof.Dr.R. Nabben, Prof.Dr.Ir.C. Vuik, Dr.M. Hanke and K. Knutsson for the opportunity and support to partake in a learning experience of a lifetime. To Dr.Ir.D.R. van der Heul, my academic supervisor, I owe a debt of gratitude for the patience, guidance and insight he provided during the arduous and fulfilling journey of conducting my thesis research.

I would like to thank Ir.H. Barneveld, my industry supervisor, and Ir.G. Horn for their support and cooperation while I conducted my thesis research with HKV Consultants. My appreciation extends to J. Ory and D. Passman, from Deep Water Energy, for providing me with the Oryon Watermill to work with and for their insight into its functionality. Many thanks are due to M. Broers, from BT Projects, for sponsoring the research work and R. Caljouw, from BT Projects, for taking the time to provide feedback with regard to the thesis results.

It goes without saying, that I extend my heartfelt gratitude to my family, for both, their emotional and financial support, in my venture to take a hiatus from the industry and pursue higher education. I thank the Numerical Analysis group at TU Delft, for giving me a workspace to conduct my research and bask in the afterglow of numerical mathematics. I extend a great deal of appreciation to my thesis room mate Hrishikesh, for the technical discussions we shared and for being a fellow passenger on my thesis journey. Last, but definitely not the least, I thank the master and PhD. students of the numerical analysis group, Dmitrii, Hassan, Merel, Deborah, Manuel, Behrouz, Jochen, Gabriella and Roel, for the good times we shared over the course of my stay at the group.

Contents

List of Figures

1

Introduction

"The purpose of computing is insight, not numbers"

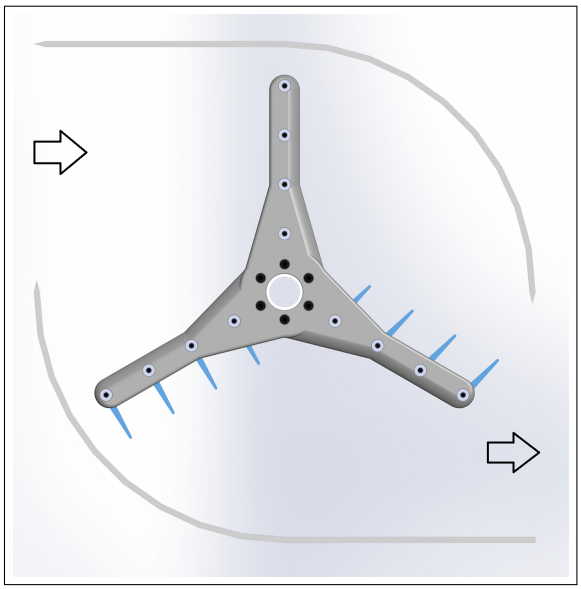
Richard Hamming

This study aims to numerically model a turbine in order to estimate its power generation capabilities and resolve its flow and structural dynamics. The validation of this simulation methodology is performed using experimentally determined turbine performance data. The developed simulation mechanism is designed for a two dimensional model of the turbine. This choice of dimensional reduction originates from the need to start the numerical modelling from the most fundamental level and build upwards. The principle purpose of the solver is to generate the torque vs rotational speed curve of the turbine, for a particular inlet flow velocity. To this effect, the simulation ideology defined involves setting up separate simulations to determine the generated torque for different turbine rotational speeds. The number of simulations needed to generate the required turbine torque curve is dependent on the number of points of resolution desired on the QvsN curve. A resolution of seven points is chosen at the optimal performance neighbourhood of the turbine.

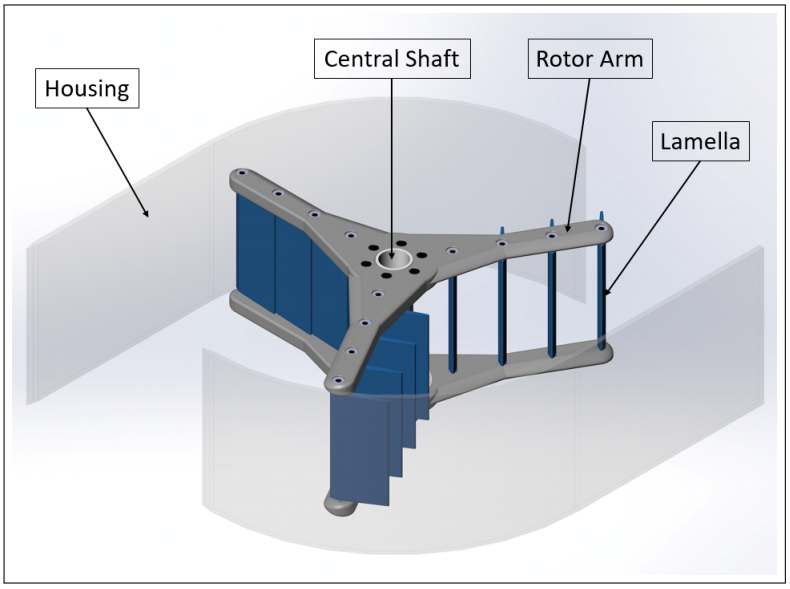
The innovative water turbine concept referred to as the Oryon Watermill (OWM), being developed by Deepwater Energy BV (DWE), is a vertical axis water turbine capable of harvesting green energy from flowing water. The device is a modular build hydro power plant, which can be economically adapted to the local water flow profile. The adaptable characteristic of the turbine lends credence to the need for numerical modelling approaches which would allow for efficient design customisation. It is worthwhile to note that the OWM has been especially designed to generate power under low pressure head conditions, which is generally the case with inland water bodies such as rivers.

The turbine primarily consists of a vertical axis three arm rotor, with four equivalently sized movable flaps/lamellas per rotor arm and encased within the turbine housing/casing. The rotor arms are connected to a drive shaft, which is in turn linked to

an electric generator. The motion of the rotor arms as well as their corresponding flaps are influenced solely by the flowing water, wherein the flaps are considered closed at a 0° angle and fully open at a 90° angle, with respect to their corresponding rotor arm. Displayed in **Fig** 1.1a and **Fig** 1.1b, are the isometric and top views, respectively, of the OWM design considered for the purpose of this study. The dimensional details of its component parts are depicted in **Fig** 4.3.

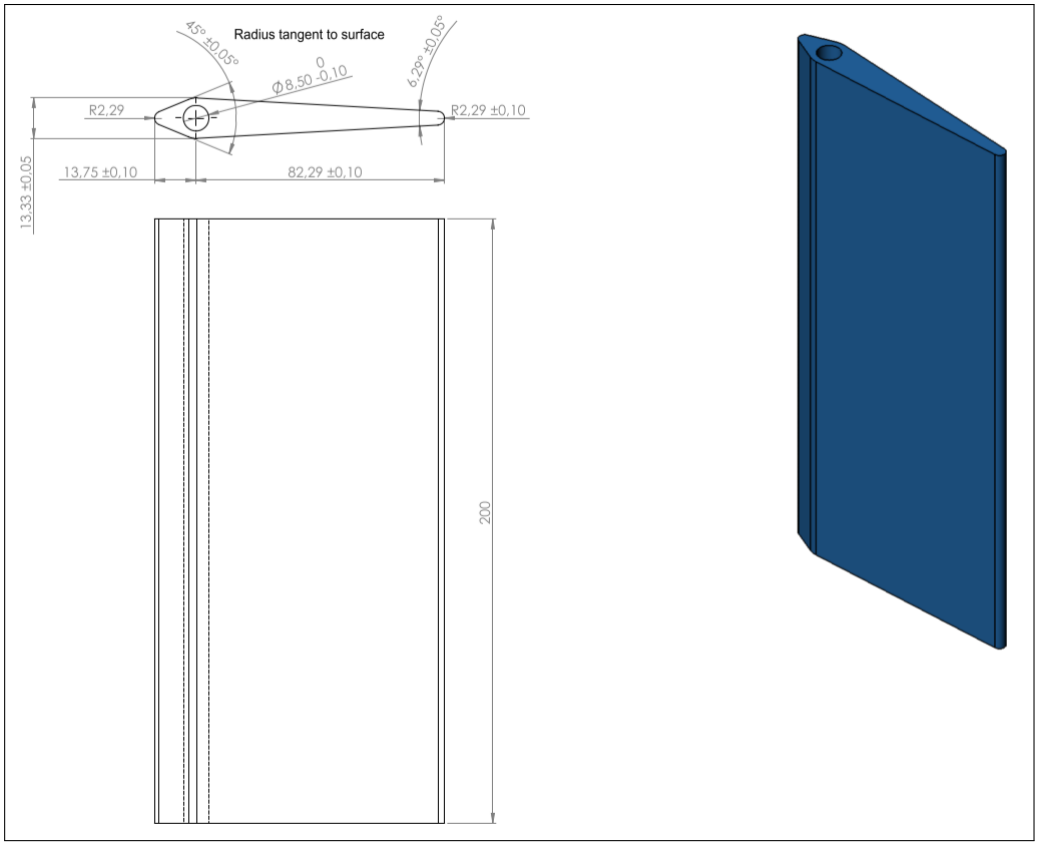


(a) The Oryon Watermill - Top View

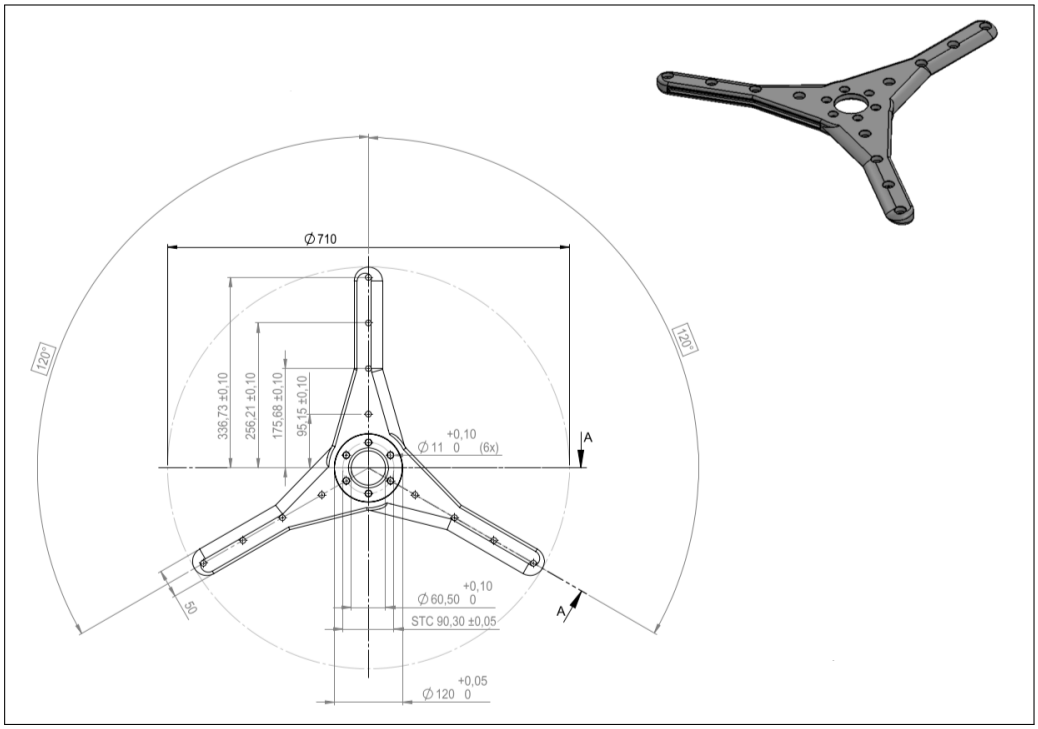


(b) The Oryon Watermill - Isometric View

Figure 1.1: Oryon Watermill Design



(a) Test Model Lamella Dimensions



(b) Test Model Wing Disk Dimensions

Figure 1.2: Oryon Watermill Test Model Component Dimensions

The OWM works as a drag based turbine, wherein the water pushes the rotor arm with closed lamellas, downstream with the flow, thereby inducing torque on the rotor. During the upstream part of the rotor arm rotation, the lamellas align themselves with the flow, thereby minimising the arm area resisting rotor motion and as a result, minimising drag. The total torque and power delivered by the water is the total of the contributions of the upstream and downstream moving rotor arms. During operation, the rotor periodically completes full rotations of 360° while the flaps on each arm periodically open and close. The turbine is encased within a housing, the purpose of which is to directionalise the incoming flow to suitably impinge upon the rotor arm and exit the housing without adversely impacting the overall flow dynamics, morphology or wildlife of the water body.

1.1. Reserach Questions

The initial research questions put forth to be answered in this research were as follows:

1. Does a three dimensional simulation model generate an approximation of the turbine net torque within an error bound of 10%, as validated by the prototype test results from MARIN?

2. Does the use of the porous screen model to replace the dynamic lamellas result in a physically appropriate solution? If so, what is its torque approximation error in relation to that of the three dimensional turbine simulation?

3. Does the FSI integrated non-smooth dynamical system model approximate the non-smooth behaviour of the lamella to be in-line with its associated physical phenomena?

4. Does the limiting of the three dimensional simulation solution to two dimensions and then extrapolating those results to relate to the actual height of the turbine, result in a physically appropriate solution? If so, what is its torque approximation error in relation to that of the three dimensional simulation?

Over the course of the study, these research questions were modified as follows:

- The investigation into a three dimensional simulation model was relegated to future work and the project was focused at developing a two dimensional simulation model and evaluating its predictive capabilities. This re-shifting of focus allows a stage wise progression in the simulation model development.

- The porous media approach to model the dynamic lamellas was deemed to not have sufficient academic and engineering value as it redirected focus away from salient modelling concepts such as non-smooth dynamics, dynamic mesh handling and fluid-structure interaction. The concept was found to be too far removed from the heart of the research.

- The investigation into the non-smooth dynamics modelling was extended into an investigation into the suitability of a number of important modelling approaches that were critical to developing a simulation model of the OWM.

The final set of updated research questions were formulated as follows:

1. What is the extent to which the two dimensional simulation model accurately predicts the performance characteristics of the turbine and is an extension of the model to three dimensions necessary? The two dimensional numerical simulation is expected to provide an idealistic performance evaluation of the turbine. This expectation stems from the flow leakage not accounted for in the two dimensional model. The flow leakage occurs through gaps within the casing, at the top and bottom of the turbine.

2. How suitable are the chosen modelling approaches in capturing the unique operational behaviour of the turbine? These unique functional behaviour can be summarised as follows:

    - The local rotational motion of the lamellas is determined by the fluid solution, which requires FSI modelling - **Section 3**.
    - The lamellas of the turbine undergo a rotation about its own axis and a revolution about the vertical shaft of the turbine, which requires the combined modelling of its rotation and translation - **Section 4.2**.
    - In the computational model of the turbine, the lamellas are pivoted at their center due to limitations arising from the use of dynamic mesh handling - **Section 4.1.1**. Virtual hinging from the lamella leading edge is adopted to match angular behaviour of the physical model and the computational model - **Section 4.3**.
    - The lamella dynamics involve non-smooth rotational motion, which involves the modelling of its non-smooth behaviour using an event based approach. Three events are defined, namely, fully open, fully closed and non-constrained motion - **Section 4.4**.
    - All lamellas contribute to the total torque generated by the turbine and require summation of their contributions to determine the total turbine torque **Section 4.5**.

1.2. Future Implications of Research

The furture purpose of developing a simulation model for the OWM orbits about the following three major agendas:

- The OWM being a customisable turbine, is meant to have its design particulars optimised with respect to the nature of the water body it is to be deployed in. The availability of a well developed simulation means to realise the custom design optimisation of the turbine forms an integral part of the product's deployment, use and viability.

- A sufficiently robust simulation model of the OWM could be utilised to perform numerical studies on a turbine that inherits the drag based vertical axis characteristics of the OWM. In other words, the same simulation mechanism can be effected to different configurations of the OWM.

- The simulation model of the OWM could be used to seed a black-box model of the turbine, which would subsequently be used to investigate the impact of the turbine upon the flow and morphology of the water body it is deployed in.

2

Model Tests

The Maritime Research Institute, Netherlands (MARIN) was commissioned by DWE to conduct experimental model tests on the OWM under the scope of the "Dutch MKB concept test campaign". Consequently, Report No: 28537-1-BT,titled, "Oryon Water Mill Power Generation Tests In Calm Water", was generated by MARIN and stands as the primary validation reference used in this study. The primary goal of the conducted tests were to determine the performance characteristics of the OWM and hence, pivots on dimensionless power coefficient curves calculated for a number of inlet flow velocities and turbine casing models.

The document on the model tests not only provides experimental validation data for the turbine, but also lends critical insight into the turbine behaviour. Understanding the turbine functionality plays an important role in identifying and determining the modelling approaches to be utilised in the numerical model. The two dimensional simulation model results are scaled with respect to the height of the lamella (**Fig 1.2a**), parallel to the axis of rotation of the OWM. This scaling procedure extends the two dimensional results to the third dimension and makes it comparable to the experimental results from MARIN. In the forthcoming sub-sections, a brief description of the model test setup, characteristic of flow through the device, basics of power extraction, test procedure and the necessary final results of the model tests are presented.

2.1. Model Test Setup

The tests were conducted in a controlled laboratory environment in the "Shallow Water Basin" (Binnenvaart or BT) of MARIN. The tests were performed as towing tests, wherein the OWM is towed through the BT at a velocity equivalent to that necessary as an inlet flow to the device. The test setup consisted of a six component measurement frame with an attached mounting frame for an electric generator and an attached drive shaft. It also involved the shaft of the OWM containing a torque sensor which measures forces in the x, y and z directions, with the positive direction of turbine rotation defined as clockwise when viewed along the shaft from above the water surface. Additionally, the setup included flow meters installed at several locations at the inlet and outlet of the turbine housing. This setup was then mounted under the towing carriage of the BT facility.

The model tests were conducted for several variants of turbine housing and for a number of different towing speeds. This report summary will include a review of one of the five variants tested by MARIN, with the highest evaluated power generation coefficient at a single selected towing velocity of 2.5m/s (as requested by DWE). Type IIA is chosen as the considered variant of the OWM and consists of only the curved housing segments and corresponds to the turbine system displayed in **Fig 2.1** as a sketch. This constitutes the specific turbine test setup of the OWM, previously referred to at the beginning of this chapter.

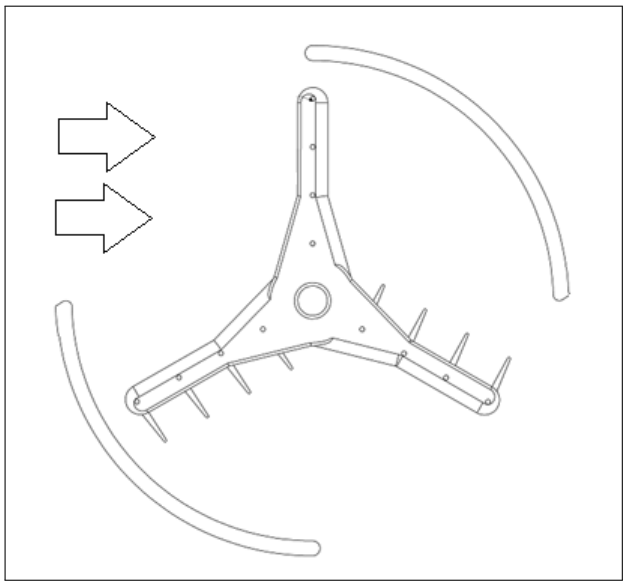


Figure 2.1: The Oryon Watermill Variant IIA - Simulation Model

2.2. Turbine Flow Characteristics

The Reynold's number for a considered flow case is determined by the following equation.

$$Re = \frac{\rho u L}{\mu}, \tag{2.1}$$

where, ρ is the density of the fluid, u is free stream velocity near the considered surface, L is the characteristic length of the considered body and μ is the dynamic viscosity of the fluid. For the case of the OWM, we chose the lamella to be the body based on which we estimate the Reynold's number of the flow is calculated and the fluid to be water. The free stream velocity inside the turbine casing is assumed to be half of the inlet velocity and the corresponding calculation is enlisted below.

$$\begin{gathered} \rho = 1000 kg/m^3 \qquad \mu = 1 \times 10^{-3} Pas \qquad v = 1 m/s \qquad L = 0.1 m \\ Re \approx 1 \times 10^5. \end{gathered} \tag{2.2}$$

With this estimation being greater than 2000, we see the flow to be well within the turbulent regime.

2.3. Power Extraction and Test Procedure

When the turbine is operated without the electric generator applying a counter torque, it is said to be rotating at a free running rate and for the alternate case, it is said to be producing electrical power. The amount of power generated by the turbine, P, depends on the rotational rate of the rotor seen here as rotations per second (RPS), N and the net torque generated by the turbine , Q. This is formulated as,

$$P = 2\pi QN. \tag{2.3}$$

Starting from the free running rotational rate, the torque and extracted power increase with decreasing rotational rate. At a certain rotational rate, an optimum working point is attained with maximum extracted power. When the rotation rate is further reduced, the effective output power begins to decrease.

MARIN used an in-house developed measurement procedure called Quasi Steady Measurement (QSM) for the model tests, during which the turbine is towed at a constant speed through the water while the rotation rate of the rotor is varied linearly, starting from a state of rest, to a maximum RPM slightly lower than the free running rate and then back to rest (referred to as a "sweep"). The variation of the rotational rate while having a constant inlet flow is achieved by varying the counter torque applied to the turbine shaft. Inverse to the rotational variation, the counter torque is started at a rotation limiting value, reduced to zero and then raised back up to the limiting value. During each run, the measurement data, namely torque and forces, are sampled by the acquisition sensors as a function of the rotation rate of the rotor. This relates to the fact that, with just one sweep of the rotation rate, the QSM procedure captures the required data needed to assess the performance and characteristics of the turbine.

2.4. Model Test Results

MARIN presents the power and forces measured during the model tests in non-dimensional form. The non-dimensional power coefficient considered is C_p , which represents the ratio between energy extracted and total energy available in a flow tested in unrestricted water. **Fig 2.2** is included to consolidate the concept behind the flow case.

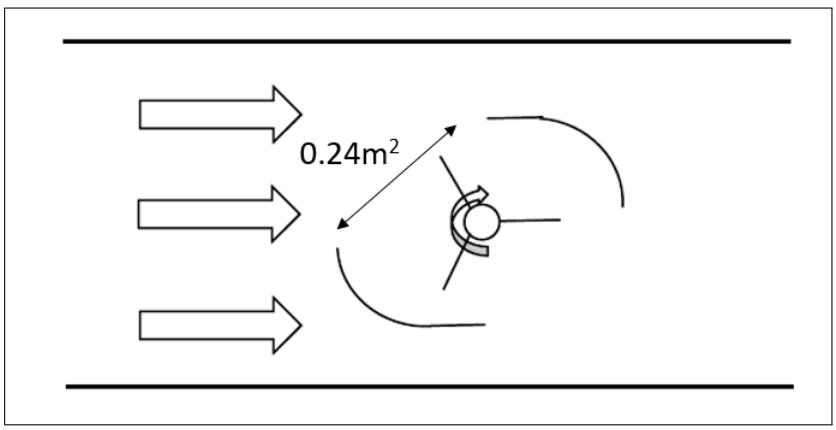


Figure 2.2: Unrestricted Water Turbine Test

The available power in unrestricted water is based on the kinetic energy in the rotor projected area (A) and is calculated by the equation below,

$$P_{available} = \frac{1}{2}\rho A v^3. \tag{2.4}$$

Thus, the power coefficient in unrestricted water formulates to the result below,

$$C_p = \frac{P}{P_{available}} = \frac{2\pi QN}{\frac{1}{2}\rho A v^3}. \tag{2.5}$$

The maximum value of C_p for the considered case as estimated from the model test is tabulated below, Displayed in **Fig 2.3** is the power coefficient estimated by the model

Table 2.1: MARIN Model Test Summary for Simulation Case

OWM Variant	**Test Speed**	**Max. C_p**	**Reynold's Number**	**A**
IIA	2.5m/s	0.0553	1×10^5	$0.24m^2$

tests, wherein C_p is represented as a function of the tip speed ratio (TSR). The TSR is a non-dimensional number defined as the ratio between the velocity of the rotor tip and the freestream velocity of inlet flow. The formulation for TSR is given by,

$$TSR = \frac{v_{tip}}{v} = \frac{2\pi RN}{v}, \tag{2.6}$$

where v_{tip} is the velocity of the rotor tip and R is the radius of the turbine.

For each quantity, three curves are plotted, where the red curve represents the variations of the considered value during the QSM runs by way of a polynomial fit and the grey curve depicts the average value of the two sections of the test run (for increasing and decreasing RPM). The differences between the two branches of the curve illustrate the variation of the measurement signal during the test runs. Included below, is a fifth order polynomial representing the red curve.

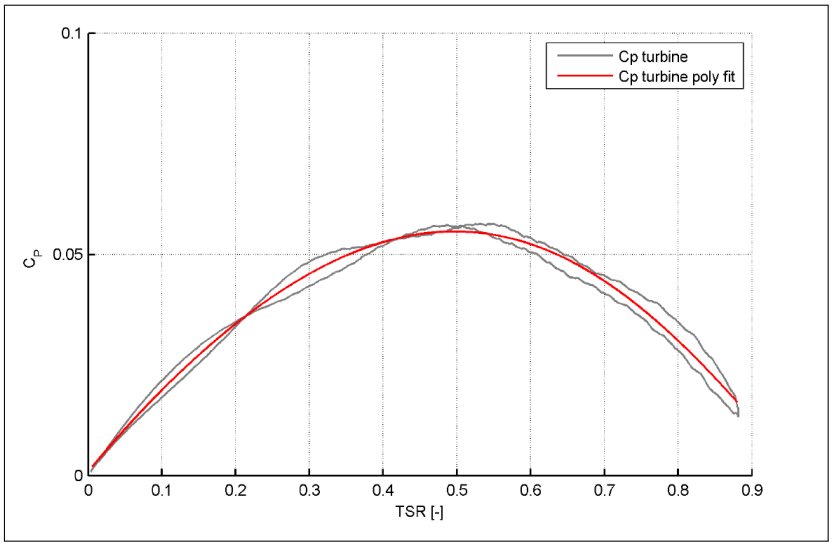


Figure 2.3: Unrestricted Flow MARIN Model Test Results : C_p Vs TSR - OWM IIA at 2.5m/s [1]

$$c_p(TSR) = 0.180689 \times (TSR)^5 - 0.323335 \times (TSR)^4 + 0.125402 \times (TSR)^3 - 0.193918 \times (TSR)^2 + 0.203442 \times (TSR) + 0.000895288. \quad (2.7)$$

When $TSR = 0$, it follows that $N = 0$, from **Eq 2.6**. When $N = 0$, it can be seen from **Eq 2.5**, that $C_p = 0$. Which in other words, relates to the fact that a stationary turbine generates no power and results in $C_p(TSR = 0) = 0$.

2.5. Derived Physical Quantities

In order to alternatively visualise the power coefficient curve presented in **Fig 2.3**, a torque (Q) versus rotations per second (N) plot is derived from the available data. The rotations per second (N), can be calculated from the TSR values according to the equation below,

$$N = \frac{TSR\ v}{2\pi R}. \quad (2.8)$$

Utilising the C_p vs TSR data from **Eq 2.7**, the torque is derived using the equation below,

$$Q = \frac{C_p \rho A v^3}{4\pi N}. \quad (2.9)$$

From the formulations in **Eq 2.8** and **Eq 2.9**, the resultant torque vs rotational speed curve, for a constant inlet flow velocity of $2.5m/s$, is depicted in **Fig 2.4**. From this curve, torque is seen to rapidly grow in the vicinity of $N = 0$. The torque is instead, expected to remain some finite value that resists the torque induced by the incoming flow to maintain the turbine's stationary behaviour. Thus, the formulation is invalid in the vicinity of $N = 0$ and the curve is to be truncated accordingly.

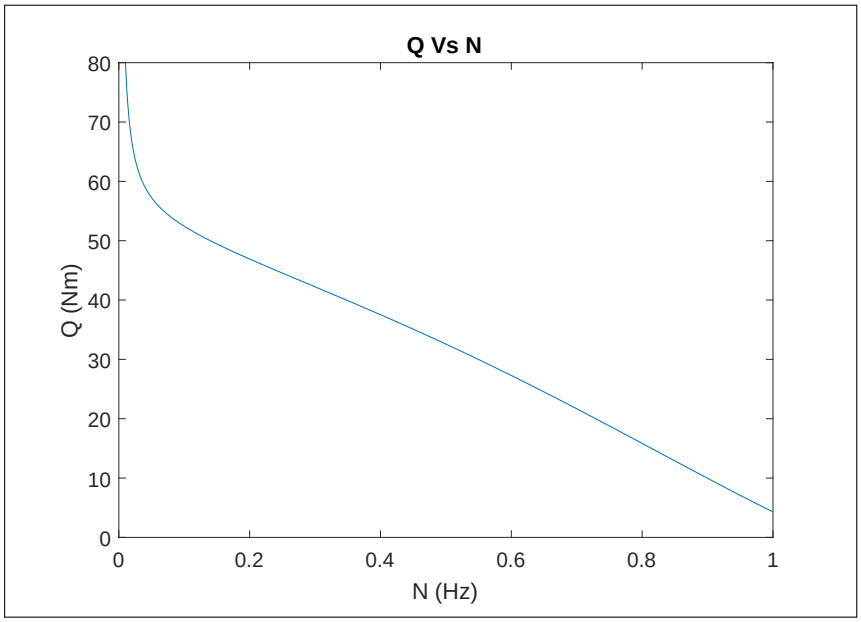


Figure 2.4: Torque Curve Derived from Unrestricted Flow MARIN Model Test Results : Q Vs N - OWM IIA at 2.5m/s

The truncated curve is displayed in **Fig 2.5**

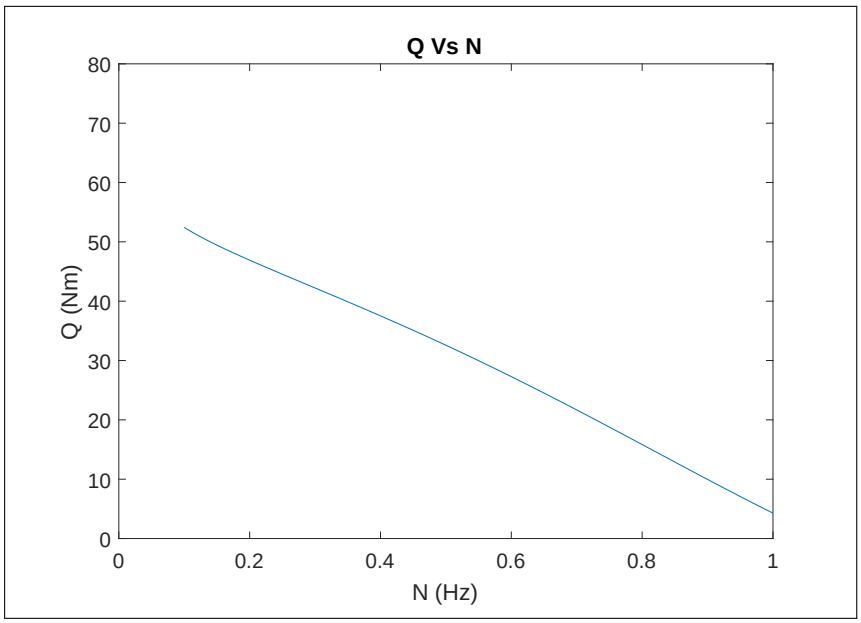


Figure 2.5: Truncated Torque Curve Derived from Unrestricted Flow MARIN Model Test Results : Q Vs N - OWM IIA at 2.5m/s

If the torque curve is extrapolated towards a higher RPS value, the torque would turn negative. In such a case, the turbine would consume power rather than generate it. Hence, only the range of rotational speeds less than one and greater than zero are pertinent to the study.

3

Fluid Structure Interaction

The OWM being a power generating turbine, is run by the convective fluid impinging upon the structural face of the turbine rotor and thereby inducing its motion. The fluid and structure induce a change in momentum upon one another, which is referred to as an interaction between the fluid and the structure and thus establishes the need to incorporate Fluid-Structure Interaction (FSI) into its simulation mechanism. In this particular problem, the structure is assumed to undergo no deformation and purely displace as a solid body. The FSI modelling is based on the pre-existing single body, partitioned, FSI solver included in OpenFOAM 2.4.0. This stands as the start template upon which the rest of the model developments are built. The resulting FSI solver is realised as a solid body rotational motion solver for the twelve lamellas of the OWM.

FSI involves the handling of the coupling between fluid flow and a flexible mechanical structure while taking into consideration the exchange of momentum and energy occurring during the coupling process. As FSI is the interaction of some movable and/or deformable structures with an internal or surrounding fluid flow, it can be relatively stationary and/or oscillatory in nature. In oscillatory interactions, the strain induced in the solid structure causes it to move and/or deform, such that the source of strain is reduced and the structure returns to its former state, only for the process to repeat. Relatively stationary interactions involve rigid body motions wherein the entire structure undergoes motion as a whole and its component parts remain stationary with respect to itself.

FSI problems are multiphysics (multiple simultaneous physical phenomena) problems for which there exists two major simulation approaches, namely, the monolithic approach and the partitioned approach, wherein the equations governing the flow and the mechanics of the structure are solved simultaneously with a single solver and solved separately with two distinct solvers, respectively. The monolithic approach requires a custom built solution mechanism for the particular combination of physical problems at hand and tends to be more accurate due to the uniformity of mathematical and numerical methodologies at play. The partitioned approach enables algorithmic modularity as a flow and structural solver are to be coupled and facilitates solution of the flow equations and the structural equations with different, possibly more efficient techniques, thereby

making them independent of one another. The one draw back of the partitioned approach is the need for the development of a sufficiently accurate coupling algorithm. For the modelling investigation at hand, a partitioned approach is preferred, as it provides a flexible framework to combine pre-existing fluid and structural solvers to resolve the modelling problem.

3.1. FSI - The Partitioned Approach

In order to build a general understanding of the partitioned/non-monolithic approach to FSI, we consider Newton's second law of motion applied to a mechanical structure in the linear analogue.

$$\frac{d\mathbf{m}}{dt} = \sum \mathbf{F_i}, \tag{3.1}$$

with $\mathbf{m}$ being the momentum of the structure and $\mathbf{F_i}$ the applied forces like gravity and viscous force. This approach involves computing the left hand side term with a structural solver, the right hand side term with a fluid flow solver and resolving the equality with a coupling scheme to update relevant data between the solvers. In a generalistic sense, the left hand side of this simple equation depicts the dynamics of the mechanical structure, including its deformation, wherein the structure is composed of a number of particles with a mass, having non-uniform accelerations due to varying interactions with internal material and fluid forces. The right hand side depicts the net force acting on the structure (derived out of the fluid pressure at the structure surface) after the flow solution has been converged up to a suitable bound in continuity error.

For an FSI calculation, the data exchanged correspond to the partial pressure, supplied by the fluid flow solver and the position and velocity field of the mechanical system, supplied by the structural solver. In **Fig 3.1**, a pictorial representation of the fundamental concept behind a partitioned approach to FSI is presented.

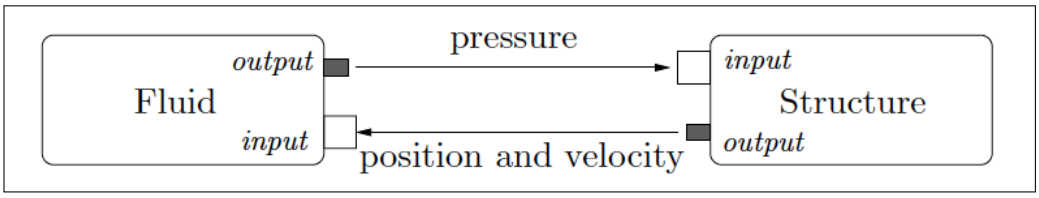


Figure 3.1: Principle of a Partitioned FSI Approach [2]

3.2. The Fluid Model

The foundation of the fluid model is laid by establishing its modelling particulars. This involves defining a fundamental principle of fluid behaviour, a modelling paradigm and a frame of reference, as enumerated below,

- The fluid involved in the dynamics of the problem is water. Water being a Newtonian fluid, the fundamental aspect of Newtonian fluid behaviour is considered. For a Newtonian fluid, deformation under shear force is such that its deformation

velocity is a functional of the shear forces acting on it. i.e,

$$\tau \propto \frac{\partial \xi}{\partial t}, \tag{3.2}$$

where τ represents the shear forces and ξ, the deformation.

- The fluid is modelled as a continuum, wherein each fluid element represents a very large number of molecules whose macroscopic properties are obtained by averaging over an ensemble of fluid molecules. This modelling paradigm describes the evolution of continuum field properties, such as pressure, density, temperature and velocity.

- Utilising the Cartesian co-ordinate system in three dimensional space, the Eulerian approach is adopted as a frame of reference, within which, the observer remains at a fixed position and as a result, the field is described as a function of space and time. Thus, we aim to generate the governing equations in conservation form.

As an additional note, since the problem at hand does not involve thermal effects, the energy equation of fluid mechanics is truncated from the considered governing equations.

3.2.1. Navier-Stokes Equations for Incompressible Flows

The differential conservation form of the scalar momentum equations of fluid flow, better known as the Navier-Stokes equations, are displayed below along with the incompressibility constraint.

$$\frac{\partial}{\partial t}\left(\rho \cdot \mathbf{V}\right) + \nabla \cdot \left(\rho \mathbf{V} \otimes \mathbf{V}\right) = \nabla \cdot \underline{\underline{\mathbf{T}}} + \rho \cdot \mathbf{f}, \tag{3.3a}$$

$$\nabla \cdot \mathbf{V} = 0, \tag{3.3b}$$

where, $\mathbf{V}$ is the velocity of the fluid passing through an infinitesimally small fluid element, ρ is the density of the fluid element, $\underline{\underline{\mathbf{T}}}$ is the Cauchy stress tensor that describes the surface shear forces effecting the fluid element and $\mathbf{f}$ being the body force acting on the fluid element. Further details on the derivation of the fluid equations can be referred to in [3].

A transformation from vector notation to index notation is performed to remain in line with general practice and for the ease of use. The transformation of the vector notation in equation set **3.3** into index notation, is performed based on the following three rules,

- V_i denotes the i^{th} Cartesian component of $\mathbf{V}$, and x_i denotes the i^{th} component of $\mathbf{x}$.

- Differentiation is denoted by a comma, such that,

$$V_{i,j} = \frac{\partial V_i}{\partial x_j}. \tag{3.4}$$

- Repeated indices imply summation, such that, for 3-dimensional Cartesian space we have,

$$V_{i,jj} = V_{i,11} + V_{i,22} + V_{i,33} = \frac{\partial^2 V_i}{\partial x_1^2} + \frac{\partial^2 V_i}{\partial x_2^2} + \frac{\partial^2 V_i}{\partial x_3^2}. \tag{3.5}$$

Making this transformation, we have,

$$\boxed{\begin{aligned} (\rho V_i)_{,t} + (\rho V_i V_j)_{,j} &= T_{ij,j} + \rho f_i, \qquad (3.6a)\\ V_{i,i} &= 0, \qquad (3.6b) \end{aligned}}$$

where, δ_{ij} being the kronecker delta, we define,

$$T_{ij} = -p\delta_{ij} + 2\mu\epsilon_{ij}, \tag{3.7a}$$

$$\epsilon_{ij} = \frac{1}{2}(V_{i,j} + V_{j,i}). \tag{3.7b}$$

Here, p is the pressure inside the fluid element, μ is the dynamic viscosity of the fluid and ϵ_{ij} is the stress-strain tensor associated to the fluid element.

3.2.2. Fluid Flow Formulation on Moving Domains

The considered problem statement includes the fluid flow impinging on the rotor of the turbine, thereby leading to its rotational motion, which in turn translates into a case of flow on moving domains involving fluid-structure interaction. This requires the need to recast the governing equations of fluid mechanics into a framework that accounts for the motion of the fluid in a moving domain.

There exists two classical formulations, namely Eulerian and Lagrangian, based on the relations between the observer being fixed in space and being attached to the fluid particle, respectively. The equations of sets **3.3** and **3.6** correspond to the Eulerian approach as it exists in the conservation form. The Lagrangian approach is one generally used in closed domains because of its limitation that any particle leaving the domain must be replaced by a new one [2]. To address the matter at hand, involving moving domains, a third formulation exists, based on the combination of both the classical approaches, known as the Arbitrary Lagrangian Eulerian (ALE) approach.

The fundamental concept behind the ALE approach is illustrated in **Fig 3.2**, using an electrical analogy of the measurement of the flow of electrons across a stationary / moving *probe or frame of reference*, in an electric cable [2].

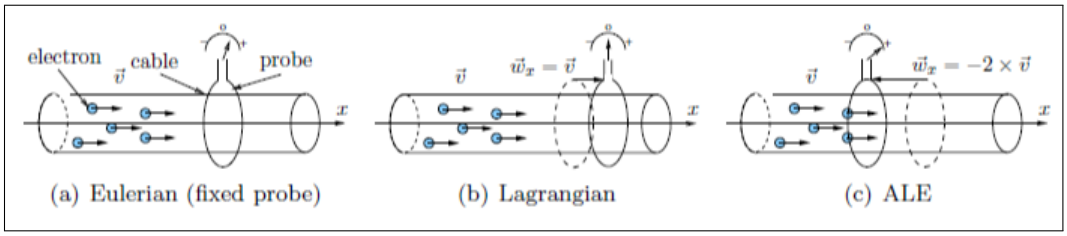


Figure 3.2: Electrical Analogy for Flow Measurement Relative to Probe Motion [2]

All three approaches are illustrated in the analogy wherein the probe represents the reference domain, the flow of electrons represents the transport of mass, momentum or energy of the fluid and the ammeter reading depicts the flux $\vec{F}$ measured with respect to the reference domain, wherein the flux term is defined as a quantity q (mass, momentum or energy), flowing through a section S, per unit time. Consequently, the Eulerian approach involves a stationary probe, the Lagrangian approach includes a probe moving with the electrons and the ALE approach is depicted with the probe moving in the direction opposite (considered arbitrarily for the one dimensional analogy) to the flow of electrons. The measured flux $\vec{F}$, in the ALE appraoch is as shown in equation **3.8**,

$$\vec{F} = \int_S q(\vec{v} - \vec{w}_x) \cdot \vec{n} dS, \tag{3.8}$$

where $\vec{v}$ is the velocity of the electrons, $\vec{w}_x$ the velocity of the probe motion and $\vec{n}$ the normal to dS, the infinitesimal cross sectional surface element the electron passes through. Correctly calculating the flows passing through a section moving at velocity $\vec{w}_x$, is vital for ensuring the conservation of mass, momentum and energy [2]. The ALE approach, just like the Lagrangian approach, also makes use of a reference domain, however, the major difference from the Lagrangian approach is that the motion of the reference domain does not follow the motion of the fluid itself [4].

3.2.3. ALE Formulation for Fluid Mechanics

In order to correctly resolve the fluid solution on moving domains, an ALE formulation of the governing equations of fluid flow is essential. This formulation is based on effecting the space-time case of the Piola transformation [5] upon the Navier-Stokes Equations for incompressible flow given by equation **3.3a** . The Piola Transformation is a classical result from continuum mechanics which defines a vector field on a reference domain, when given an arbitrary vector field on a spacial domain. A salient feature of this transformation is that it preserves the conservation structure of the vector field in the reference configuration. The partial differential conservation form of the ALE formulation for the Navier -Stokes Equations for incompressible flow in vector and index notation, respectively, are given below.

$$\frac{1}{\hat{J}} \frac{\partial \hat{J} \rho \mathbf{V}}{\partial t}\Big|_{\hat{\mathbf{x}}} + \nabla \cdot (\rho \mathbf{V} \otimes (\mathbf{V} - \hat{\mathbf{u}}) - \underline{\underline{\mathbf{T}}}) - \rho \mathbf{f} = 0, \tag{3.9}$$

$$\frac{1}{\hat{J}} (\hat{J} \rho V_i)_{,t}\Big|_{\hat{\mathbf{x}}} + \left(\rho V_i (V_j - \hat{u}_j) - T_{ij}\right)_{,j} - \rho f_i = 0, \tag{3.10}$$

where, $\hat{J}$ is the determinant of the deformation gradient from the reference domain to the space-time domain, $\hat{\mathbf{u}}$ is the velocity of the reference domain and $\hat{\mathbf{x}}$ is the co-ordinates in the reference domain and $\Big|_{\hat{\mathbf{x}}}$ denotes the time derivative taken holding $\hat{\mathbf{x}}$ fixed. Further details on casting the fluid equations into the ALE formulation can be referred to in [4]. The boundary conditions used for the simulation setup are as follows:

- The inlet boundary conditions are selected as inhomogeneous Dirichlet for velocity and homogeneous Neumann for pressure.

$$\mathbf{V} = \langle \mathbf{V}_{inlet} \rangle, \tag{3.11a}$$

$$\frac{\partial p}{\partial \hat{\mathbf{n}}} = 0, \tag{3.11b}$$

  where $\mathbf{V}$ and p are the velocity and pressure fields at the boundary, respectively and $\hat{\mathbf{n}}$ is the vector normal to the inlet boundary.

- The outlet boundary conditions are selected as homogeneous Neumann for velocity and homogeneous Dirichlet for pressure.

$$\frac{\partial \mathbf{V}}{\partial \hat{\mathbf{n}}} = 0, \tag{3.12a}$$

$$\mathbf{p} = 0, \tag{3.12b}$$

  where $\mathbf{V}$ and p are the velocity and pressure fields at the boundary, respectively and $\hat{\mathbf{n}}$ is the vector normal to the outlet boundary.

- The no-slip wall boundary condition is defined as homogeneous Dirichlet for velocity and homogeneous Neumann for pressure.

$$\mathbf{V} \cdot \hat{\mathbf{n}} = 0, \tag{3.13a}$$

$$\mathbf{V} \cdot \hat{\mathbf{t}} = 0, \tag{3.13b}$$

$$\frac{\partial p}{\partial \hat{\mathbf{n}}} = 0, \tag{3.13c}$$

  where $\mathbf{V}$ and p are the velocity and pressure fields at the wall boundary, respectively, $\hat{\mathbf{n}}$ is the vector normal to the wall boundary and $\hat{\mathbf{t}}$ is the vector tangential to the wall boundary.

- The slip wall boundary condition is defined as homogeneous Dirichlet for normal velocity, homogeneous Neumann for tangential velocity and homogeneous Neumann for pressure.

$$\mathbf{V} \cdot \hat{\mathbf{n}} = 0, \tag{3.14a}$$

$$\frac{\partial \mathbf{V} \cdot \hat{\mathbf{t}}}{\partial \hat{\mathbf{n}}} = 0, \tag{3.14b}$$

$$\frac{\partial p}{\partial \hat{\mathbf{n}}} = 0, \tag{3.14c}$$

  where $\mathbf{V}$ and p are the velocity and pressure fields at the wall boundary, respectively, $\hat{\mathbf{n}}$ is the vector normal to the wall boundary and $\hat{\mathbf{t}}$ is the vector tangential to the wall boundary.

A pictorial representation of the case setup with the used boundary conditions is displayed in **Fig 3.3**

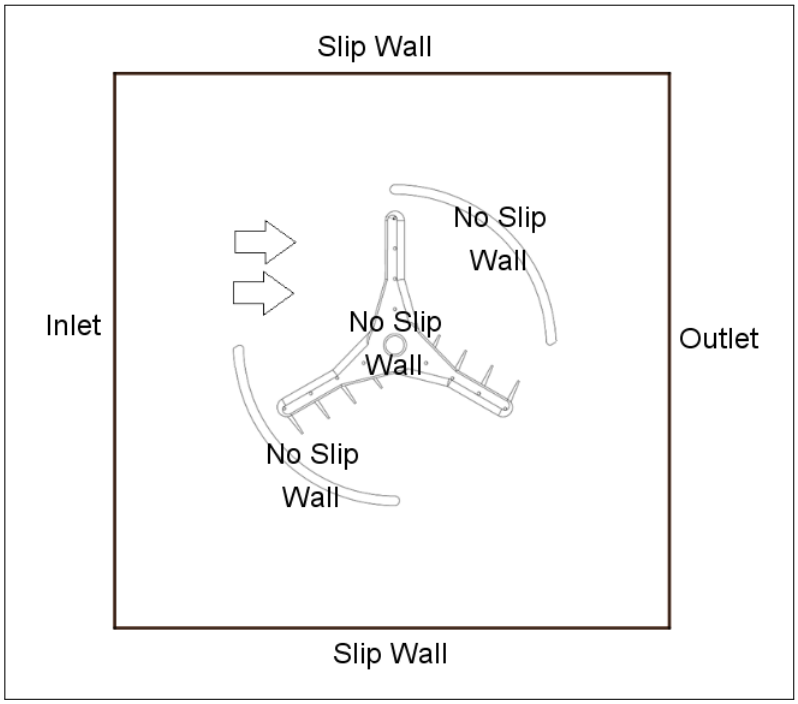


Figure 3.3: Boundary Conditions of Case Setup

A stationary divergence free velocity vector field is used as an initial condition to the fluid model in the considered domain. It is a solution to a stationary laminar flow problem that has the same boundary conditions as that of the transient problem for which it is used as an initial condition.

3.3. Structural Model

The structural model involves the description of the nested motion of the lamellas, which consists of a rotational and translational part. Enlisted below are the special case considerations for the structural solution related to the nested motion of the OWM lamellas.

- The FSI approach resolves the rotational dynamics of the lamella in its local co-ordinate system, with its linear or orbital dynamics considered a function of the rotational speed prescribed to the rotor arms .
- The OWM being a vertical Axis Turbine (VAT), the local rotation of the lamellas are fully described by a rotation about a single vertical axis passing through its pivot point. This reduces the problem down to resolving the rotational dynamics about the z-axis in the Cartesian co-ordinate system.
- An entire lamella is considered as a single solid body and undergoes no deformation. By solid body, we refer to the idea that each lamella of the turbine functions as a single consolidated entity made up of an ensemble of particles and behaves as a lumped structure with regard to its dynamics.
- With respect to the translational motion, the lamella is represented by its central pivot point.

The local rotational component of the structural model for the lamellas based on the corresponding special considerations listed above are given by the following system of first order ordinary differential equations:

$$I_z \frac{d\phi_z}{dt} = L_z, \tag{3.15a}$$

$$\frac{dL_z}{dt} = \tau_z, \tag{3.15b}$$

where, ϕ_z and I_z are the angular displacement and moment of inertia of the body about the z-axis passing through its pivot point, while L_z and τ_z are the angular momentum and the torque acting on the body along the z-axis passing through its pivot point.

The translational component of the structural model for the lamellas is based on its circular orbit about the central shaft of the turbine at constant angular velocity. The central shaft axis passes through the origin and is parallel to the z-axis. The relations describing this motion as as formulated below.

$$\frac{\partial\theta}{\partial t} = \omega_z \Rightarrow \theta = \omega_z t, \tag{3.16a}$$

$$\mathbf{v} = \omega_z \times \mathbf{x}_{old}, \tag{3.16b}$$

$$\mathbf{x}_{new} = \mathbf{v}\mathbf{x}_{old}, \tag{3.16c}$$

where, θ is the angular displacement of the lamella about the turbine shaft axis, ω_z is the the angular velocity of orbit along the turbine shaft axis, $\mathbf{v}$ is the velocity tangential to the orbital path, with $\mathbf{x}_{old}$ and $\mathbf{x}_{new}$ being the old and new positional vectors of the lamella pivot point. It must be noted that equation **3.16a** is reduced to a simple equation as the angular velocity of orbit is constant in the given problem.

The structural model is initialised with a null vector for angular momentum (L_z) and torque (τ_z) about the lamella axis with a non-zero vector for the angular velocity (ω_z) about the turbine central axis.

3.4. Discretisation of the Fluid Model

The Finite Volume Method is used as the means of numerically solving the fluid model equations. The FVM approach involves the tessellation of the spacial domain, wherein it is sub-divided into non-overlapping boundary conforming control volumes or finite volume cells. This leads to a larger construct known as meshes or grids, which contain the definition of each finite control volume along with its connectivity, in the considered domain. Numerically solving on these grids involves truncating an infinite series, which leads to the solved values being an approximation with an inherent solution error known as discretisation error. This method is pivoted on the conversion of volume integrals to surface integrals using the divergence theorem and then evaluating them as fluxes at the surfaces of each finite volume cell. It capitalises on the conservative nature of the fluxes flowing in and out of a control volume cell. For further details on the FVM, refer [6].

The slight deviation in notation within this section includes:

- The discretisation operators are denoted in bold, while the vectors are denoted normally.
- The subscript is not relevant to the index notation and refers to the finite volume element the term is associated with.

The discretised finite volume formulation of the fluid equations in Arbitrary Lagrangian Eulerian form can be schematically written as,

$$\frac{d}{dt}\left(V_i(t)\mathbf{W}_i\right) + \mathbf{\Phi}_i^c\left(\mathbf{W}_j, x, \dot{x}\right) = \mathbf{R}_i\left(\mathbf{W}_j, x\right). \tag{3.17}$$

In this relation,

- V_i represents the volume of the finite volume cell i.
- $\mathbf{W}_i$ is the discretised value of the unknowns at the center of the cell, representing an average over the cell. Similarly, $\mathbf{W}_j$ is the same, but for the neighbouring cell j.
- $\mathbf{\Phi}_i^c$ is the discretised convective term that includes the convection term related to mesh motion.
- $\mathbf{R}_i$ represents the discretised contribution of the source and viscous terms.
- x is a location in space and $\dot{x}$ is the speed of the moving reference frame (finite volume element).

3.5. Discretisation of the Structural Model

The rotational component of the structural model is solved through time integration. The handling of the non-smooth dynamics of the lamella requires a review of the time integration method utilised in the structural solver. With Leapfrog integration being the choice of time integration method of solving 3.15, consider a second order system of first order differential equations describing the motion of a particle with unit mass, as seen below.

$$\frac{d\mathbf{x}}{dt} = \mathbf{v},, \tag{3.18a}$$

$$\frac{d\mathbf{v}}{dt} = \mathbf{F}, \tag{3.18b}$$

where $\mathbf{x}$ is the position of the particle, $\mathbf{v}$ is the velocity of the particle and $\mathbf{F}$ is the force acting on the particle. The second order system of differential equations, 3.18a and 3.18b, are numerically integrated over time using Leapfrog Integration. The Leapfrog Integrator is a second order accurate, time integration scheme which defines the velocity and positions in a staggered manner along the progression of time. More specifically, the velocity defined at the mid-points of the interval is utilised to define the position at forward integral positions. The sketch below summarises the staggered approach of the Leapfrog integration scheme.

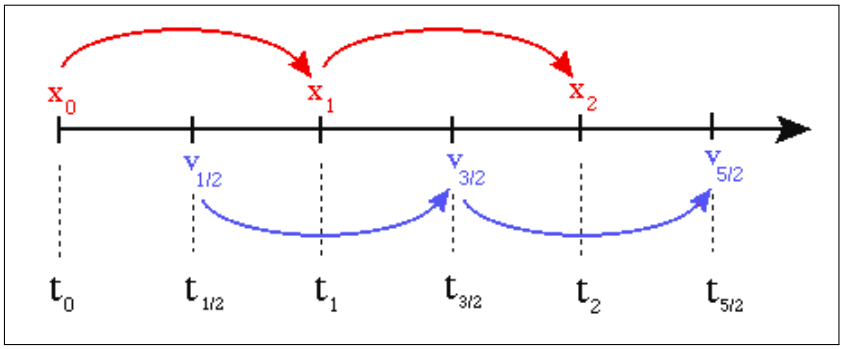


Figure 3.4: Leapfrog Integration Scheme

The numerical algorithm for such an approach with an initialisation of $\mathbf{x}^0$ and $\mathbf{v}^0$ is given below,

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{v}^{n+\frac{1}{2}}\Delta t, \tag{3.19a}$$

$$\mathbf{v}^{n+\frac{3}{2}} = \mathbf{v}^{n+\frac{1}{2}} + \mathbf{F}^{n+1}\Delta t, \tag{3.19b}$$

where $n = \{0, Z^+\}$. Using the Leapfrog integrator poses two concerns, which are enlisted below:

- The algorithm requires to be started off by calculating the first half step of velocity, i.e $\mathbf{v}^{n+\frac{1}{2}}$.
- Both velocity and position are not available at the same instance of time.

This matter can be resolved by a single alteration to the leapfrog integration methodology. By splitting 3.19b into two half steps with 3.19a ordered in between them. By this modification, a starting mechanism is embedded into the algorithm and the value of velocity at the same time instance of position, is made available. This method is known as the velocity Verlet method and requires an additional initialisation of force or acceleration at the start. Being a modification on the leapfrog algorithm, it is of second order accuracy as well.

Applying the velocity Verlet method to the second order system of differential equations given in 3.15a and 3.15b, the numerical algorithm for the considered structural solver is arrived at. With $i = \{0, Z^+\}$ and all superscripts referring to the time step number, the numerical integration formula for the considered integration scheme is initialised by the entities, L_z^0, ϕ_z^0, τ_z^0, and

$$\underline{\underline{\mathbf{Q}}}^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

In this initialisation step, $\underline{\underline{\mathbf{Q}}}^0$ is the orientation tensor describing the rotational orientation of the body. The numerical algorithm starts with an angular momentum half step,

$$L_z^{n+\frac{1}{2}} = L_z^n + \frac{\gamma \tau_z^n \Delta t}{2}, \tag{3.21}$$

where γ is the acceleration damping. The angular momentum half step in **3.21** is used to define the full step change in angular position,

$$\Delta\phi_z^{n+1} = \frac{L_z^{n+\frac{1}{2}}\Delta t}{I_z}. \tag{3.22}$$

The change in angular position, denoted by $\Delta\phi_z^{i+1}$, is used to define the rotational tensor that describes the corresponding incremental rotation about the z-axis,

$$\underline{\underline{\mathbf{R}}}_z^{n+1} = \begin{bmatrix} \cos\Delta\phi_z^{n+1} & -\sin\Delta\phi_z^{n+1} & 0 \\ \sin\Delta\phi_z^{n+1} & \cos\Delta\phi_z^{n+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.23}$$

The rotational tensor, $\underline{\underline{\mathbf{R}}}_z^{n+1}$, is then used to update the orientation tensor to the position description corresponding to the next time step.

$$\underline{\underline{\mathbf{Q}}}^{n+1} = \underline{\underline{\mathbf{Q}}}^{n} \cdot \underline{\underline{\mathbf{R}}}_z^{n+1}. \tag{3.24}$$

Subsequently, the acceleration is updated by calculating the net force from the sufficiently converged fluid solution. In all steps except the zeroth step, acceleration relaxation acceleration relaxation is performed as follows,

$$\tau_z^{n+1} = \lambda\tau_z^{n+1} + (1-\lambda)\tau_z^{n} \quad | \quad i > 0, \tag{3.25}$$

where λ is the acceleration relaxation factor. Finally, in preparation for the next iteration, the angular momentum correction step is undertaken using the updated acceleration.

$$L_z^{n+1} = L_z^{n+\frac{1}{2}} + \frac{\gamma\tau_z^{n+1}\Delta t}{2}, \tag{3.26}$$

4

Modelling Blocks in the Structural Solver

A model is a physical, mathematical, or logical representation of a system entity, phenomenon, or process. A simulation is the implementation of a model over time [7].

Having established the OWM's functional characteristics and testing procedure in **Chapter 1** and **Chapter 2** , the matter of defining the required modelling approaches to embed these characteristics into structural solver is addressed. The modelling concepts explored in detail within the forthcoming sections are enlisted below.

- The structural solution is effected in the FSI problem using a non-conformal dynamic mesh handling methodology known as a sliding mesh interface, described in **Section 4.1.1**. Being a non-conformal interface method, it uses an interpolation scheme to transfer flux data across the interface. The sliding mesh interface is used to implement the rotational motion of the twelve lamellas and the rotor arms.

- A mesh zone is a collection of a particular set of finite control volumes that represent a subset of the discretised domain upon which the simulation is carried out. The mesh zones pertinent to the simulations carried out are defined in **Section 4.1.1**. The mesh zones outlined by a sliding mesh interface and connected to the lamella surface, exist within the mesh zone of the three rotor arms. Thus, the description of the lamella mesh zone's motion dynamics include both, a rotational as well as an orbital characteristic, as formulated in **Section 3.3**. This results in a system of nested moving mesh zones, which is captured in the numerical simulation by embedding nested motion dynamics within the developed solver.

- Due to the limitations originating from the use of sliding mesh interfaces for the dynamic mesh handling, the lamella rotations are modelled as rotations pivoted at the center, rather than at its leading edge. This calls for a modelling approach to ensure the application of an FSI calculated angular displacement equivalent to being physically pivoted at its end, while being computationally pivoted at its center.

- A close examination of the mobility range of the lamellas, uncovers a 90 degree range of operation, which represents a non-smooth dynamical system whose motion stops abruptly at the 90 degree and 0 degree angular positions. It is deemed to be a non-smooth mechanical phenomena as the lamellas do not operate under gradual movement dynamics, but instead, stops intermediately and reverses the direction of movement. This non-smooth behaviour is encapsulated within the developed solver using an event-based methodology.
- The simulation ideology entails determining the torque generated by the turbine for different turbine rotational speeds. To this effect, from the resolved dynamics of the lamellas, their individual contributions to the total torque generated by the turbine is determined. By summing up the torque contributions determined for each lamella and averaging over a single period of the turbine's operation, a suitable estimation of the total turbine torque is achieved.

4.1. Dynamic Mesh Handling

As the name suggests, FSI problems involve the determination of the structural response based on the fluid forcing and vice-versa. This establishes the existence of a fluid-structure boundary in every FSI problem and requires its dynamical behaviour to be reflected in the discretisation grid being used. The update of the grid according to the resolved structural dynamics of the problem is known as dynamic mesh handling. In other words, the nodes that define the cells in the domain must be updated as a function of time, which results in the mesh solutions being inherently unsteady.

As the physical problem of the OWM involves moving boundaries, i.e turbine rotor arms and its nested lamellas, this modelling concept applies to the resolution of the current problem statement. The particular treatment of meshes to account for the dynamic behaviour of the boundaries introduces another classification within FSI analyses, subgrouped as conformal and non-conformal mesh methods. These methods differ based on the nature of the interface used to facilitate the dynamic handling of the mesh.

Conformal mesh methods require the use of a conformal interface between the relatively dynamic mesh zones, such that nodal connectivity is maintained across the interface while keeping up with the domain motion. This can be achieved by mesh deformation or displacement up to a certain extent of motion or by local or global re-meshing. Non-conformal mesh methods involve the use of a non-conformal interface between the dynamic mesh zones wherein nodal connectivity is not required to be maintained and data is transferred across the interface using an interpolation scheme. Such a scheme allows for a wider range of dynamic mesh motion without having to resort to re-meshing. The computational savings and larger freedom of mesh movement is offset by accuracy concerns while using interpolation methods for data transfer across interfaces. An example of a non-conformal mesh interface that exists between different mesh zones is depicted in **Fig 4.1**.

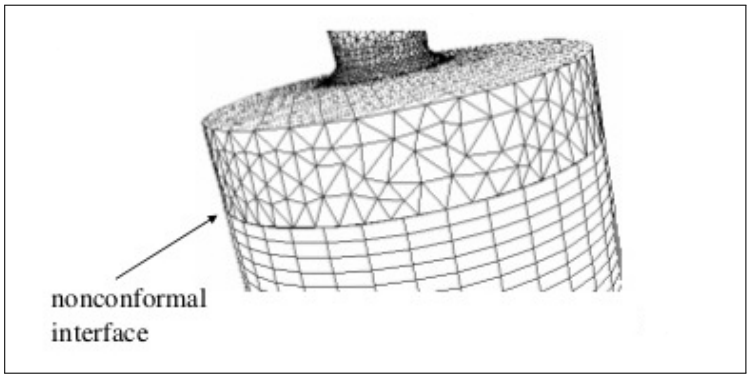


Figure 4.1: Non-Conformal Mesh Interface Between Two Mesh Zones.

4.1.1. Sliding Mesh

A special case of a non-conformal dynamic mesh handling method in which all of the boundaries and the cells of a given mesh zone move together as a rigid-body motion is known as a sliding mesh. In this case, the nodes of the mesh zone move in a consolidated manner, wherein the cells within the considered cell zone do not deform. As long as the interfaces stay in contact with one another,"slide" along the interface boundary, the non-conformal interfaces can be dynamically updated and fluid can computationally "flow" from one zone to the other.

Sliding meshes see a strong applicability in FSI problems related to rotational motion, such as the simulation of turbine functionality. A fundamental requirement for the use of sliding meshes is that their interface boundaries must not intersect. Geometrical care is to be taken in defining a suitable sliding mesh interface and its corresponding FSI boundary. The sliding mesh interface methodology is applied to the OWM by truncating its lamellas and shifting the point of computational rotation from its physical pivot point to the center of the mesh zone. **Fig** 4.2 presents the geometric transformation performed to allow this means of dynamic mesh handling.

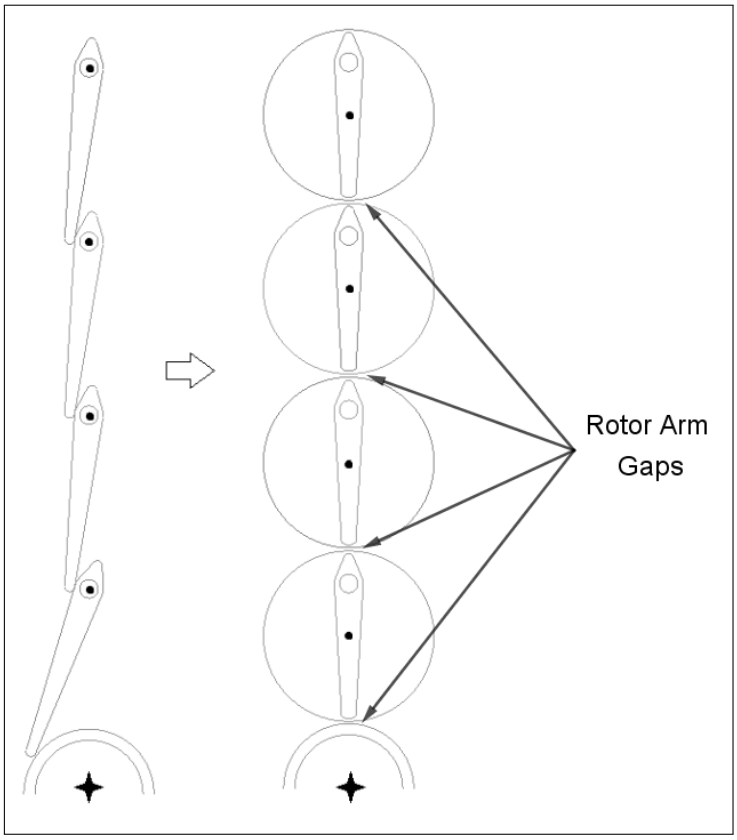
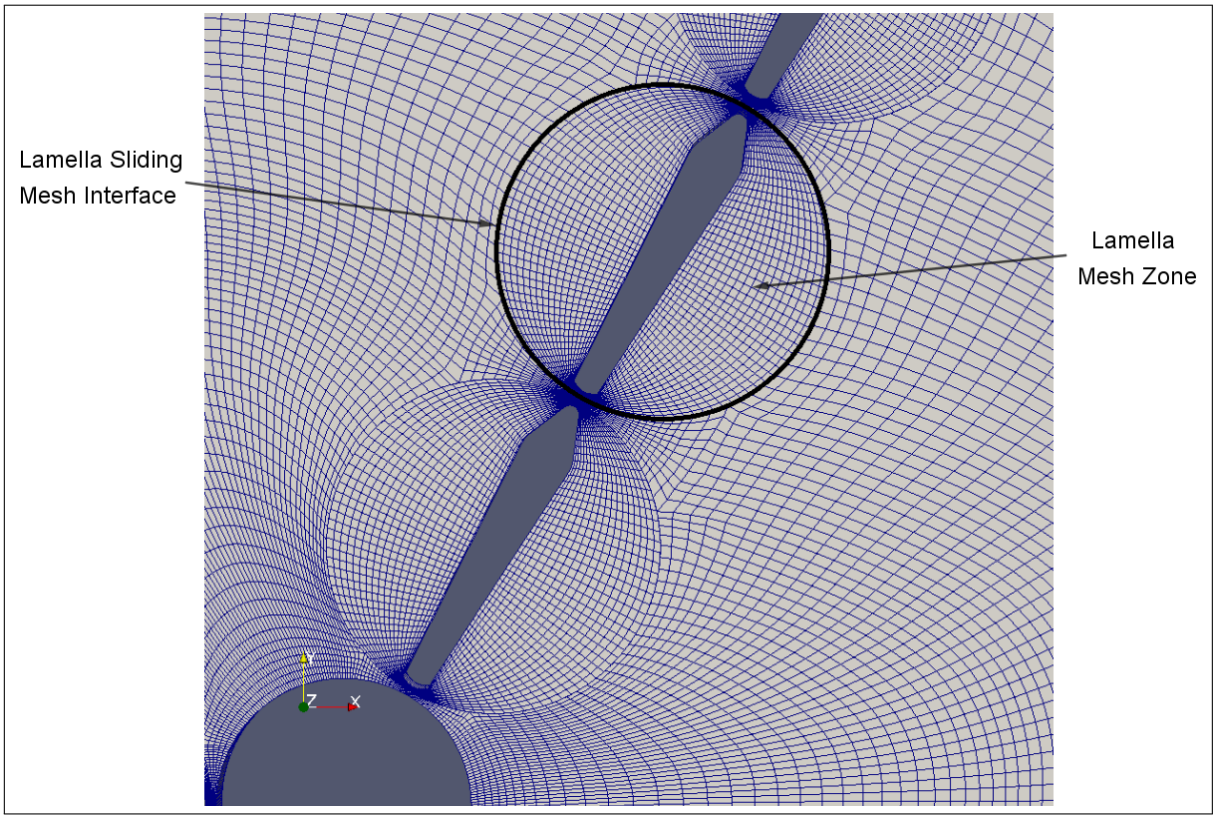
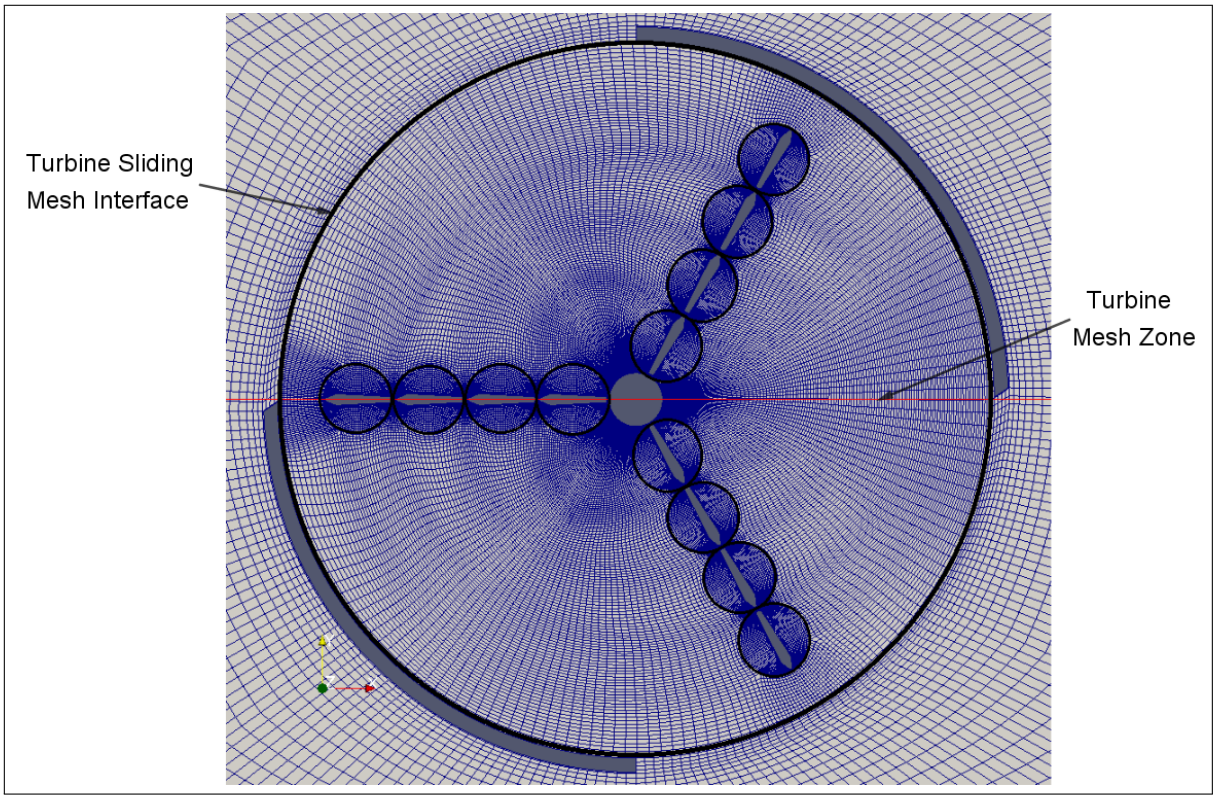


Figure 4.2: Geometry Transformation to Computational Model

The lamella arrangement on the left in **Fig 4.2**, represents the true geometric particulars of a single rotor arm and its component lamellas, whereas, that on the right represents the geometry of a single rotor arm with re-approximated lamellas outlined by their circular non-conformal interfaces. The points on each arrangement depict the lamella pivot position in the physical and computational models, respectively. The lamella gaps in the computational model are identified in **Fig 4.2** and are a byproduct of ensuring the non-conformal interfaces do not intersect. The region of the lamella grid enclosed by the circular sliding mesh interface is referred to as the Lamella Mesh Zone (LMZ) and is depicted in **Fig 4.3a**. The region of the turbine grid within the casing and excluding the twelve LMZ's, is known as the Turbine Mesh Zone (TMZ) and is depicted in **Fig 4.3b**. The intersection points of the grid lines are referred to as the nodes of the mesh zone.



(a) OWM Lamella Mesh Zone



(b) OWM Turbine Mesh Zone

Figure 4.3: OWM Dynamic Mesh Zones

4.2. Imposed Nested Solid Body Motion

From the computational model on the right in **Fig 4.2**, it can be seen that the dynamics of the LMZ involves rotation about an axis perpendicular to the 2D plane and passing through its pivot point (demarcated by a black dot), as well as, revolution about an axis perpendicular to the 2D plane and passing through the center of the turbine (demarcated by the black star). The motion dynamics of the TMZ involves only a rotation about an axis perpendicular to the 2D plane and passing through the turbine center. **Section 3.3** involved the means of solving for the attributes that define the motion of the body at a given time step. Once these attributes are determined, a suitable means of implementing them upon a dynamic mesh zone is to be defined. The forthcoming sections introduce the means by which rotational and translational dynamics are effected upon the dynamic mesh zones.

4.2.1. Rotational Motion Implementation

The rotational motion in this study is implemented using quaternions. Quaternions form a number system $\mathbb{H}$ with three imaginary dimensions and a real part. They can represent 3D orientations/rotations and can do so without being hindered by the Gimbal lock that Euler angles suffer from. Gimbal lock is the loss of one degree of freedom in a three-dimensional, three-gimbal mechanism. This occurs when the axes of two of the three gimbals are driven into a parallel configuration, "locking" the system into rotation in a degenerate two-dimensional space. Further reading on the Euler angles and the Gimbal lock can be found in [8]. The quaternion number system is defined as below:

$$\mathbb{H} = \{x_0 + x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k} \quad | \quad x_0, x_1, x_2, x_3 \in \mathbb{R}\}, \tag{4.1a}$$

$$q \in \mathbb{H}, \quad q = x_0 + x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}. \tag{4.1b}$$

Quaternion multiplication is defined by,

$$\begin{aligned}
&(x_0 + x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}) \cdot (y_0 + y_1\mathbf{i} + y_2\mathbf{j} + y_3\mathbf{k}) \\
&= (x_0y_0 - x_1y_1 - x_2y_2 - x_3y_3) \\
&+ (x_0y_1 + x_1y_0 + x_2y_3 - x_3y_2)\mathbf{i} \\
&+ (x_0y_2 - x_1y_3 + x_2y_0 - x_3y_1)\mathbf{j} \\
&+ (x_0y_3 + x_1y_2 - x_2y_1 + x_3y_0)\mathbf{k}.
\end{aligned} \tag{4.2}$$

An alternative short hand representation of a quaternion, can be written by identifying $Re(\mathbb{H})$ with $\mathbb{R}$ and $Im(\mathbb{H})$ with $\mathbb{R}^3$. This representation is shown below.

$$\mathbf{q} = \alpha + \mathbf{x}, \tag{4.3}$$

where, $\mathbf{q} \in \mathbb{H}$, $\alpha \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^3$.
Quaternions have the following properties:

1. Quaternion multiplication is not commutative,

$$\mathbf{pq} \neq \mathbf{qp}, \tag{4.4}$$

where, $\mathbf{p}$, $\mathbf{q} \in \mathbb{H}$.

2. Quaternion multiplication is associative,

$$(\mathbf{pq})\mathbf{r} = \mathbf{p}(\mathbf{qr}), \tag{4.5}$$

where, $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r} \in \mathbb{H}$.

3. Conjugate of a quaternion is defined as,

$$\bar{\mathbf{q}} = \alpha - \mathbf{x}, \tag{4.6}$$

where, $\mathbf{q} = \alpha + \mathbf{x}$.

4. Quaternions inherit the Euclidean norm,

$$|\mathbf{q}| = (\sum_{i=0}^{3} q_i^2)^{\frac{1}{2}}, \tag{4.7a}$$

$$|\mathbf{q}|^2 = \mathbf{q}\bar{\mathbf{q}}, \tag{4.7b}$$

where $\mathbf{q} \in \mathbb{H}$.

5. Inverse of a quaternion is defined as,

$$\mathbf{q}^{-1} = \frac{\bar{\mathbf{q}}}{|\mathbf{q}|^2}, \tag{4.8}$$

where $\mathbf{q} \in \mathbb{H}$.

The following construction relates the use of quaternions to describe rotation in $\mathbb{R}^3$. Consider, $a \in \mathbb{R}^3$, $|a| = 1$, $\alpha \in \mathbb{R}$ and $\mathbf{q} = \cos(\frac{\alpha}{2}) + \sin(\frac{\alpha}{2})\mathbf{a})$. Then for all $\mathbf{y} \in \mathbb{R}^3$ we have:

- $\mathbf{q}\mathbf{y}\bar{\mathbf{q}} \in Im(\mathbb{H}) = \mathbb{R}^3$.
- The map $F : \mathbb{R}^3 \to \mathbb{R}^3$, $\mathbf{y} \mapsto \mathbf{q}\mathbf{y}\bar{\mathbf{q}}$ is a rotation around axis $\mathbf{a}$ by an angle α, when $\mathbf{y}$ is the global positional co-ordinates of a node in the mesh zone being rotated. The map F is explicitly defined as,

$$F(\mathbf{y}) = \cos(\alpha) + \sin(\alpha)(\mathbf{a} \times \mathbf{y}). \tag{4.9}$$

Example : Consider the three standard Cartesian unit vectors $\hat{\mathbf{i}} = (1,0,0)$, $\hat{\mathbf{j}} = (0,1,0)$ and $\hat{\mathbf{k}} = (0,0,1)$. If one was to rotate $\hat{\mathbf{i}}$ about $\hat{\mathbf{k}}$ by an an angle $\alpha = \frac{\pi}{2}$, the required quaternion is as defined in the construction above,

$$\mathbf{q} = \cos(\frac{\pi}{4}) + \sin(\frac{\pi}{4})\hat{\mathbf{k}}. \tag{4.10}$$

Performing the rotation using the corresponding map defined above,

$$F(\hat{\mathbf{i}}) = \mathbf{q}\hat{\mathbf{i}}\bar{\mathbf{q}}, \tag{4.11}$$

$$= \cos(\frac{\pi}{2}) + \sin(\frac{\pi}{2})(\hat{\mathbf{k}} \times \hat{\mathbf{i}}), \tag{4.12}$$

$$= \hat{\mathbf{j}} \tag{4.13}$$

4.2.2. Full Motion Description

The grouping together of a translation describing vector and a rotation describing quaternion, forms a tuple or pair known colloquially within the OpenFOAM community as a septernion. The septernion is a cumulative description of both the rotation and translation of a body and can be represented as displayed below:

$$\mathbb{S} = \{(l_1\mathbf{i} + l_2\mathbf{j} + l_3\mathbf{k}), (x_0 + x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}) \quad | \quad x_0, x_1, x_2, x_3, l_1, l_2, l_3 \in \mathbb{R}\}, \tag{4.14a}$$

$$s \in \mathbb{S}, \quad s = \big((l_1\mathbf{i} + l_2\mathbf{j} + l_3\mathbf{k}), (x_0 + x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k})\big). \tag{4.14b}$$

Septernion operations with vectors involve addition/subtraction and represent a linear motion update to the septernion (vector addition).

$$(\mathbf{x}, \mathbf{q}) \pm \mathbf{y} = (\mathbf{x} \pm \mathbf{y}, \mathbf{q}), \tag{4.15}$$

where, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ and $\mathbf{q} \in \mathbb{H}$.
Septernion operations with quaternions involve multiplication operations and represent a rotational motion update to the septernion (quaternion multiplication).

$$(\mathbf{x}, \mathbf{q}) * \mathbf{p} = (\mathbf{x}, \mathbf{q} * \mathbf{p}), \tag{4.16}$$

where, $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{q}, \mathbf{p} \in \mathbb{H}$.
Septernion operations with another septernion involve vector addition, quaternion multiplication and the rotation of a vector by a quaternion.

$$(\mathbf{x}, \mathbf{q}) * (\mathbf{y}, \mathbf{p}) = (\mathbf{x} + \mathbf{p}\mathbf{y}\bar{\mathbf{p}}, \mathbf{q} * \mathbf{p}), \tag{4.17}$$

where, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ and $\mathbf{q}, \mathbf{p} \in \mathbb{H}$.

The final septernion formed to represent the motion dynamics of the considered lamella for every time step, is calculated as follows:

$$s^n = (\mathbf{x^n}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}) * \mathbf{q}^n * (\mathbf{x^0}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}), \tag{4.18}$$

where, n is the time step number, $\mathbf{x} \in \mathbb{R}^3$ is the position vector from origin to center of considered lamella mesh zone, $\mathbf{q} \in \mathbb{H}$ is the quaternion calculated from the rotational change between orientation tensors $\underline{\underline{\mathbf{Q}^n}}$ and $\underline{\underline{\mathbf{Q}^0}}$.

At any given time step, the rotational component of the LMZ motion is described by a quaternion and the revolution about the turbine shaft axis is described by a simple translation vector connecting the old and new centers of the mesh zone. The overall motion of a single LMZ is described by a conjoined entity formed by the combination of the quaternion and the translation vector, known as a septernion. Effecting the septernion upon each node in its corresponding LMZ, results in the dynamic mesh implementation for the lamellas. Similarly, the overall motion of the TMZ is described by a single

quaternion. Effecting a pure rotation septernion, built out of the quaternion describing the motion of the TMZ, upon each node in the region results in the dynamic mesh implementation for the turbine arms.

4.3. Virtual Hinging of Center Pivoted Lamellas from Leading Edge

The lamellas are pivoted at their center as a modelling approximation taken up in order to allow for the implementation of the sliding mesh interface method as a means of dynamic mesh handling. In order to remain in-line with the dynamics of the OWM, whose lamellas are pivoted at their leading edge, the virtual hinging modelling approach is adopted. The modelling methodology is built on the essence of conserving the closing and opening dynamic of the lamellas as seen in the physical case. Thus, the aim of the modelling approach is to effect an angular change on the center pivoted lamella that coincides with the angular change resulting from being pivoted at the leading edge of the lamella. This results in the need to calculate the net torque at the lamella leading edge and then re-calibrate this torque to be applied at the center. The re-calibration of the torque is performed by effecting it with a moment of inertia ratio, as described in the forthcoming section.

4.3.1. Moment of Inertia Ratio

Consider two systems made up of a thin body each. We refer to the first system as the physical system and the second as the computational system. Each body has a length L, with a mass M concentrated at its center of mass. The Pivot Point (PP) of the systems are demarcated by the ∘ symbol and the Torque Calculation Point (TCP) is demarcated by a × symbol. They are placed in a time invariant uniform vector field $\mathbf{F}(t,\mathbf{x}) = \mathbf{F}$, as shown in the figure below.

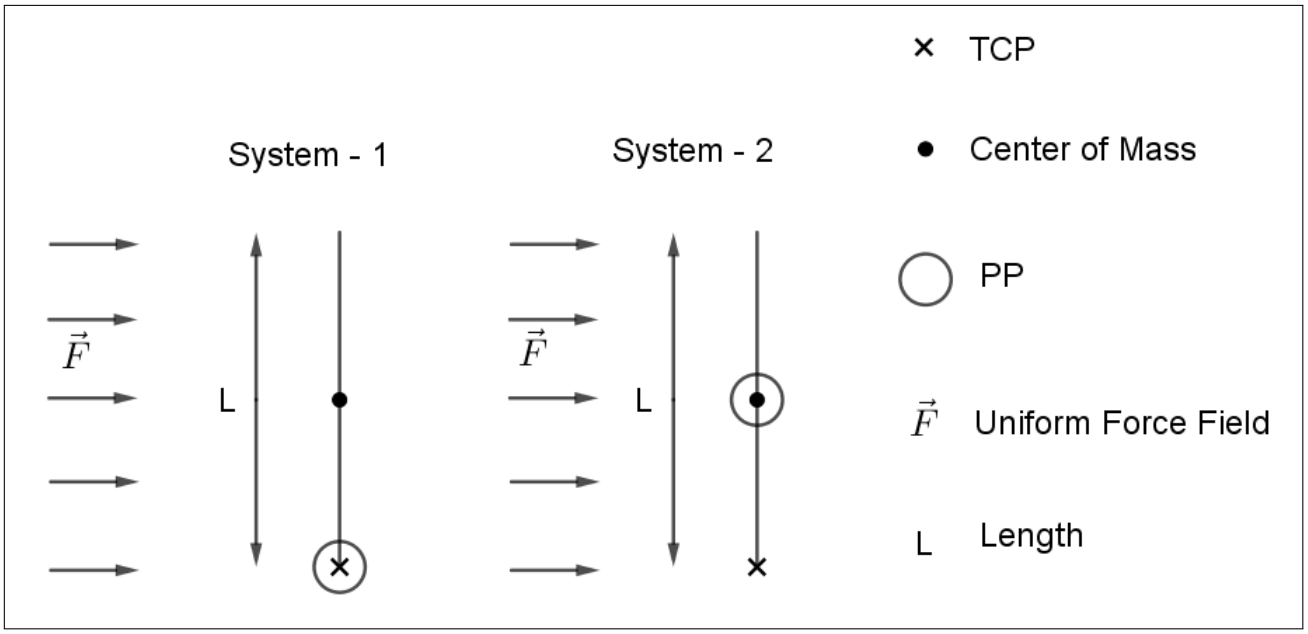


Figure 4.4: System Orientation

While the superscript j stands to signify the system number and suprscript s stands to sigfy the point position of torque consideration, assume two forms of torque realisations as follows,

- The numeric realisation of torque, $\boldsymbol{\tau}_n^{j,s} = \mathbf{r}_i^{j,s} \times \mathbf{F}_i^{j,s}$, wherein,
  - The body is discretised into N nodes, n_i,
  - The distance between the nodes and the point of torque consideration is $\mathbf{r}_i$, where i is the node position number.
- The analytic realisation of torque, $\boldsymbol{\tau}_a^{j,s} = I^{j,s}\boldsymbol{\alpha}^{j,s}$, wherein,
  - $\boldsymbol{\alpha}^{j,s}$ is the Angular acceleration of the body along the axis passing through the point of torque consideration and perpendicular to the 2D plane. The angular acceleration is the second time derivative of angular position, $\phi(t)$.
  - $I^{j,s}$ is the moment of inertia of the body about the axis passing through the point of torque consideration and perpendicular to the 2D plane.

System - 1

$$\boldsymbol{\alpha}^{1,\circ} = \ddot{\phi}(t)^{1,\circ}, \tag{4.19a}$$

$$\boldsymbol{\tau}_n^{1,\times} = \sum_{i=1}^{N} \mathbf{r}_i^{1,\times} \times \mathbf{F}_i^{1,\times}, \tag{4.19b}$$

$$\boldsymbol{\tau}_a^{1,\circ} = I^{1,\circ}\boldsymbol{\alpha}^{1,\circ} = I^{1,\circ}\ddot{\phi}(t)^{1,\circ}, \tag{4.19c}$$

System - 2

$$\boldsymbol{\alpha}^{2,\circ} = \ddot{\phi}(t)^{2,\circ}, \tag{4.20a}$$

$$\boldsymbol{\tau}_n^{2,\times} = \sum_{i=1}^{N} \mathbf{r}_i^{2,\times} \times \mathbf{F}_i^{2,\times}, \tag{4.20b}$$

$$\boldsymbol{\tau}_a^{2,\circ} = I^{2,\circ}\boldsymbol{\alpha}^{2,\circ} = I^{2,\circ}\ddot{\phi}(t)^{2,\circ}, \tag{4.20c}$$

From the equations for either system depicted above, it is evident that the numerical torque realisations in both systems are equivalent as the calculations are performed for the same body, at the same torque calculation point and effected by the same forcing vector field.

$$\boldsymbol{\tau}_n^{1,\times} = \boldsymbol{\tau}_n^{2,\times}. \tag{4.21}$$

As the point position of torque consideration in the first system overlap, the two forms of torque realisations are equivalent.

$$\boldsymbol{\tau}_n^{1,\times} = \boldsymbol{\tau}_a^{1,\circ}. \tag{4.22}$$

According to the aim of the modelling approach, the two systems are to have the same angular dynamics. Hence, their angular accelerations are set to being equivalent.

$$\boldsymbol{\alpha}^{1,\circ} = \boldsymbol{\alpha}^{2,\circ}. \tag{4.23}$$

From 4.21 and 4.22, we have,

$$\boldsymbol{\tau}_a^{1,\circ} = \boldsymbol{\tau}_n^{2,\times}. \tag{4.24}$$

Using **4.23** to equate **4.19c** and **4.20c**, we have,

$$\frac{\boldsymbol{\tau}_a^{1,\circ}}{I^{1,\circ}} = \frac{\boldsymbol{\tau}_a^{2,\circ}}{I^{2,\circ}}. \tag{4.25}$$

Setting **4.24** in **4.25**, we have,

$$\boldsymbol{\tau}_a^{2,\circ} = \frac{I^{2,\circ}}{I^{1,\circ}} \boldsymbol{\tau}_n^{2,\times}. \tag{4.26}$$

Thus, the required torque needed to be effected at the central pivot point of system-2, in order to ensure equivalent angular dynamics as that of system 1, is given in **4.26**. $\frac{I^{2,\circ}}{I^{1,\circ}}$ is referred to as the momentum ratio for the modelling approach.

4.4. Non-Smooth Mechanics

Adopting a top view of the OWM, as that depicted in **Fig 1.1a**, a rotational baseline is defined by establishing that the device rotates clockwise when impinged upon by the fluid while generating power. Considering a single lamella, its motion specifics are summarised in **Fig 4.5**, which is a visual representation of the description of its motion presented in **Chapter 1**.

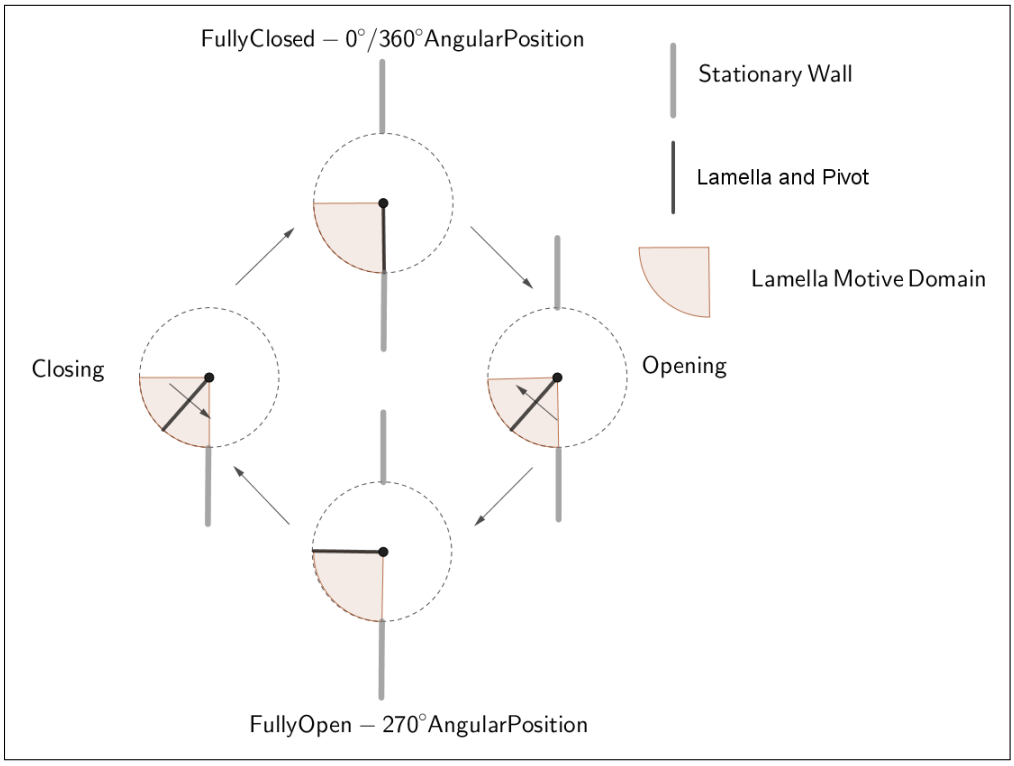


Figure 4.5: Lamella Motion Description

In order to describe the angular dynamics of the center pivoted lamella, the lamella local anugular position system is established. This angular position baseline sets precedence for further discussion later on in the chapter and is displayed in **Fig 4.6**.

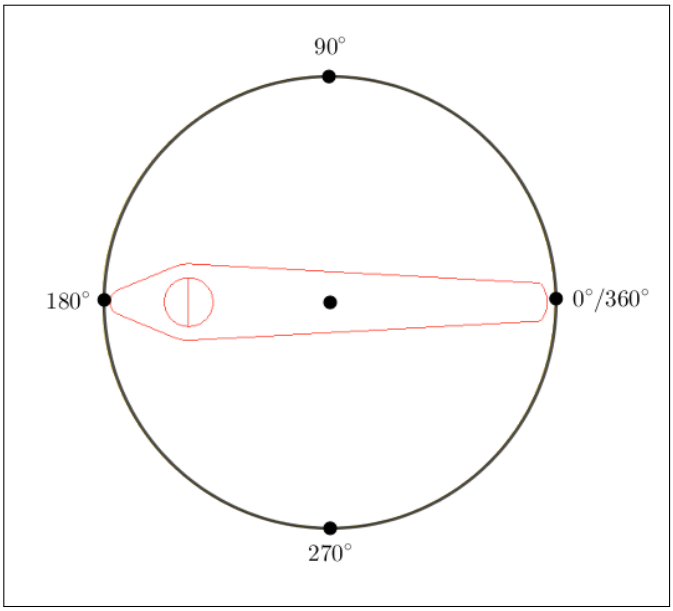


Figure 4.6: Lamella Local Angular Position

Physically, there exists a stopper which constrains the lamella from rotating in the anit-clockwise direction when in the fully closed or 360/0° angular position. Additionally a stopper exists at the fully open or 270° angular position and prevents the lamella from rotating any further in the clockwise direction. It is this stopper that ensures that the lamella is constrained, thereby allowing any force acting upon it to be transmitted to the rotor arms, resulting in the generation of power. If it were not for the stopper, the lamellas would merely rotate about their axes and not transmit any force over to the rotor arm it is nested in.

The stopper is effected in the simulation by defining a proximity region at the limiting angular position, which behaves as a perfect momentum sink. The perfect momentum sink character of the proximity region results in it being a zone within which the accumulated momentum of the lamella is instantaneously reduced to zero and the lamella is restrained from rotating clockwise or anti-clockwise (depending on the position of the proximity region encountered), any further.

This is implemented using an event selection system. The three events defined are CCW constraint, CW constraint and non-constraint. The selection of which event the lamella is undergoing is performed based on its angular position, its current torque, its current angular momentum and the definition of the proximity regions. A simplistic algorithm depicting this event selection system is enlisted in **Alg** 1. Once the event is determined, the lamella is either allowed to move freely or effected by the perfect momentum sink to instantaneously decelerate.

Algorithm 1 Non-Smooth Mechanics

procedure Angular Position Based Event Selection
  $\theta \leftarrow$ *Lamella Angular Position*
  $\tau \leftarrow$ *Lamella Torque at Pivot Point*
  $\pi \leftarrow$ *Lamella Momentum*
  $\Psi \leftarrow$ *Proximity Region Size*
  $P_C \leftarrow$ *Proximity Region Boundary - Fully Closed*
  $P_O \leftarrow$ *Proximity Region Boundary - Fully Open*
  if $\Big((\tau > 0 \quad || \quad \pi > 0) \quad \&\& \quad (\theta \geq (P_C - \Psi) \quad \&\& \quad (\theta \leq P_C))\Big)$ **then**
    Select *CCW Constraint Event*
  else
    if $\Big((\tau < 0 \quad || \quad \pi < 0) \quad \&\& \quad (\theta \leq (P_O + \Psi) \quad \&\& \quad (\theta \geq P_O))\Big)$ **then**
      Select *CW Constraint Event*
    else
      Select *Non-Constraint Event*

4.5. Turbine Torque Calculation

The means by which the total torque generated by the turbine is determined is defined in this section and the methodology is referred to as Turbine Torque Calculation (TTC). The TTC method involves the computation of the total torque generated by the turbine for a given time step. This involves the summation of the torque contributions from all twelve lamellas for the considered time step. The averaged TTC is computed by a summation of the TTC's for a range of time steps and dividing the amount by the number of time steps considered. The computed TTC for the two dimensional computational grid is finally scaled to the actual height of the turbine to relate to the actual dimensions of the problem. The representation of a single lamella's torque and force constructs are pictorially represented in **Fig** 4.7.

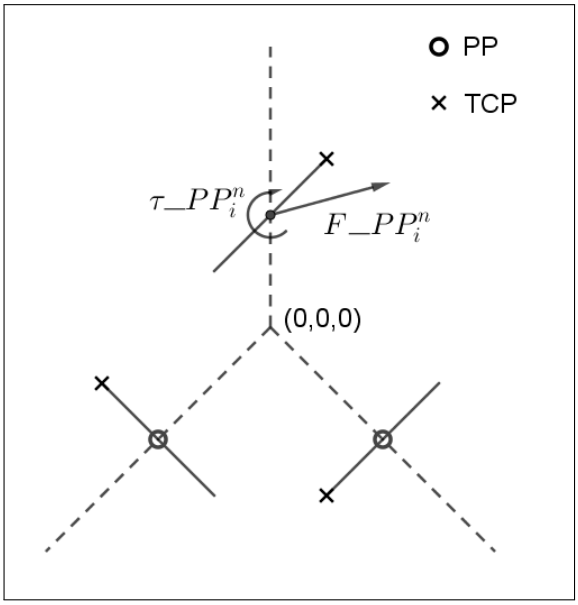


Figure 4.7: Turbine System

Establishing the following nomenclature,

- Subscript i refers to the lamella number such that, $i = 1, 2...11, 12$.
- Superscript n refers to the time step number, $n = 1, 2....$
- PP stands for pivot point of the lamella.
- $\mathbf{F}$ stands for the net force on the lamella and $\boldsymbol{\tau}$ stands for the net torque on the lamella.
- $\mathbf{r}$ is the position vector from origin/turbine center to the pivot point of the lamella.

The Torque Contribution (TC) of each lamella is defined dependent on the dynamic even the lamella is undergoing. When the lamella is fully closed or fully open, both $\mathbf{F}\_PP$ and $\boldsymbol{\tau}\_PP$ play a role. This results from the fact that when the lamella is in one of the constrained events, all forms of forcing on it is transmitted to the arm and thus contributes to the torque calculation for the lamella. When the lamella is in free motion or non-constrained motion, only $\mathbf{F}\_PP$ plays a role. This is due to the fact that the local torque on the lamella is used to rotate the lamella and remains untransmitted to the rotor arm. This is summarised as,

$$\text{Fully Open/ Fully Closed} \Rightarrow TC_i^n = (\mathbf{r}_i \times \mathbf{F}\_PP_i^n) + \boldsymbol{\tau}\_PP_i^n, \tag{4.27a}$$

$$\text{Non-Constrained} \Rightarrow TC_i^n = (\mathbf{r}_i \times \mathbf{F}\_PP_i^n). \tag{4.27b}$$

The GTC of the turbine for a single time step is computed as,

$$GTC^n = \sum_{1 \le i \le 12} TC_i^n. \tag{4.28}$$

The averaged GTC over a time step range from n_{start} to n_{end},

$$avg\_GTC = \sum_{\substack{n_{start} \le n \le n_{end} \\ 1 \le i \le 12}} TC_i^n / (n_{end} - n_{start}). \tag{4.29}$$

The averaged torque calculation re-scaled to the full height of the turbine is computed by,

$$cal\_avg\_GTC = avg\_GTC \frac{H}{th}, \tag{4.30}$$

where, H is the true height of the turbine and th is the single cell thickness of the two dimensional grid (Quasi 3D) generated for the simulation.

5

Analysis Process and Modelling Verification

The process involved in the numerical predictive analysis of the OWM is described in this chapter. A CFD analysis typically involves three phases, namely, pre-processing, computation and post processing. Pre-processing involves geometry idealisation and mesh generation. The computation phase includes simulation case setup and running the simulation. The post processing phase focuses on visualising the generated turbine numerical data. For the problem at hand, the process begins at the arrival of the requisite 3D model of the OWM and ends on determining the post processed results of the turbine simulations.

Additionally, this chapter documents the verification steps taken up during the development of the simulation mechanism for the OWM. The verification is performed for the implementation of each modelling concept individually. The logic behind the verification and the results for which the approach is considered to be verified is discussed. The nested solid body rotation, pivot point shift, non-smooth mechanics and the global torque calculation are the modelling approaches for which verification is carried out.

5.1. CFD Analysis Work Flow

The geometry of the OWM for which the numerical approach is to be developed is a 3D model. Having concluded to address the problem with a 2D formulation, this 3D geometry of the turbine has to be reduced to its 2D equivalent, while conserving as many features of its design as possible. Being a vertical axis turbine, the optimal choice of a 2D reduced geometry comes from considering the cross section of the device. The 3D model of the OWM and its selected 2D section is displayed below in **Fig** 5.1.

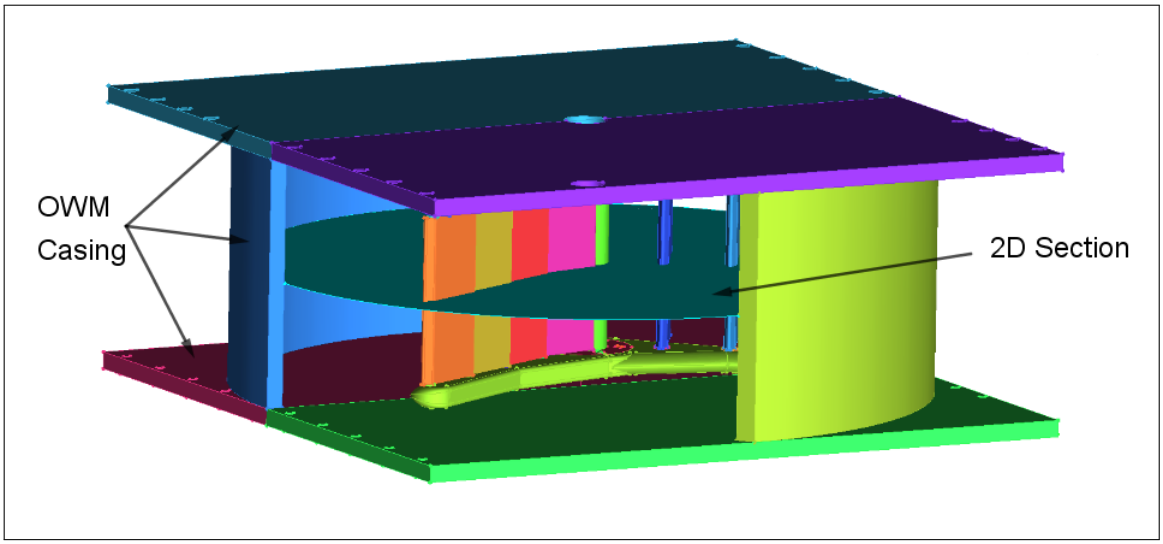


Figure 5.1: OMW Computational Model 2D Slice

With the 2D section established, the idealised geometry can be developed. The idealisation of the input geometry involves making geometry approximations that allow for the use of the desired modelling approaches without deviating too far from the original geometry. In the case of the OWM, the required geometry approximation for the 2D section comes from facilitating the use of the sliding mesh interface. The necessary geometry transformation is described in **Sec** 4.1.1. Finally, the idealised model, used as the stencil for mesh generation, and its component labeling is displayed in **Fig** 5.2. The labels starting with 'L' refer to the lamella and those starting with 'R' refer to the rotor arms.

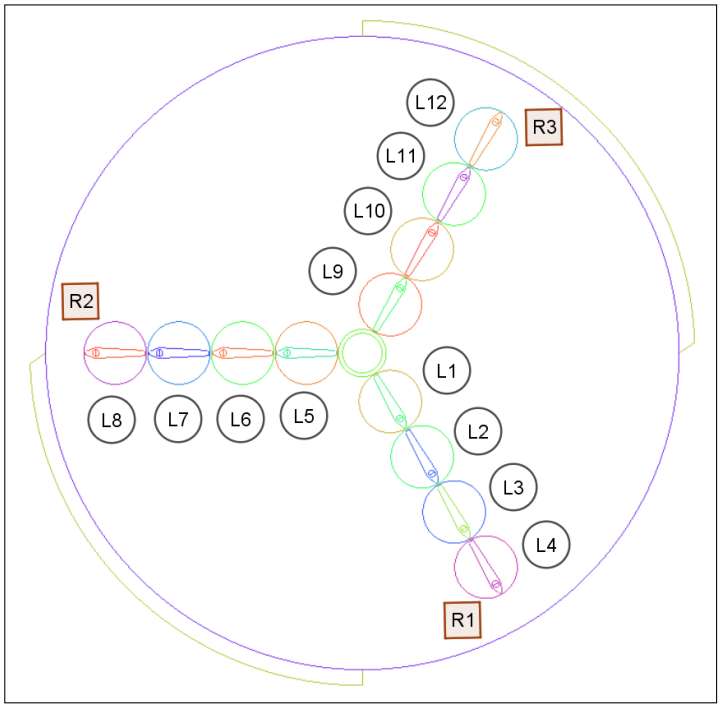


Figure 5.2: OMW 2D Computational Model Layout

Once the idealisaed geometry is finalised, the desired mesh is generated from it. The domain of the turbine 2D idealised section is discretised as quadrilateral or rectangular finite volume elements. This is performed using the blocking methodology implemented in a commercial mesh generation software known as ICEMCFD. The blocking methodology pivots about the concept of subdividing the domain into smaller mesh blocks and defining the node distributions for said blocks. Blocking is used specifically for the generation of meshes made up of quadrilateral (2D) or hexahedral (3D) finite volume elements. Further details about blocking can be referred to in [9] .The blocking defined for this work is displayed in **Fig 5.3**. The difference in the blocking structure of one of subdomains included between the rotor arms in **Fig 5.3**, is a by product of ICEM CFD's blockmeshing limitations and does not really impact the final generated mesh.

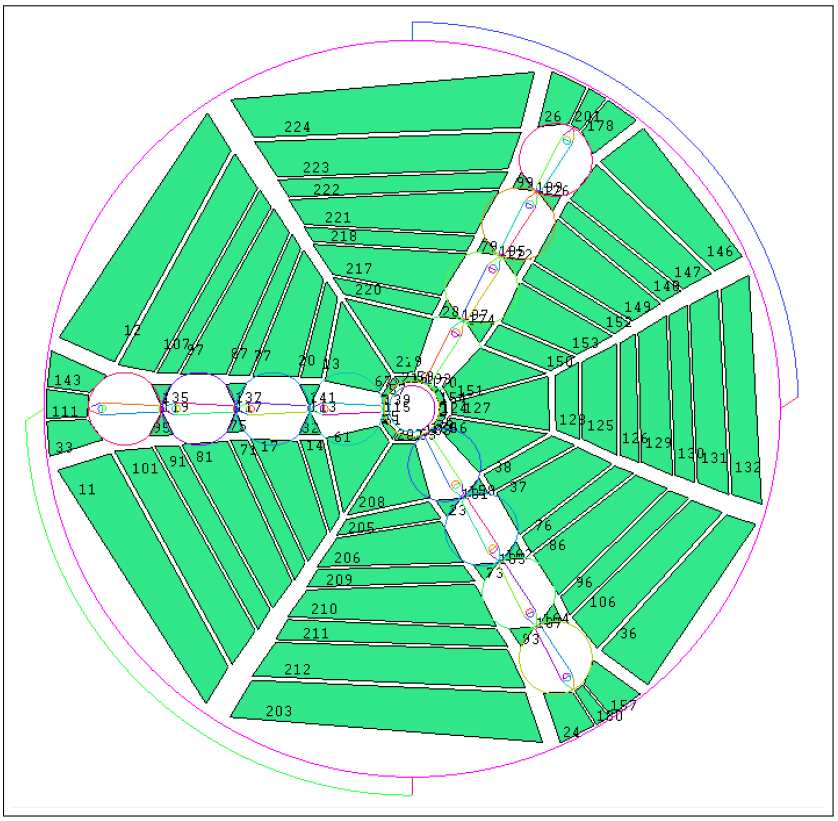


Figure 5.3: OWM Blocking Layout

This 2D mesh is then extruded by a single element thickness in the direction perpendicular to its plane, resulting in the formation of a mesh made up of hexahedral finite volume elements. This results in the final mesh being three dimensional with the flow equations not solved in the direction of extrusion. Such a mesh is known as a quasi-3D grid. For further details on discretisation for the finite volume method, see [10]. The finalised turbine mesh is displayed in **Fig 4.3b**.

This turbine mesh is then placed in a bounding box, whose boundary conditions are setup according to the diagram presented in **Fig 3.3**. The inlet boundary condition is defined at the leftmost face and the outlet boundary condition is defined at the rightmost face. The top and bottom faces of the bounding box is defined as a "slip" boundary

condition in order to not have an influence in the flow being simulated but still limit the extent of the external domain. The "no-slip" wall boundary condition is used for all the lamella and casing faces. For further details about general boundary conditions, see [6]. The edges of the bounding box are placed at such a distance that the flow dynamics near the turbine remain uninfluenced by its presence. A table summarising the mesh and geometry specifications used for this work is presented in **Tab 5.1**.

Table 5.1: Mesh and Geometry Specifics

Quasi 3D Mesh Thickness	Mesh Size (Hexahedral Elements)	Boundary Proximity to Turbine			
		Left	Right	Top	Bottom
0.005m	84,528	1.5m	2.6m	1m	1m

Once the mesh has been generated and the required boundary conditions are determined, the simulation case is setup on OpenFOAM. OpenFOAM (Open source Field Operation And Manipulation) is a C++ toolbox for the development of customized numerical solvers. Having been developed on the C++ programming language, it inherits the modular and object oriented characteristics from its programming language. Object oriented programming is built on the foundation of class definitions. The inherent characteristics of modularity and data abstraction embedded in class structures enables structured programming, utilising pre-existing libraries. For further details about developing custom solvers and functionality in OpenFOAM, see [11].

The class hierarchy of the OpenFOAM class developed in this work is displayed in **Fig 5.4**. The class combines the multibody motion handling capabilities of the *multiSolidBodyMotionFvMesh* class and the velocity Verlet based FSI capabilities of the *sixDoFRigidBodyMotion* class to form a foundation upon which the modelling approaches described in **Chapter 4** are encoded. The final developed class is named *nestedRotatingNonSmoothFSIMotion*.

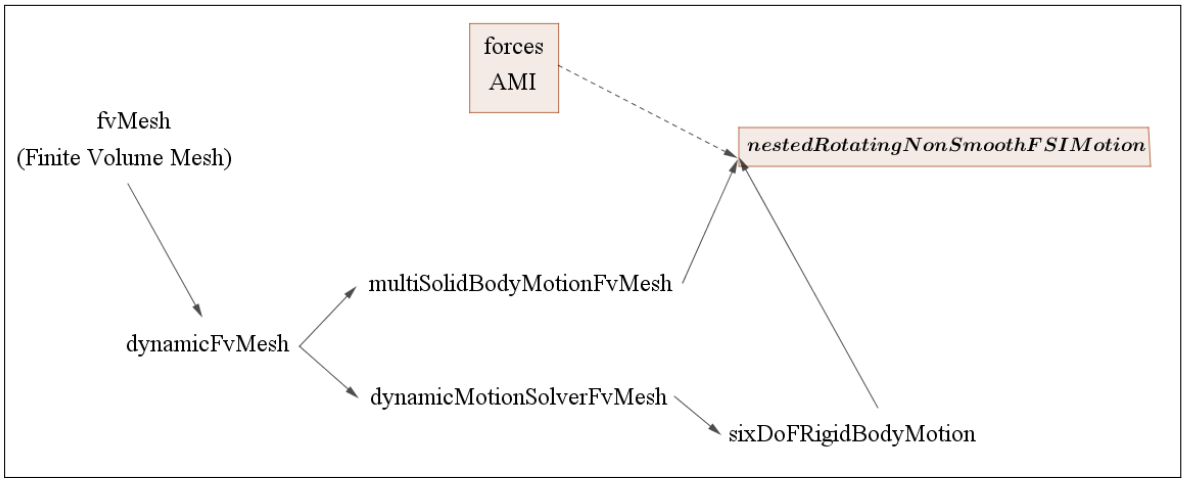


Figure 5.4: OpenFOAM Solver Class Heirarchy

The *nestedRotatingNonSmoothFSIMotion* class is utilised in the simulation case setup

of each of the twelve lamellas. The simulation is then executed through serial computation rather than parallel computation due to the fact that the developed solver is found to only function serially. However, extending the parallel capabilities of the solver is definitely possible and should be investigated further. The large set of simulation results are written into a log file which is then filtered using Linux filter scripts. The log file filtering extracts the time evolution of physical quantities related to the turbine dynamics. These filtered results are then translated into numerical matrices in MATLAB. These numerical matrices are then post processed to generate the desired predictive data. A work flow map of the entire numerical analysis process is displayed in **Fig 5.5**

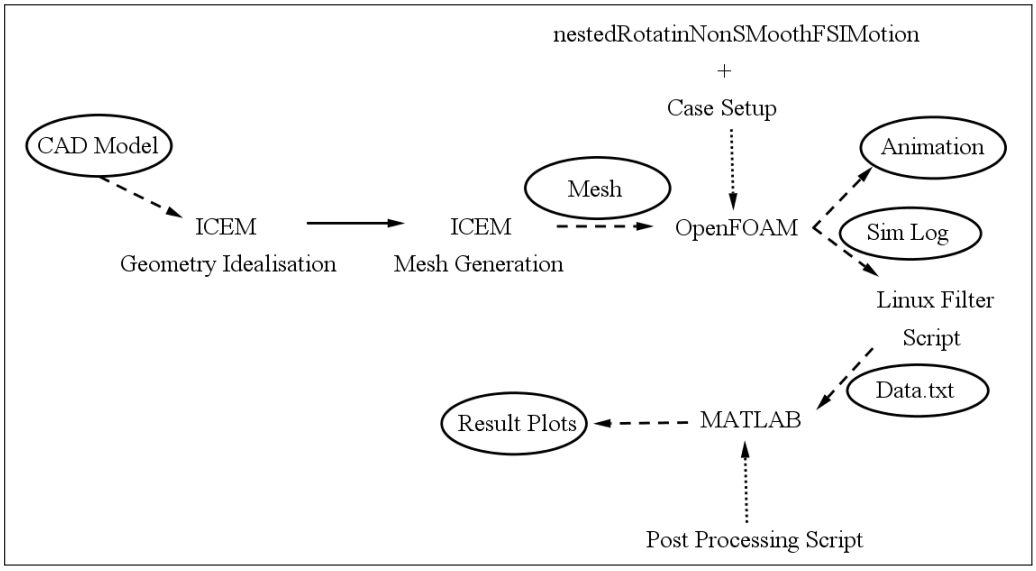


Figure 5.5: Analysis Work Flow

5.2. Model Verifications

Verification involves the task of checking whether the solver results match the analytically expected values/behaviour, derived from a theoretical understanding of the modelling. Verification of each modelling concept is considered individually and thereby results in four verifications conducted to investigate the implementation of each of the four modelling implementations.

5.2.1. Imposed Nested Solid Body Motion

The verification of nested solid body motion involves prescribing a constant angular rotation to both the outer rotational system and the nested rotational system and checking if the expected rotational dynamics are observed. A simulation case of a nested lamella within a rotational domain with three arms is setup. The outer rotational system is prescribed with an angular velocity of $\frac{\pi}{4} rad/s$ (Counter Clockwise) and the nested rotational system is prescribed with an angular velocity of $\pi rad/s$ (Counter Clockwise). A preview of the rotational systems are displayed in **Fig 5.6**.

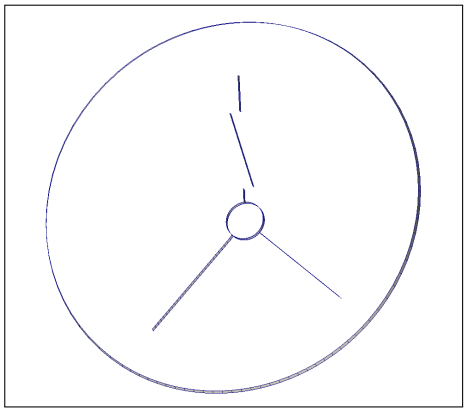
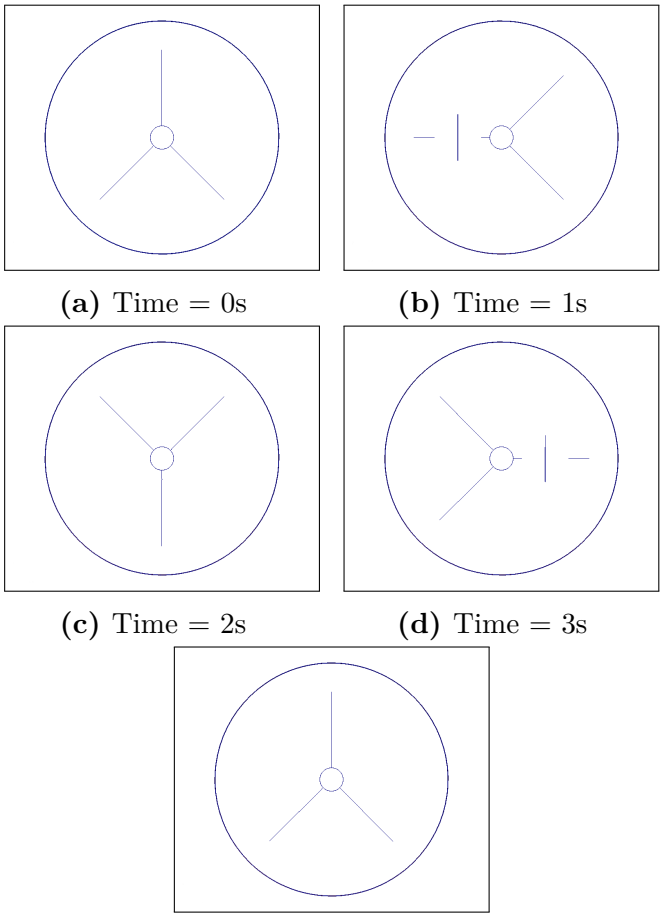


Figure 5.6: Nested Solid Body Motion Verification Setup

It is straight forward to observe from the resultant animation,**Fig 5.7**, that the outer system completed a full period rotation in four seconds and the nested system completes two period rotations in four seconds. This verifies that the prescribed speeds are correctly implemented in the modelling approach.



(a) Time = 0s **(b)** Time = 1s

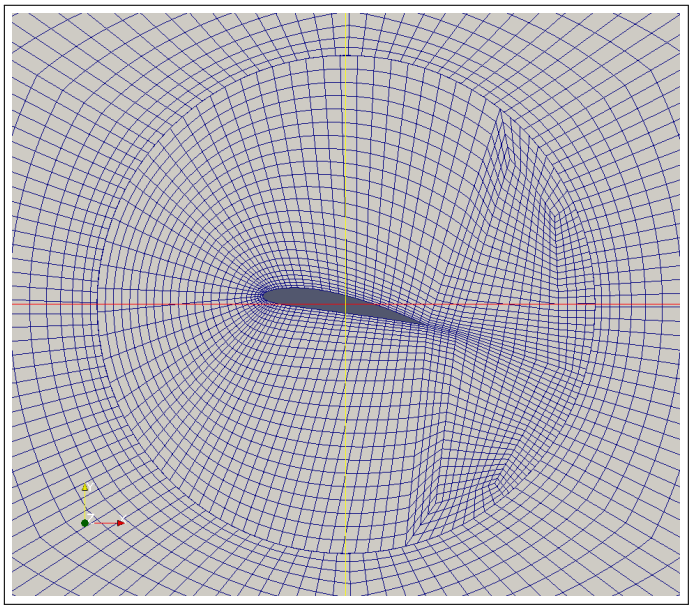
(c) Time = 2s **(d)** Time = 3s

(e) Time = 4s

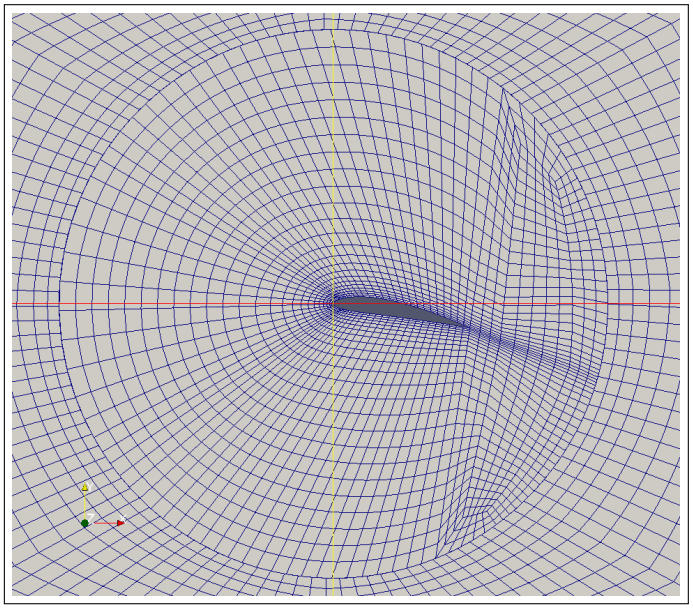
Figure 5.7: Nested Solid Body Motion Verification

5.2.2. Virtual Hinging of Center Pivoted Lamellas from Leading Edge

For the verification of this modelling approach, two rotational system simulations with the same initial angular position, the same velocity field, but differently pivoted, are setup. The system pivoted at its center is effected with the modelling approach and the evolution of their angular positions over time, is observed. Both lamellas are impinged upon by the same velocity field and rotational motion in both systems are induced. The simulation grids for both systems are displayed in **Fig 5.8**.



(a) Center Pivoted Lamella



(b) Leading Edge Pivoted Lamella

Figure 5.8: Pivot Point Shift Verification

From the simulation results, the curves representing the change in angular positions of both systems are determined. These curves are presented in **Fig 5.9**. Thus, it can be observed that both bodies undergo the same angular motion with the passage of time, even though they are differently pivoted. From these reuslts, the modelling approach is considered to be verified.

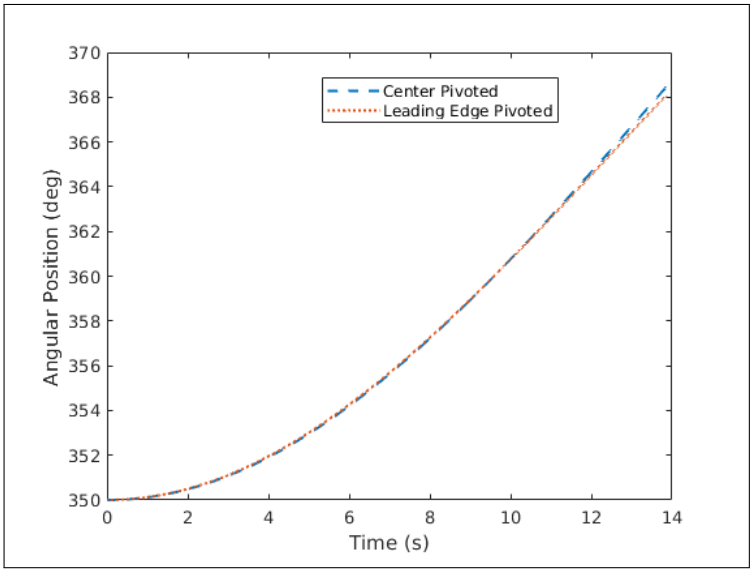


Figure 5.9: Time Evolution of Angular Position of Differently Pivoted Lamellas

The slight divergence in the angular variation of the differently hinged lamellas at a higher angle of attack is considered to be due to the difference in the fluid forcing between the two individual lamella simulations. This originates from a dissimilarity in the laminar flow solution in the presence of fluid separation.

5.2.3. Non-Smooth Mechanics

The verification process for the modelling of non-smooth mechanics involves checking whether the rotational body is constrained at the defined angular boundary positions. This verification is directly achieved from the angular behaviour of the lamella presented in **Fig 5.10**, wherein the lamella is defined to have constraints defined at the 360° and 345° angular positions while being forced by an oscillating inlet boundary condition. This verification merely shows the fact that the lamella is successfully constrained at the defined angular positions but further evaluation is required to determine its impact on the fluid solution and the accuracy of its physical phenomena capture. This is investigated in **Chapter 6**.

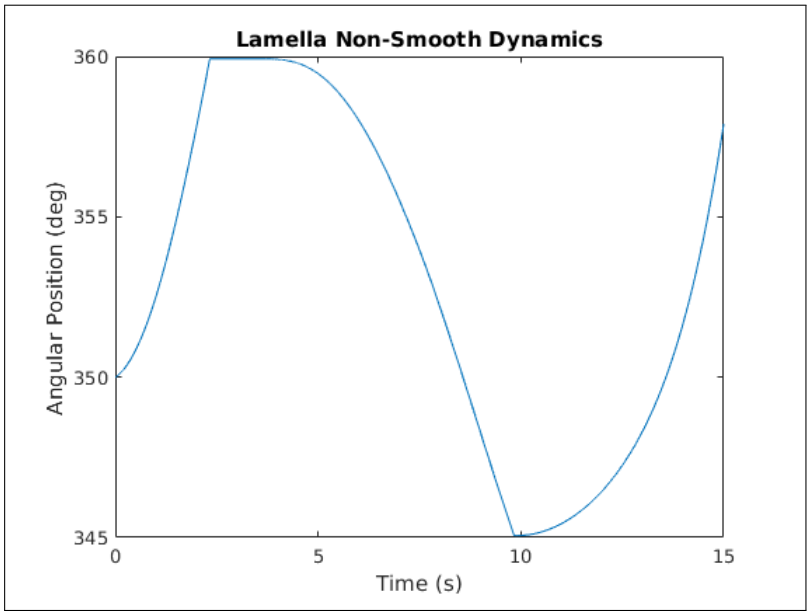


Figure 5.10: Lamella Non-Smooth Dynamics Verification

5.2.4. Turbine Torque Calculation

TTC is verified by considering a set of three lamellas placed within a constant velocity field and calculating the GTC value for the system both analytically and numerically. To this effect, the three lamella's are arranged as per **fig** 5.11. These lamella's are inclined at angle of attack equivalent to 5°.

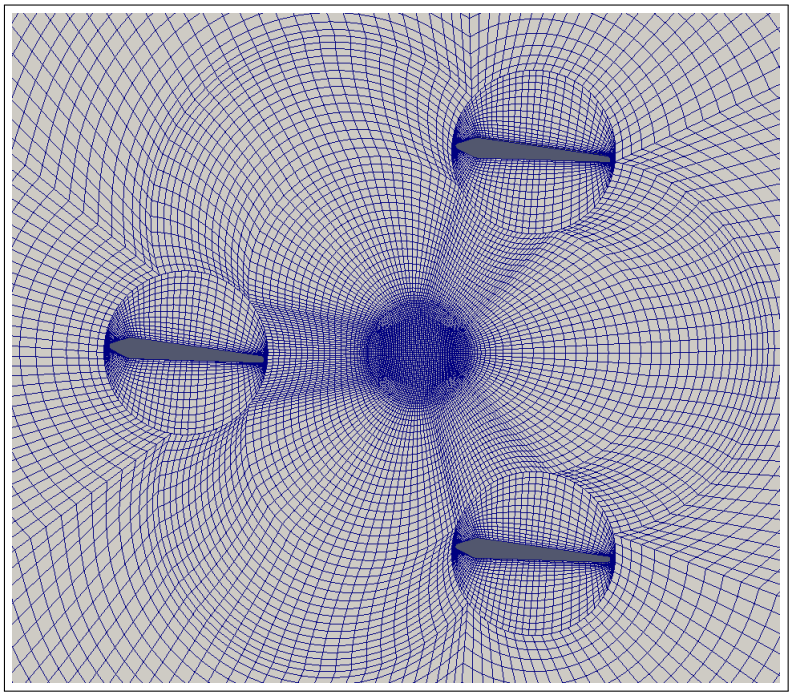


Figure 5.11: Turbine Torque Calculation Verification

The numerical simulation is performed and the TTC of the system is calculated as per the modelling approach detailed in **Sec** 4.5. The TTC value is determined to be

about $0.083Nm$ for a computational slice of thickness equivalent to $0.005m$. Scaling this TTC result to unit thickness, we have,

$$TTC_{scaled}^{numeric} = \frac{TTC_{unscaled}}{0.005} = 16.54Nm. \tag{5.1}$$

Considering the analytic solution for the following assumptions/settings and setup criteria:

- The lamellas are assumed to be thin flat plates.
- The lamellas are placed at a small angle of attack.
- The fluid is water.
- The lamella is a symmetric aerofoil and hence its center of pressure overlaps it aerodynamic center. As a result, the net force on the lamella acts at the pivot point and there exists no torsional force at the pivot point.
- The calculated net force acts purely in the vertical direction due to the assumptions of the lamella considered as a thin flat plate and its angle of attack being quite small.

Table 5.2: TTC Analytic Solution Particulars

Lamella Chord Length (c)	Lamella Angle of Attack (α_a)	Density of Water (ρ)	Free Stream Velocity of Flow (Q_∞)
$0.096m$	$0.0873rad$	$1000Kg/m\hat{}3$	$2.5m/s$

The net force acting at the pivot point of the lamella, in such a setup, is given by the following formulation found in [12],

$$F_y = \rho Q_\infty^2 \pi c \alpha_a = 170N. \tag{5.2}$$

Since the torsional force in the analytical evalution is assumed to be zero, the TTC of the system is determined by the cross product of the net force and positional vector from the system center to the lamella pivot point. Peforming the cross product calculations and summing them up for the three lamella's, the modelling approach detailed in **Section 4.5**, $TTC^{analytic} \approx 12Nm$.

The two results are found to be similar and the modelling approach is considered to be verified. The inexactness of the analytic result originates from the strong assumptions taken in calculating the net force and torque at the pivot point of the lamella's.

6

Results and Discussion

The turbine simulation results generated from the developed numerical turbine model are evaluated in this section. The simulations are setup such that the lamellas of the turbine are constrained by placing the non-smooth dynamic boundaries (**Section 4.4**) at the $0°/360°$ and $270°$ lamella local angular positions. The simulations are run up to 5 periods of rotor arm revolutions and the results from the fifth period of revolution are utilised. In each section, the observations are made just before the data is presented in the plots and the conclusions are drawn at the end of the section.

The simulation results are compared to that of the MARIN experimental results, described in **Chapter 2**, as part of the validation process. The turbine validation acts as an evaluation that provides an overall picture of the seven simulations performed for the seven considered rotational speeds of the rotor arms. Additionally, the computational time spent in generating the turbine's performance curve is presented and commented upon.

Apart from the comparison with the MARIN test results, the results are cross checked against behaviour expected from such a turbine simulation. The additional parameters evaluated for include, the periodicity of the lamella motion dynamics, the time variant continuity error of the fluid solution, the net torque time signal and the counter torque time signal of the turbine. Finally, the dynamical functioning of the turbine is visualised with the help of a few uniformly spaced snippets of the animation generated from the simulation results. With the optimal rotor arm rotational speed of the turbine being near 0.6 RPS, this rotational speed is used as the simulation instance for which these evaluations are carried out.

The average counter torque on the turbine is used to provide an insight into the influence of the lamella drag on the turbine functionality. Correlating the design of the OWM to that of a variable pitch vertical axis turbine, the chapter comes to a close with a design suggestion to further improve the turbine's torque generating capabilities.

6.1. Simulation Validation

The torque generated by the turbine at different rotor arm rotational speeds is plotted and compared to the same measured by MARIN during the experimental turbine tests. These results are presented in tabular form in **Tab 6.1**.

Table 6.1: Numerically and Experimentally Determined Torque Values

RPS of Rotor Arms (Hz)	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Numerically Determined Torque (Nm)	66.8	65	59	51.8	45.1	36.2	27.1
Experimentally Determined Torque (Nm)	42.1	37.4	32.4	27	21.3	15.4	9.5
Factor of Overprediction	1.6	1.7	1.8	1.9	2.1	2.3	2.8
Average Factor of Overprediction				2.03			

As seen from the plot in **Fig 6.1**, the torque curve is seen to be an overprediction of the true torque generation capabilities of the turbine. The torque is overpredicted by a factor of two as compared to the experimental results. It is noticeable that the results show an accurate representation of the curve trend.

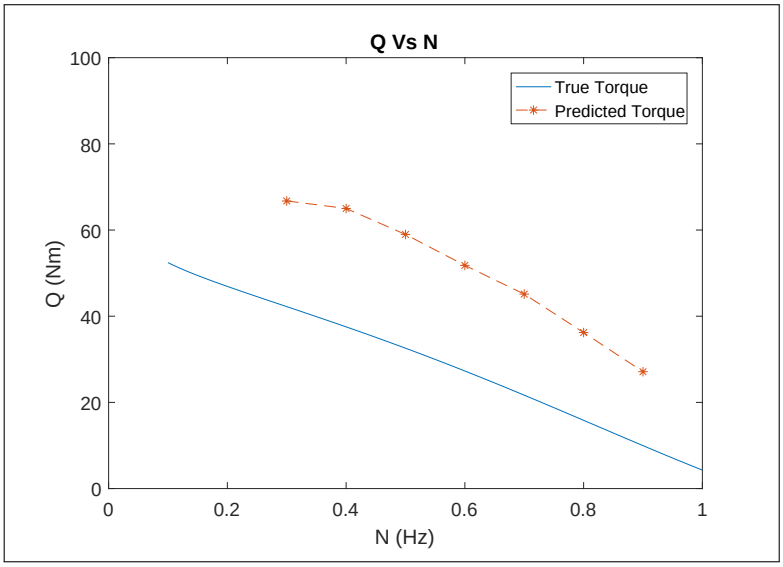


Figure 6.1: Torque Vs Rotational Rate Curve

A non-dimensional curve plot of the results in **Fig 6.1** relating the turbine's coefficient of power and tip speed ratio is considered in **Fig 6.2**. An over prediction of the turbine's optimal performance rotational speed range is observed in the simulation results. Though overpredicted, it still provides a moderately good estimation for this criteria. This range is found to be around the **0.6** to **0.65** TSR neighbourhood, as compared to **0.5** to **0.55** TSR neighbourhood seen in the experimental test results.

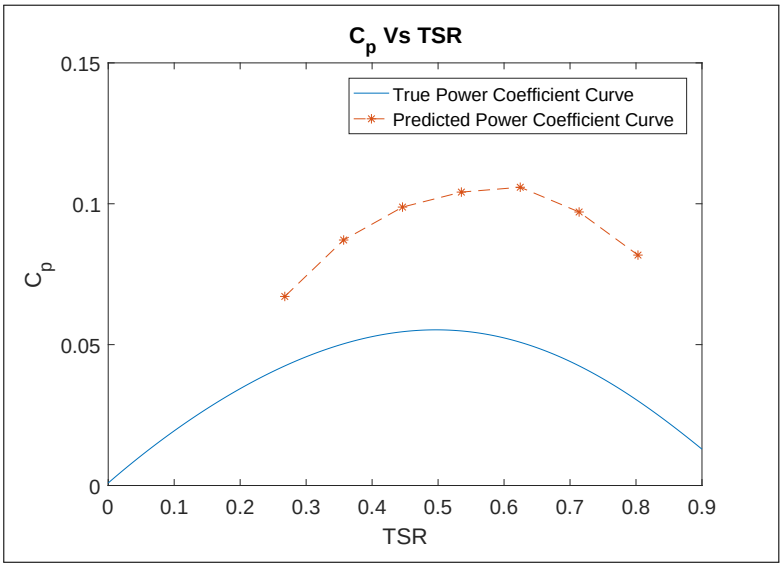


Figure 6.2: Coefficient of Power Vs Tip Speed Ratio Curve

From the overprediction seen in the validation comparisons, one could postulate that the two dimensional simulation mechanism provides an idealistic evaluation of the turbine such that all the flow entering the turbine is involved in the torque generation endeavour. A closer look at the turbine reveals gaps in the top and bottom ends along the turbine shaft. These gaps are of about $0.0225m$ in height, each and are demarcated in **Fig 6.3**.

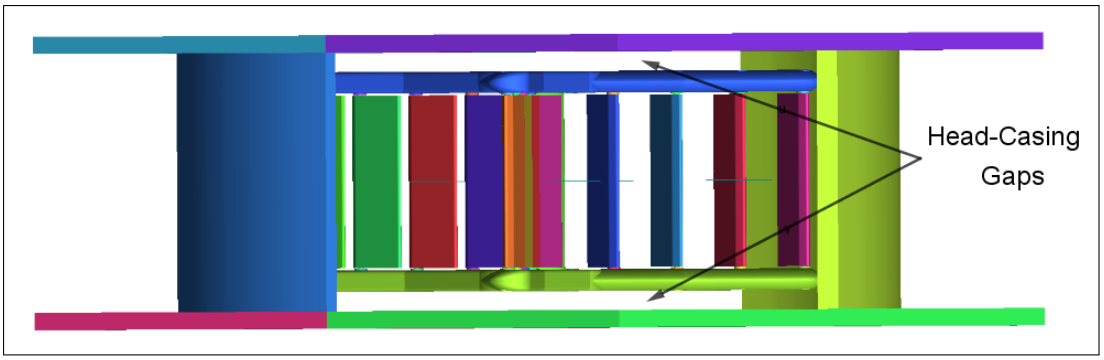


Figure 6.3: OWM Uncaptured Turbine Head-Casing Gaps

The gaps result in the leakage of the flow entering the turbine and would contribute to a diminished torque output. It is this leakage that is not accounted for in the 2D computational model and the postulated reason for the overprediction in the 2D model. Additionally, the simulation considers a laminar flow model for a flow known to be turbulent. This lack of a turbulence model in the fluid modelling is believed to impact the drag prediction in the simulation. This could lead to an underprediction of the drag effects and thereby an overprediction of the torque generation capabilities of the turbine. Considering the two plausible explanations for the overpediction, the lack of a 3D model stands out the most, as a flow leakage through a total gap of 0.045m (15% of the turbine height) is on the higher side of gaps expected within a turbine housing. The 2D result could be translated into an approximation of the 3D scenario by a scaling

that accounts for the additional gap in the vertical direction. This scaling can be based upon the following proportionality relation between torque and velocity, that originates from Bernouli's equation for incompressible flow:

$$\tau \propto \mathbf{F} \propto p \propto \mathbf{v}^2, \tag{6.1}$$

where, τ is torque, p is pressure and $\mathbf{v}$ is velocity. Such a means of scaling could be investigated in order to derive a suitable means of translating the 2D result into 3D and would be a computationally efficient means of addressing the overprediction in torque. It is understandable that one would require data from the 3D simulation to consolidate the scaling means and thereby further motivates the need to extend the simulation model to a 3D solution.

If one was to consider turbulence modelling, a "low Reynold's number formulation turbulence model" would be required in order to derive a meaningful realisation of the turbulence. Each lamella moves through a range of angular positions, from being perpendicular to flow to being aligned to flow. Since fluid separation occurs for large angles of attack, separation is a mojor fluid event in the turbine dynamics. It is the presence of separation that motivates the need for a full resolution approach over a wall function approach, with regard to the turbulence modelling. Considering the following formulation for turbulent boundary layer thickness for a flat plate,

$$\delta \approx \frac{0.37L}{Re^{\frac{1}{5}}}, \tag{6.2}$$

the expected boundary layer thickness for this simulation case can be estimated. As per the Reynold's number calculations from **Section 2.2**,

$$\delta \approx \frac{0.37 \times 0.1}{10^{5 \times \frac{1}{5}}} \approx 3.7 \times 10^{-3} m. \tag{6.3}$$

Thus to model turbulence, a considerably high resolution boundary layer mesh would be required, which in turn would demand the need for parallelisation to make the computational efforts viable. The fact that the solver runs in only serial becomes a limitation which restricts the fluid modelling to a laminar flow solution. On the contrary, a laminar flow solution forms a fundamental start point to establish numerical results for this study, encourages its use in this work

The perceived difference between the physical (3D) and computational (2D) models can be visualised as presented in **Fig 6.4**.

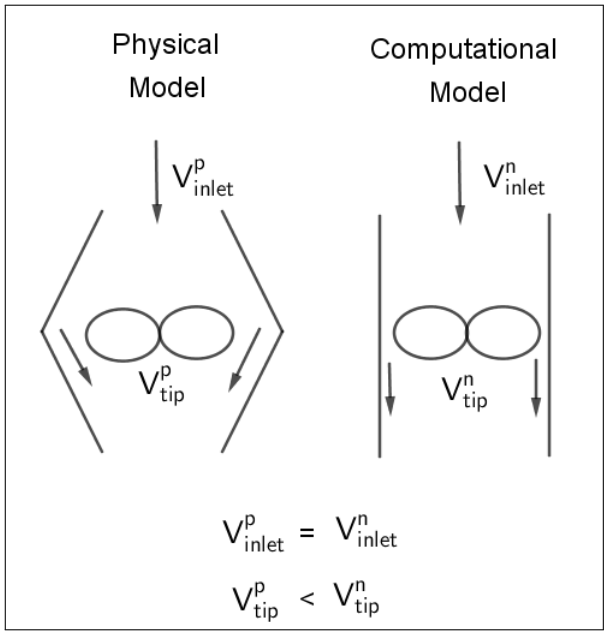


Figure 6.4: Physical Model Vs Computational Model

The velocity of flow within the casing is different for the two cases based on the principle of mass continuity. From this, the shift in the predicted optimal performance range as compared to that of the 3D experimental results are justified to the extent that they are not expected to be the same. On this note, a three dimensional numerical analysis of the turbine performance would be an important investigation to derive further understanding of the simulation capabilities of the developed solver.

The computational time involved in generating the predictive performance curve is the time required to perform the seven simulations that determine each point of the performance curve. For a 2D simulation of about 85,000 hexahedral elements (**Tab 5.1**), it takes about 20mins of computational time to generate results for a simulation time of 0.05s . The table presented in **Tab 6.2** enumerates the computational time for each of the seven simulation instances.

Table 6.2: Computational Time Estimate

RPS (Hz)	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Simulation Time [5 Periods] (sec)	16.67	12.5	10	8.33	7.13	6.25	5.56
Computational Time [5 Periods] (day)	4.7	3.5	2.8	2.3	2	1.8	1.6

The developed solution mechanism functions in serial, thereby requiring a single computational core to run a single simulation instance. Eventhough the simulation instance is run in serial, multiple simulation instances can be run simultaneously. Thus, with seven available cores, one could numerically estimate the turbine performance curve of this study, in about five days. Distribution of about 60,000 volume elements per pro-

cessor establishes an approximate optimal element distribution for parallel processing (specific to OpenFOAM). Considering the number of elements in this 2D study, parallelisation is considered to make a minimal difference in the computational time. However, the computational time for the current simulation instances are strongly dependent on the time step size. For a set of 3D simulation instances, both parallelisation and optimal time stepping will have a significant impact on the computational times.

6.2. Lamella Periodicity

The simulation results for a particular prescribed rotor arm rotational speed is expected to converge to a state of periodic behaviour in the lamella dynamics, if there exists no variation in in the inlet condition to the turbine. Thus, a simulation case is run up to a number of turbine rotational periods to arrive at a more accurate overall prediction. This establishes the lamella periodicity as an important factor to determine the appropriateness of the simulation's transient behaviour within a particular period.

The periodicity of the movement of the lamellas is evaluated by two means. Firstly, a lamella's periodicity with itself between two neighbouring periods constitutes an inter period evaluation. The lamella's periodicity with its sister lamellas (equivalent radial position) on the other two arms, within the same turbine arc region, constitutes its intra period evaluation. The third set of radially distant lamella's (from the turbine shaft), namely L3, L7 and L11 (**Fig 5.2**) for the 0.6 RPS case, are considered for these evaluations.

The intra period lamella angular dynamics within the fifth period are seen to show a lack of overlap, as presented in **Fig 6.5**.

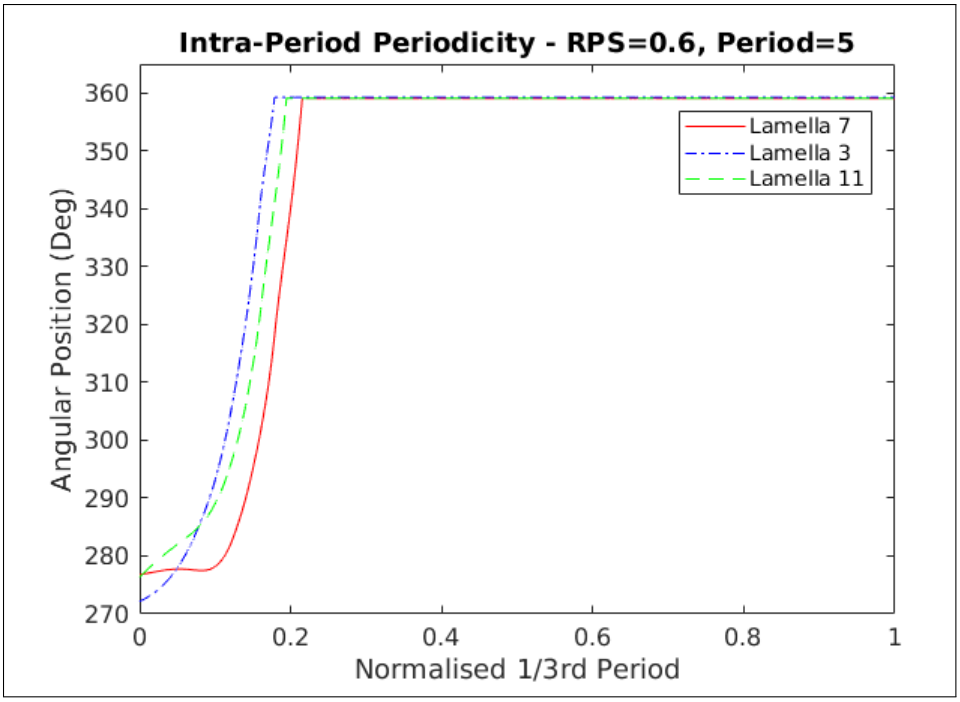


Figure 6.5: Intra-Period Periodicity

It is observed in **Fig 6.6**, that the considered lamellas show a lack of overlap in their inter-period angular dynamics between periods 4 and 5.

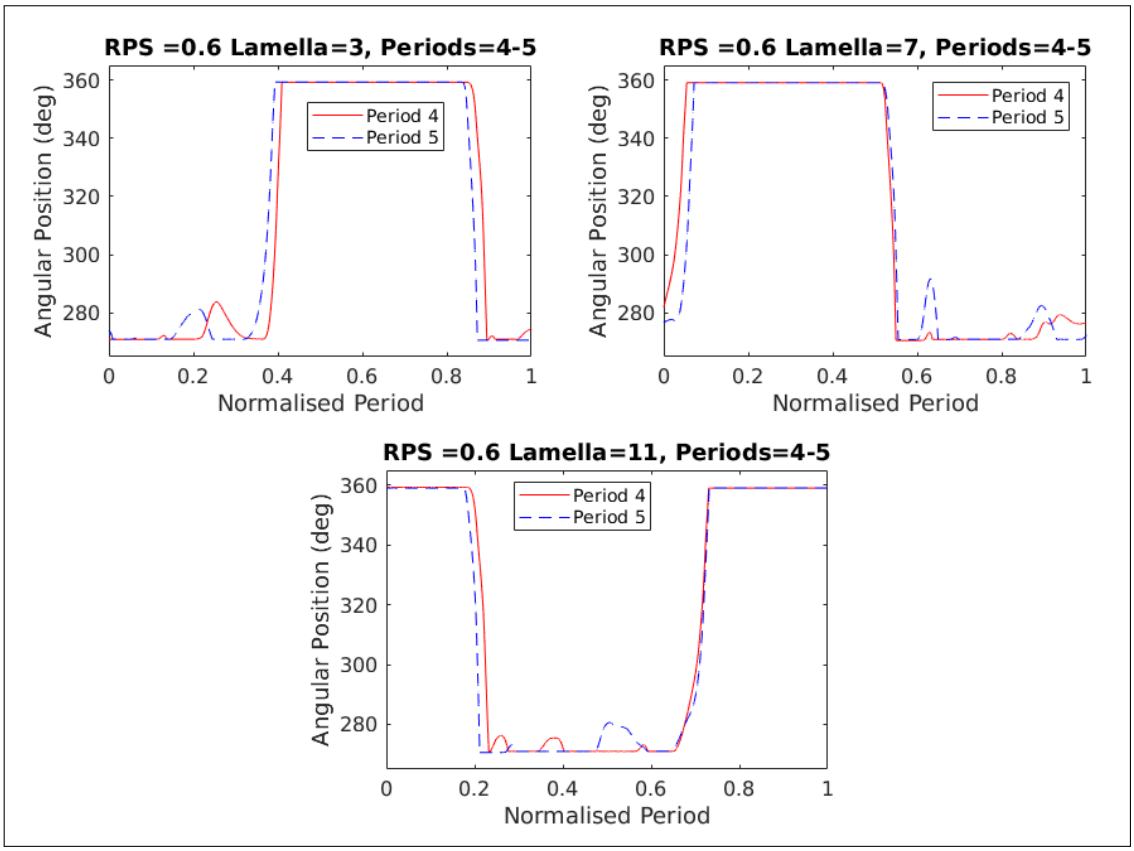


Figure 6.6: Inter-Period Periodicity

The lack of sufficient periodicity in the lamella dynamics points to the fact that the turbine dynamics have not been adequately converged. The fact that the turbine dynamics have not converged is considered to be due to the lamellas being sensitive to the fluid forcing, wherein, the sensitivity is most pronounced in the opening and closing phase. Adding a physical damping term to the lamella motion dynamics is hypothesised to reduce its sensitivity and promote convergence to a better periodic behaviour. The additional damping would also induce larger allowable time steps for a particular choice of Courant number and thereby help reduce the computational time, while ensuring the accuracy of torque prediction is unaffected.

6.3. Continuity Error

The non zero value up to which the divergence of the velocity (**Eq 3.3b**) is reduced to, within a finite volume cell during a simulation is known as local continuity error. The summation of the local continuity error in every cell within the considered domain is referred to as global continuity error. Finally, the summation of global continuity errors at every time step over the simulation period, forms cumulative continuity error.

The cumulative continuity error provides a measure of the amount of mass gained or lost during the propagation of the simulation over time and is required to be kept below

a certain threshold, preferably within an order of 10^{-8}, to maintain the integrity of the simulation.

For an $RPS = 0.6$ simulation, five rotational periods involves running the simulation up to about 8.33 seconds. It can be seen in **Fig 6.7**, that the cumulative continuity error sees a non-monotone negative accumulation up to an order of 10^{-8}.

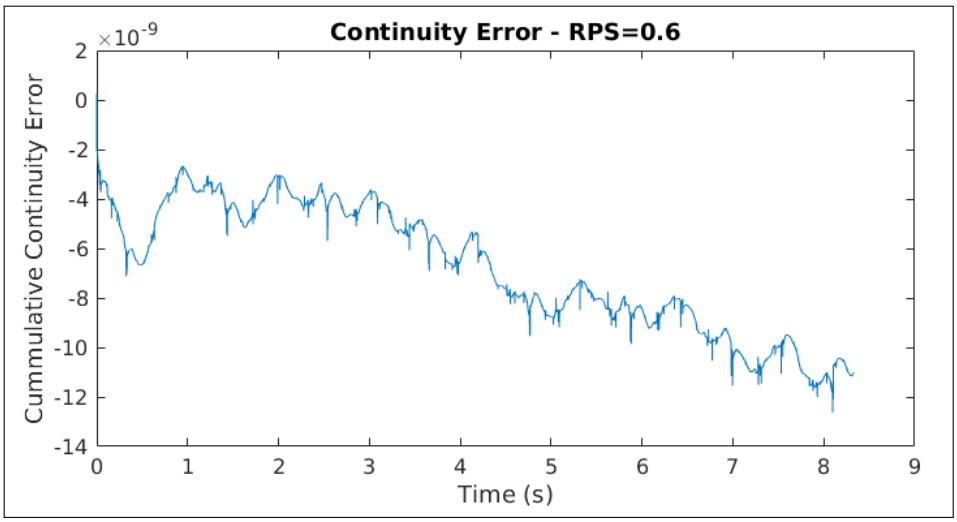


Figure 6.7: Cumulative Continuity Error Over Simulation Time

The fact that the cumulative continuity error exists at the near boundary of the allowable threshold, points to a lack of accuracy in the flow solution. This reduces the credibility of the fluid solution and further investigation into the diminished accuracy of the transient fluid solution is undertaken in **Section 6.4**.

6.4. Time Signal Analysis of Net Torque and Drag Induced Counter Torque

The representation of the torque is established as follows:

- The OWM rotates clockwise during its operation and thereby, the sign of the generated torque is negative and the drag induced counter torque is positive, as per the right hand thumb rule.
- In order to make the torque time signal more intuitive, the signal is represented as a positive entity when the turbine is generating power and the drag based counter torque that acts against the power generating torque is represented as a negative entity.
- The torque/counter torque time signal from the simulation with the 2D computational grid is with respect to the single element thickness of $0.005m$. This signal is then scaled up to torque per unit length.

As discussed in **Section 4.5**, the net torque generated by the turbine is computed by a summation of the twelve torque contributions from each lamella at every time step.

The collection of the GTC values at each time step forms the net torque time signal of the turbine and allows for the evaluation of the turbine TTC variation with time. The drag induced counter torque is computed by the summation of the negative (intuitive sign convention) torque contributions at each time step. In **Fig 6.10** and **Fig 6.12**, the opening and closing intervals of the lamella sets in each rotor arm are designated as follows: The closing regions are labeled by **(a)** , **(c)** and **(f)** for rotor arms 2, 1 and 3, respectively. The opening regions are labeled by **(b)** , **(d)** and **(e)** for rotor arms 2, 1 and 3, respectively.

6.4.1. Time Signal Analysis : Net Torque

For a power generating turbine, the net torque time signal is expected to be positive and more or less constant over time. The net torque time signal is also correlated with the change in the cumulative continuity error and the angular dynamics of the lamellas.

The net torque time signal of the fifth period is displayed in **Fig 6.8**. Positive and negative torque spikes are observed in the time signal. These spikes are understood to be numerical artifacts which result out of a time local instability of the fluid solution. In order to filter out the unphysical spikes from the time signal, the spikes are truncated. The truncated torque signal is found to be positive. By averaging out the time signal over the considered fifth period, a period averaged net torque per unit length, of about $260N$ is estimated for a turbine rotational speed of $0.6Hz$.

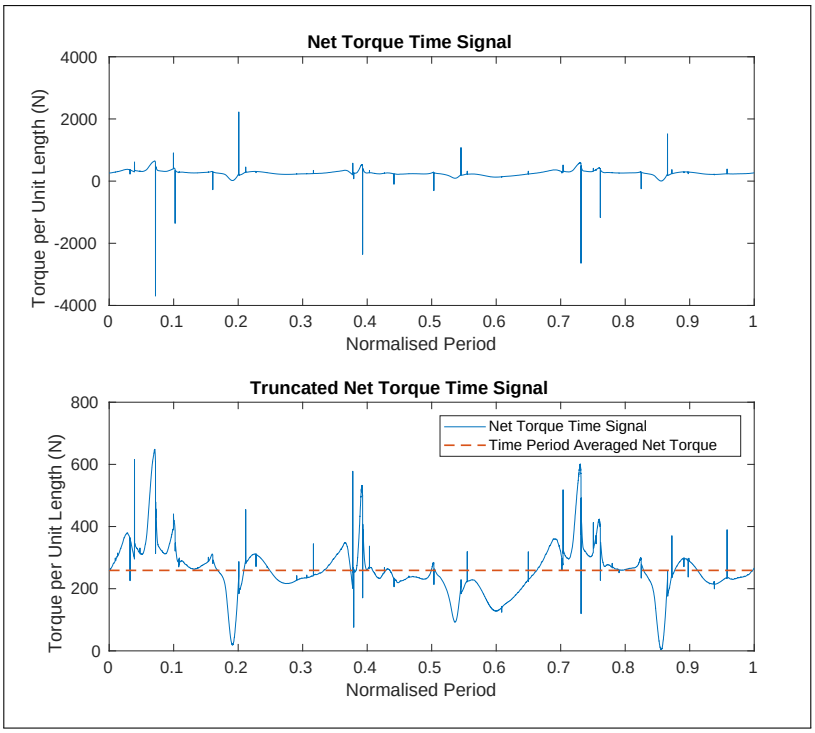


Figure 6.8: Net Torque Time Signal

The correlation of the net torque time signal with the cumulative continuity error in the the fifth period of this case is shown in **Fig 6.9**. The larger spikes in the continuity error are found to be corresponding to the spikes in the torque signal.

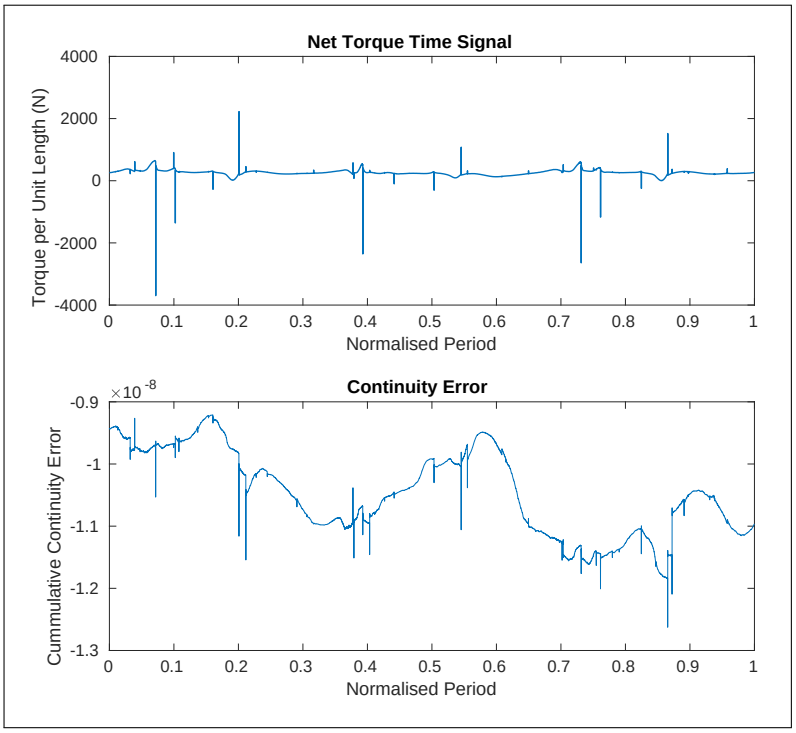


Figure 6.9: Net Torque Time Signal Correlated with Cumulative Continuity Error

The correlation of the net torque time signal with the lamella angular dynamics in the fifth period is plotted on **Fig 6.10**. It shows that in general, the torque spikes occur at the instance of the lamella closing.

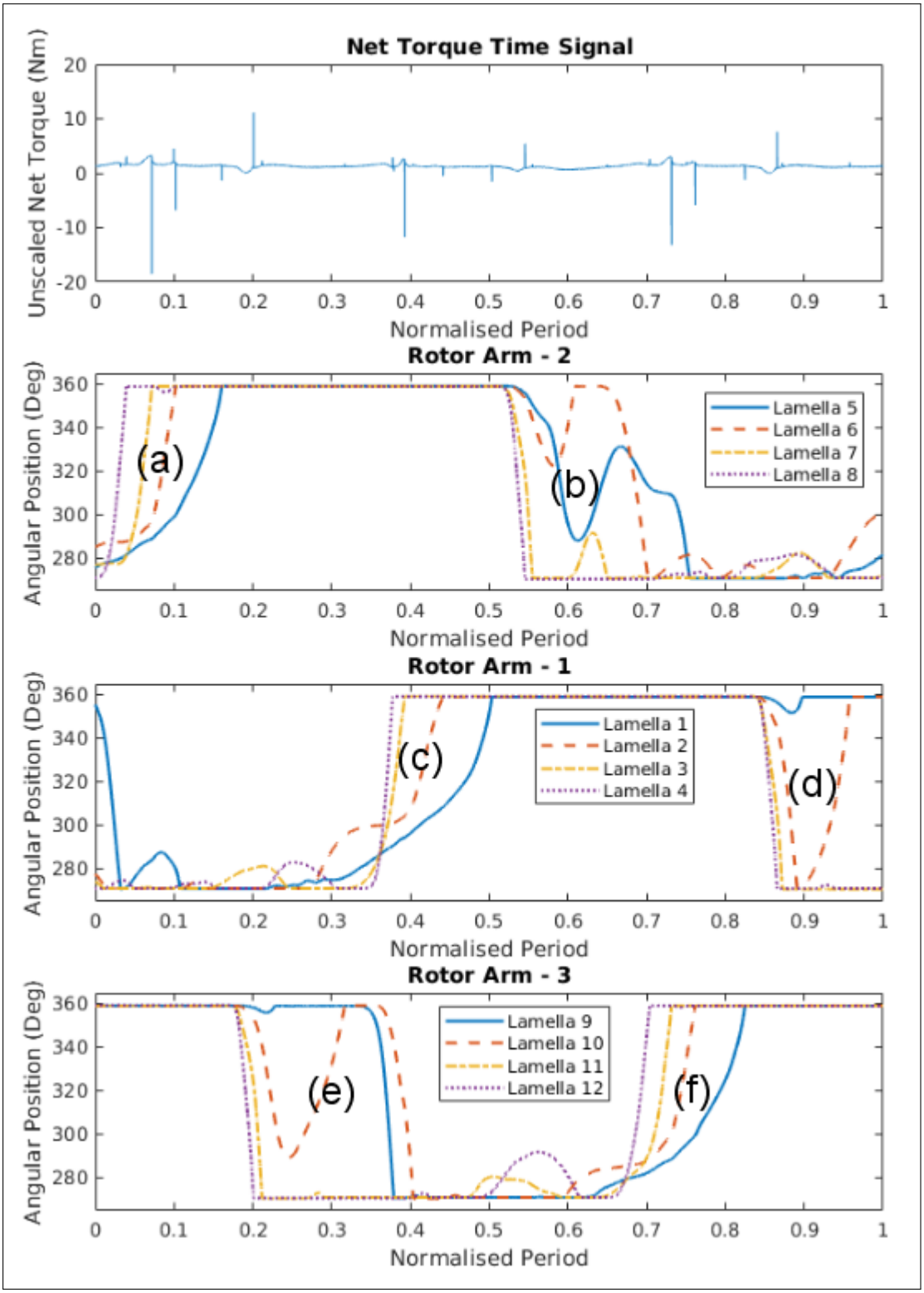


Figure 6.10: Net Torque Time Signal Correlated with Lamella Angular Dynamics

The net torque time signal spikes being correlated to the instance at which the lamellas close, leads to the conclusion that the implementation of the non-smooth dynamics results in the destabilisation of the fluid solver. The event of the lamella being instantaneously decelerated by the perfect momentum sink at the proximity region is postulated to be a far too simplistic means of dealing with the non-smooth dynamics of the lamellas. The spikes seen in the cumulative continuity error coinciding with the torque signal spikes, leads to further substantiation that the fluid solver is destabilised by the current means of modelling the non-smooth dynamics of the device. The use of a more gradual deceleration mechanism and an inner coupling loop between the fluid and structural solver are probable means of addressing the matter.

6.4.2. Time Signal Analysis : Drag Induced Counter Torque

This counter torque time signal provides an insight in the drag related losses incurred by the turbine during its operation. The drag induced counter torque time signal is correlated with the change in the angular dynamics of the lamellas.

The counter torque time signal of the fifth period is displayed in **Fig 6.11**. Similar to the net torque time signal, the counter torque time signal spikes are truncated and plotted to form a more physically appropriate time signal. By averaging out the time signal over the considered fifth period, a period averaged counter torque per unit length, of about -40 is estimated for a turbine rotational speed of $0.6Hz$.

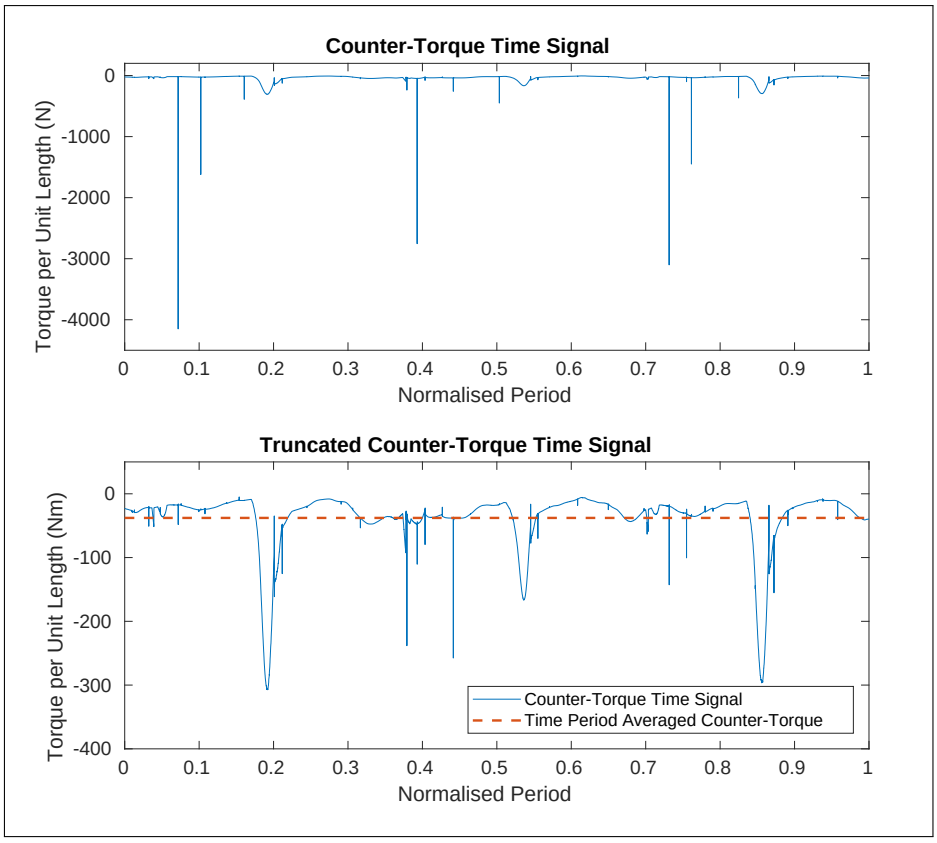


Figure 6.11: Counter Torque Time Signal Analysis

The correlation of the truncated net torque time signal with the lamella angular dynamics in the the fifth period is shown in **Fig 6.12**. The negative rise in counter torque is found to occur over the opening phase of the lamellas in a rotor arm and goes back down as soon as the lamellas reach the fully open position.

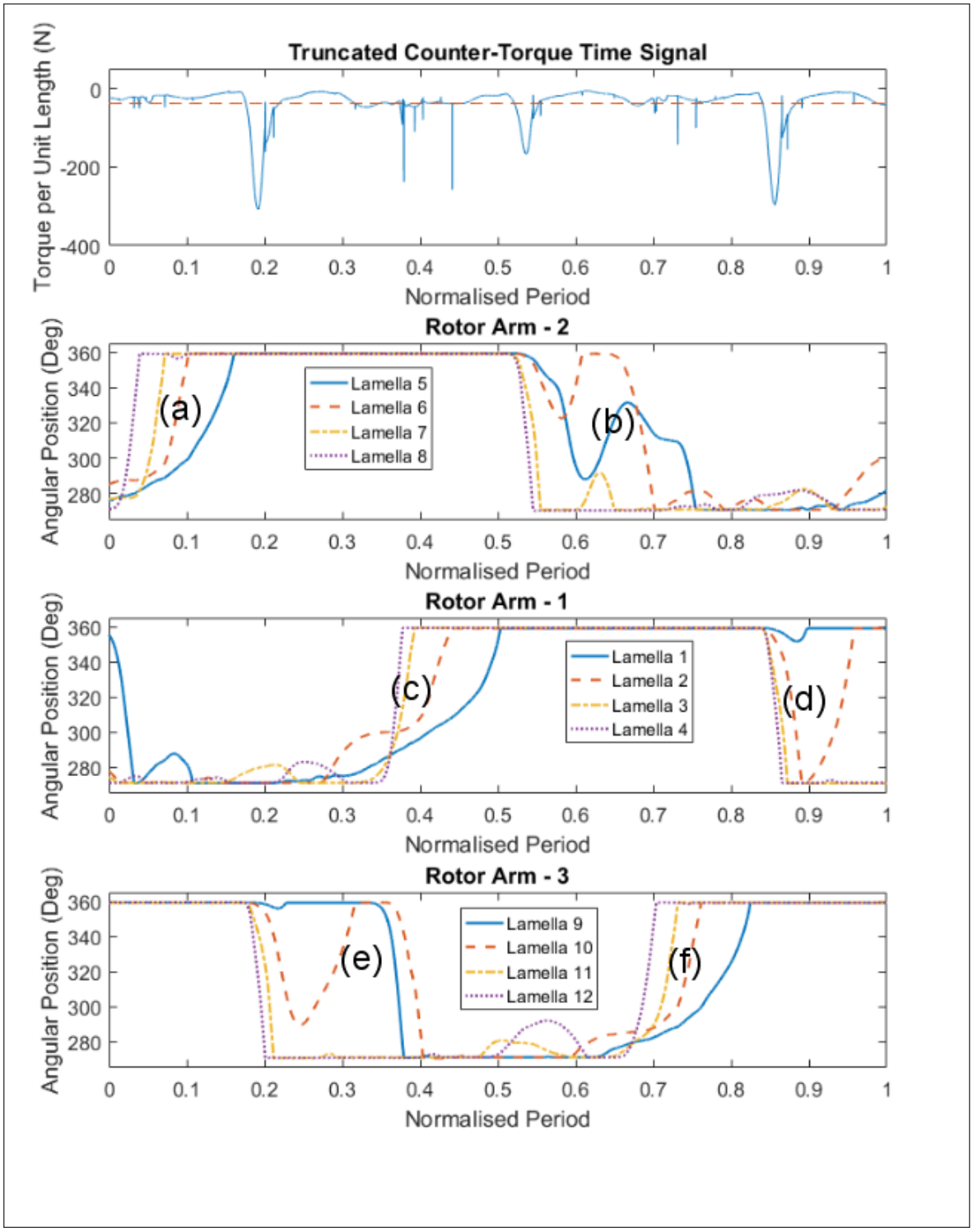


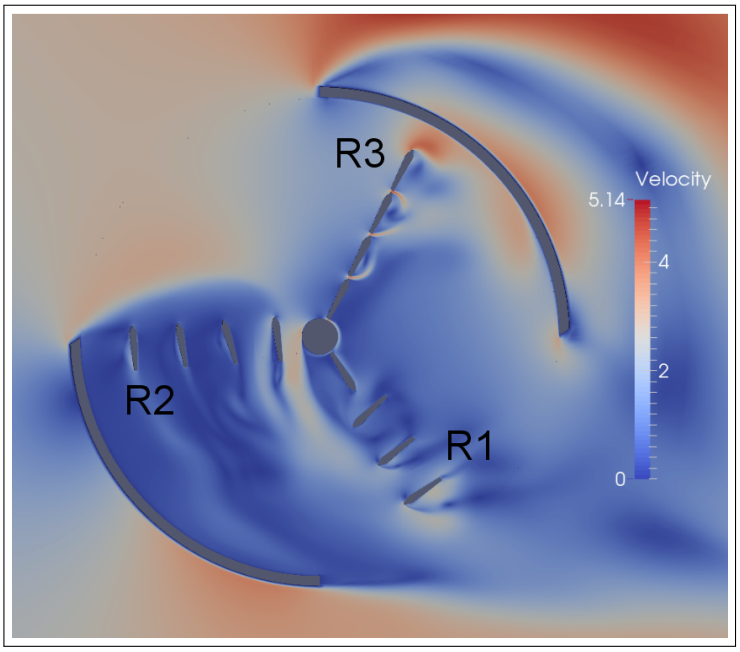
Figure 6.12: Counter Torque Time Signal Correlated with Lamella Angular Dynamics

The counter torque behaviour during the opening phase of the lamellas reflects the expected counter torque behaviour as the lamellas induce the most drag while they are transitioning from the closed position(generating torque) to the fully open position (aligned to flow) and provides credibility to the modelling of the lamella opening dynamics.

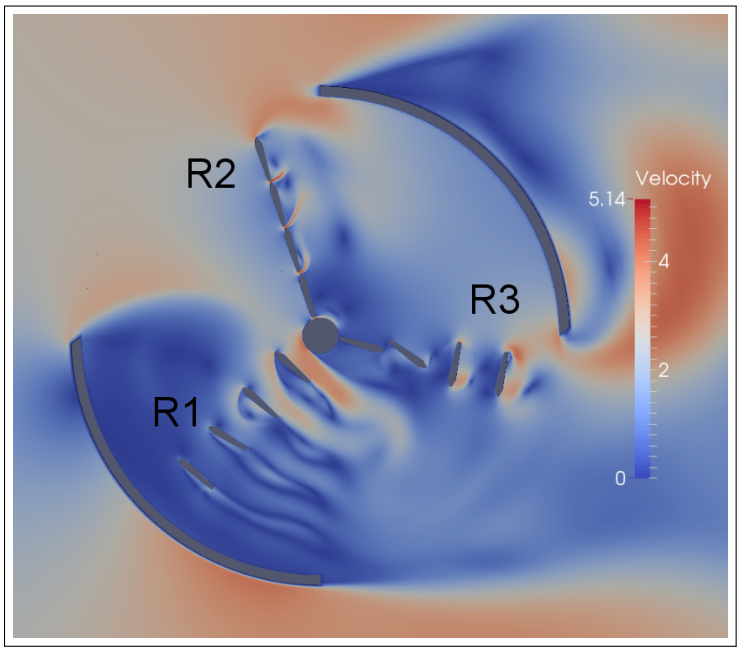
6.5. Animation - Transient Velocity Contour Plots

The turbine animation depicting the dynamic functionality of the OWM is represented by a collection of six snippets at uniform time intervals along the considered period. The need for these gaps are described in **Section** 4.1.1. Rotor arm labels are placed in each animation snippet for better clarity.

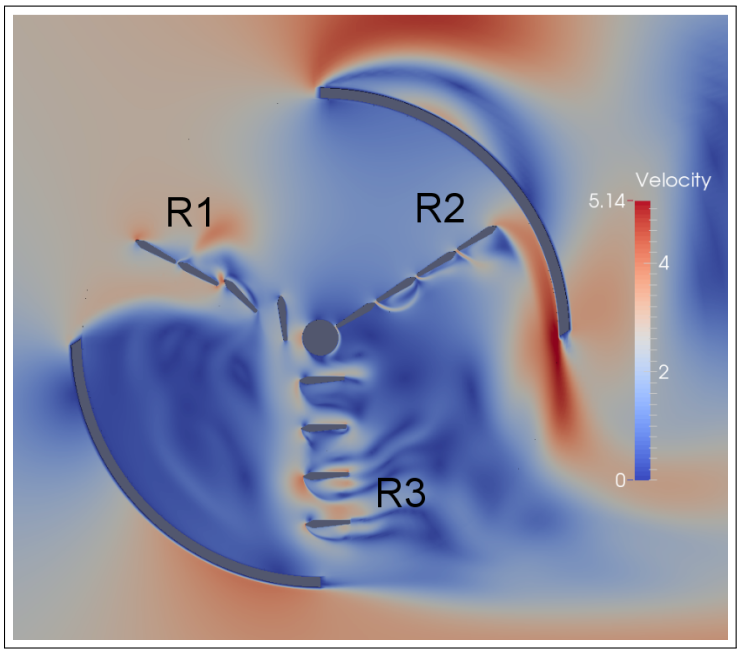
The collection of animation snippets of the fifth period of the simulation is displayed in **Fig** 6.13. The fully open lamellas being constrained to the 270° lamella-local angular position are observed. The lack of a turbulence model for the fluid solution can be seen by the fact that the velocity contours appear sharp and unmixed. Additionally, the small gaps between the lamellas of an arm are observed to be causing a section of relatively higher velocity flow at the gap sections over the span of the rotor arm.



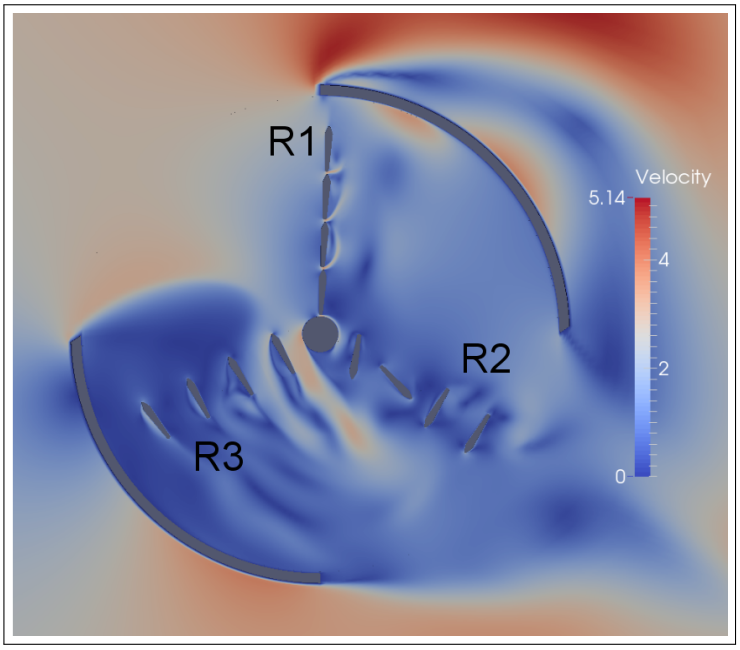
(a) Velocity Contour Plot at $t = 6.65s$



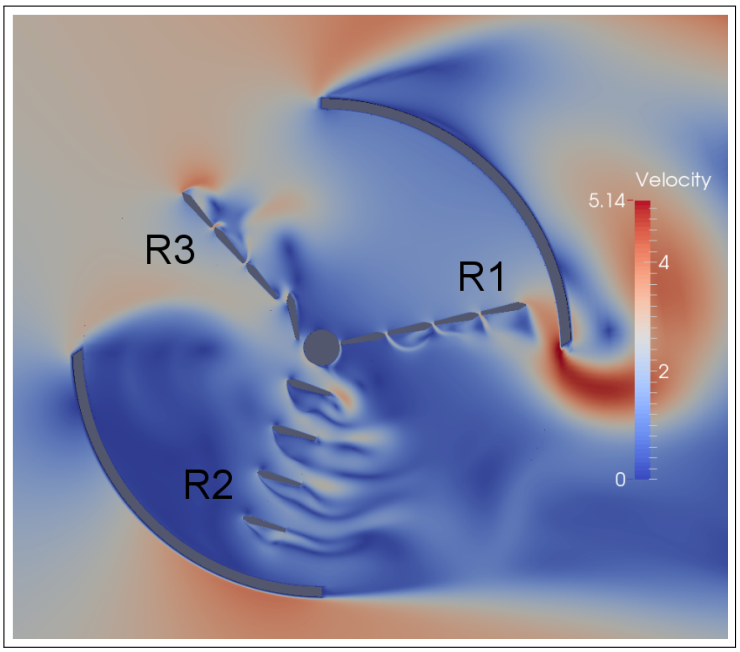
(b) Velocity Contour Plot at $t = 7s$



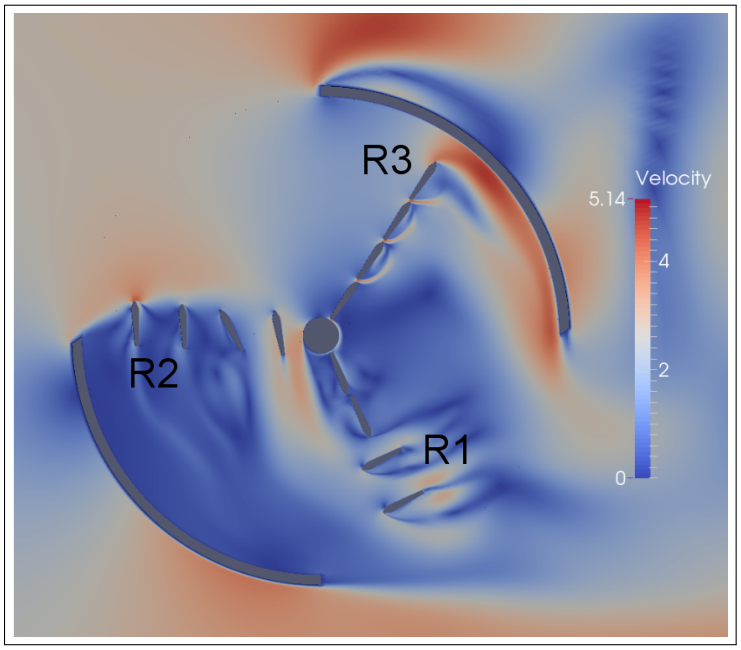
(c) Velocity Contour Plot at $t = 7.35s$



(d) Velocity Contour Plot at $t = 7.65s$



(e) Velocity Contour Plot at $t = 8s$



(f) Velocity Contour Plot at $t = 8.35s$

Figure 6.13: Strongly Constrained Case : 5^{th} Period Turbine Simulation for $RPS = 0.6$

The results seen above points to a need for investigation into suppressing the lamella gaps and using a turbulence model for the fluid solution in order to approach a better representation of the flow dynamics involved in the actual operation of the turbine.

6.6. Design Suggestion : Additional Torque with Lift Induced by Lamella Pitch Control

For a rotational speed of $0.6RPS$, the counter torque is seen to diminish the average net torque per unit length from $300N$ to about $260N$. Thus, the drag induced counter torque reduces the torque output of the turbine by about 13.5%.

If the OWM was to generate more torque and the drag induced counter torque remained the same, the impact of the counter torque on the overall torque generation of the turbine could be reduced. Considering the case of a variable pitch vertical axis turbine, the torque generated by the turbine is improved by controlling the pitch of the lamellas in such a way that an additional lift force is used to increase the turbine's torque generation. Such an approach would require a well devised lamella pitch control mechanism to artificially orient the lamellas such that they produce lift while in unconstrained motion. Thus, the lamellas would be contibuting to torque generation while fully closed (perpendicular to the inlet flow) and while open and optimally angled to flow. Thus, each lamella is effectivelly contributing positively to torque generation at all times. This would be a probable design aspect to consider in improving the torque generation capabilities of the device.

7

Conclusions and Recommendations

Based on the results and discussion of **Chapter 6**, final conclusions are drawn and the merits of the conducted research is outlined. Finally, the thesis comes to a close with recommendations for further investigations into improving the solver and putting forward a means of performing a numerical study on the impact of the turbine upon the flow and morphology of the water body it is placed in.

7.1. Conclusions

A summary of the conclusions arrived at over the course of this work are as follows:

- From the C_p vs TSR validation comparison, the torque overprediction is hypothesised to be a byproduct of not accounting for the flow gaps in the axial direction of the turbine due to the two dimensional cross sectional nature of the computational domain. The fact that there is an observable shift in the optimal turbine performance range predicted by the 2D model as compared to that of the 3D experimental results, is presumed to be a consequence of the difference in intra casing velocities between the two models.

- A turbulence model with a "low Reynold's number formulation" is required in order to simulate the turbulent flow dynamics within the turbine casing. The presence of separation in the turbine dynamics motivates the use of such a turbulence model.

- The lack of periodicity in the lamella dynamics relates to the lack of convergence in its periodic physical behaviour. This is postulated to be due to the sensitivity of the lamella to fluid forcing. The inclusion of a physical damper to the lamella model formulation would aid in the cause and additionally induce larger time steps for a particular choice of Courant number. The larger time steps would be a result of the lamella motion being more gradual.

- The validity of the fluid solution is diminished by the cumulative continuity error remaining near the allowable threshold for this error. The cumulative continuity error spikes coinciding with the torque time signal spikes, point to transient instabilities in the time variant fluid solution. These instabilities are undesirable in a transient solution and could end the simulation prematurely. The spikes are

found to occur at the lamella closing instances and thus give evidence that they are a result of not modelling the lamella closing event more gradually. Incorporating gradual contact dynamics into the lamella motion model and the addition of fluid-structure coupling iterations in the structural model is postulated to minimise instabilities in the fluid solution, thereby improving the accuracy of the fluid solution and allow for the use of larger Courant numbers which would make larger time steps viable.

- The drag induced counter torque is found to negatively rise during the opening phase of the lamella of a particular rotor arm. This is understood to be inline with the physical expectation in such a scenario.
- Regions of high velocity are identified at the lamella gaps which could be rectified by the use of porous zones at the gap region. Thus, the modelling approximation made could be nullified and the computational model can be brought closer to the actual physical behaviour.
- The solver is currently fucntional in serial and is needed to be parallelised in order to facilitate the three dimensional solution of the turbine dynamics and the use of the low Reynold's number turbulence model. This originates from the large mesh counts involved in 3D simulations and full resolution of wall boundary layers.

Restating and answering the research questions, we have:

1. *What is the extent to which the two dimensional simulation model accurately predicts the performance characteristics of the turbine and is an extension of the model to three dimensions necessary?*

   The two dimensional simulation model overpredicts the power generation capabilities of the turbine by a factor of two, while approximating trend quite accurately. Extending this current model to three dimensions is highly recommended and is expected to result in furthering the accuracy of the simulation mechanism.

2. *How suitable are the chosen modelling approaches in capturing the unique operational behaviour of the turbine?*

   The modelling approach to capture the non-smooth dynamics of the turbine lamellas are found to be insufficient to ensure stability in the fluid solution. Also, the lack of coupling in the fluid-structure solver adds to the local instabilities in the time varying fluid solution. Apart from these two issues, the remaining choice of modelling approaches are found to be suitable in capturing the operational behaviour of the turbine.

7.2. Favourable Outcomes

Enumerated below are the merits of the conducted research.

1. Having built the foundation of the required simulation approach on OpenFOAM, a fundamental means of simulating the turbine dynamics have been established. The

developed OpenFOAM class named *nestedRotatingNonSmoothFSIMotion* can act as a starting template for further development on the turbine model, wherein more resolute modelling approaches can be investigated. Considering the steep learning curve involved in writing new applications or functionalities in OpenFOAM, the foundation laid in this work can enable future work to progress unhindered by software architecture related issues.

2. The two dimensional turbine animation generated from the current state of the simulation mechanism, acts as a proof of concept that the operational characteristics of the turbine can be reproduced numerically. However, the animation may not be wholly accurate with regard to the flow dynamics or the dynamic character of the lamellas, it creates precedence where there was none to start with.

3. The results from the simulations carried out in this work, establishes a baseline for the predictive capabilities of future developed solvers. Thus, future work would have both experimental (from MARIN) and numerical results to motivate the research progression.

4. The simulations produce an accurate prediction of the torque curve trend. The trending character simulation results allow for a quick and moderately accurate determination of the optimal rotational speed for the considered turbine design and setup.

7.3. Recommendations

Listed below are a number of recommendations for future work with regard to further developing and investigating the developed solver.

- The lack of parallelisability in the solver computation, stands as the bottleneck to further investigations. This improvement in the solver is identified as future work of highest priority with regard to further solver development.

- Once the solver has been parallelised, the simulation mechanism can be extended to 3D solutions and a full three dimensional evaluation of the solver can be carried out. The 3D simulation only becomes computationally feasible when the solution can be performed in parallel.

- With parallelisation comes the capability to solve the fluid solution using a turbulence model as compared to the laminar solutions performed in this study. Water being the working fluid, the appropriate use of turbulence modelling can be pivotal in estimating the fluid solution accurately.

- By incorporating porous media zones at the lamella gaps identified in **Sec** 4.1.1, the relatively higher velocity flow through those gaps could be blocked, thereby bringing the simulation flow dynamics closer to reality.

- The modelling of the non-smooth dynamics of the lamella's could be improved by a more gradual handling of contact phenomena. The incorporation of an iterative coupling scheme between the fluid and structural solver could minimise instabilities in the fluid solution and help reduce the spikes in the torque time signals.

The addition of a physical damper to desensitise the lamella motion could be a worthwhile investigation to improve periodicity of the lamella's angular motion dynamics.

- The development of a scaling means in translating the 2D solution into a 3D result would prove computationally effective and minimise turn around time when used for turbine optimisation.
- HKV hopes to perform future studies on the effect of the OWM on the flow dynamics and morphology of the water body the turbine is to be placed in. To this effect, the available simulation model would help generate the data needed to fuel the turbine "black box" model, which would then be used to simulate its effect on the external flow and morphology of the water body it is deployed in.

Such a numerical simulation would include a case setup similar to the one used in this study and is specifically visualised in **Fig 7.1**. The components of such a simulation setup could include the following,

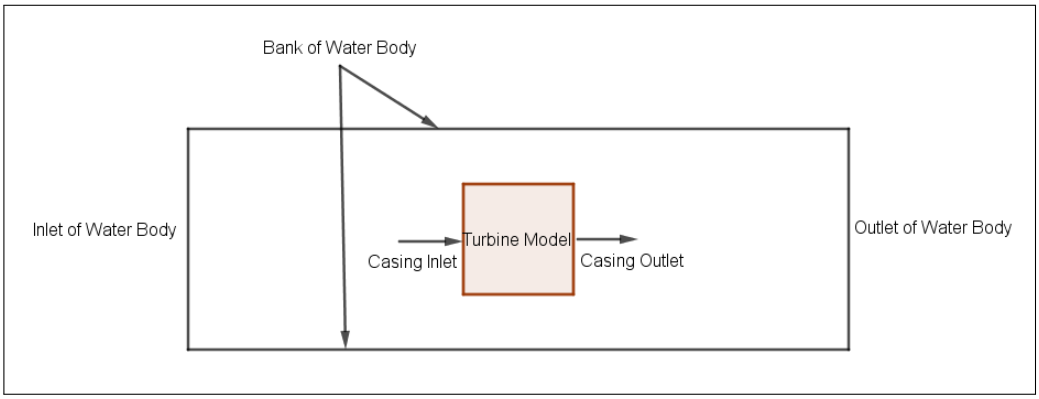


Figure 7.1: External Flow Study Case Setup

  - A turbine model that consists of a relation between the pressure drop across the turbine, the turbine rotational speed, the inlet velocity into the turbine casing, the outlet velocity from the turbine casing and the torque generated by the turbine. This relation constitutes the black box model of the turbine and would work in tandem with the external flow solution to iteratively arrive at a match between one another's turbine casing inlet and outlet velocity predictions.
  - The iterative nature of the coupling between the external flow solution and black box model solutions refer to an intra simulation method, wherein, the appropriate turbine inlet and outlet profiles are converged towards.
  - Simulating only the turbine using an improved form of the current solver and then using its resultant data to run the black box model within the external flow solution, is a more stage wise and separated process. The simulation of the external flow and turbine dynamics simultaneously is expected to be a more complex means, though it is the most intuitive.

- The external flow setup includes the boundary conditions that describe the behaviour of the external flow domain boundaries, such as the banks of the water body in consideration, the inlet and outlet boundaries to and from said water body and the relevant flow model needed to predict multiphase phenomena of sediment transport within the water body.

It is important to note that if a three dimensional hydraulic and morphological study is to be conducted, the solution of the turbine functionality and the black box approach should reflect the same dimensional extent.

"Essentially, all models are wrong, but some are useful"

George E. P. Box

Appendices

A

OpenFOAM Solver Class Code

The developed structural solver for the Oryon Watermill is an OpenFOAM class named nestedRotatingNonSmoothFSIMotion. The source and header file for the developed class is presented below.

A.1. nestedRotatingNonSmoothFSIMotion.H

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2013 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Class
    Foam::solidBodyMotionFunctions::nestedRotatingNonSmoothFSIMotion

Description
    SolidBodyMotionFvMesh 6DoF motion function.

SourceFiles
    nestedRotatingNonSmoothFSIMotion.C

\*---------------------------------------------------------------------------*/

#ifndef nestedRotatingNonSmoothFSIMotion_H
#define nestedRotatingNonSmoothFSIMotion_H

#include "solidBodyMotionFunction.H"
#include "primitiveFields.H"
#include "point.H"
#include "DataEntry.H"
#include "autoPtr.H"
#include "HashSet.H"
#include "volFields.H"
#include "sixDoFRigidBodyMotion.H"
#include "displacementMotionSolver.H"

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

```
namespace Foam
{
namespace solidBodyMotionFunctions
{

/*---------------------------------------------------------------------------*\
    Class nestedRotatingNonSmoothFSIMotion Declaration
\*---------------------------------------------------------------------------*/

class nestedRotatingNonSmoothFSIMotion
:
    public solidBodyMotionFunction
{
    // Private data

        //- Origin of the outer axis
        const vector outer_origin_;

 //- Initial origin of nested axis
        const vector nested_origin0_;

 //- Nested axis vector
        const vector nested_axis_;

        //- Outer angular velocty (rad/sec)
 const scalar outer_omega_;

 // Nested Body Initial Torque Calculation Point
 vector TCP0_;

        //Derived Torue or Torque output from forces.H solver
        scalar DT_;

 //Nested Body Initial Trailing Edge Point
        vector TE0;

 //Nested Body Unshifted Trailing Edge Point
        vector TE_US0;

 // Nested Body Updated Trailing Edge Point
        vector TE;

 //Nested Body Updated Trailing Edge Point
        vector TE_US;

 // Nested Body Updated Torque Calculation Point
 vector TCP_;

 //fvmesh
 const fvMesh& mesh;

 // list of Boundary Surface Patches for Force Calculations
 wordReList patches_;

        //- Patches to integrate forces
```

```
        const labelHashSet patchSet_;

//- Reference density required by the forces object for
        //  incompressible calculations, required if rhoName == rhoInf
        scalar rhoInf_;

        //- Name of density field, optional unless used for an
        //  incompressible simulation, when this needs to be specified
        //  as rhoInf
        word rhoName_;

//- Stopper Constarint Active??
scalar stopConstraint;

//Angular Size of Proximity Region in degrees
        scalar proximityRegionSize;

//Proximity Boundary Counter Clockwise
        scalar proximityBoundaryPos;

//Proximity Boundary Clockwise
scalar proximityBoundaryNeg;

//- Current time index (used for updating)
label curTimeIndex_;

//- Six dof motion object
        sixDoFRigidBodyMotion motion_;

//System Torque
vector tau;

//Fun with Flags
scalar flagPos;
scalar flagNeg;

    // Private Member Functions

        //- Disallow copy construct
        nestedRotatingNonSmoothFSIMotion(const
            ↪ nestedRotatingNonSmoothFSIMotion&);

        //- Disallow default bitwise assignment
        void operator=(const nestedRotatingNonSmoothFSIMotion&);


public:

    //- Runtime type information
    TypeName("nestedRotatingNonSmoothFSIMotion");


    // Constructors

        //- Construct from components
        nestedRotatingNonSmoothFSIMotion
```

```
        (
            const dictionary& SBMFCoeffs,
            const Time& runTime
        );

        //- Construct and return a clone
        virtual autoPtr<solidBodyMotionFunction> clone() const
        {
            return autoPtr<solidBodyMotionFunction>
            (
                new nestedRotatingNonSmoothFSIMotion
                (
                    SBMFCoeffs_,
                    time_
                )
            );
        }


    //- Destructor
    virtual ~nestedRotatingNonSmoothFSIMotion();


    // Member Functions

        //- Return the solid-body motion transformation septernion
        virtual septernion transformation();

        //- Update properties from given dictionary
        virtual bool read(const dictionary& SBMFCoeffs);

};


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

} // End namespace solidBodyMotionFunctions
} // End namespace Foam

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

#endif

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

A.2. nestedRotatingNonSmoothFSIMotion.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2014 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------

\*---------------------------------------------------------------------------*/

#include "nestedRotatingNonSmoothFSIMotion.H"
#include "addToRunTimeSelectionTable.H"
#include "polyMesh.H"
#include "pointPatchDist.H"
#include "pointConstraints.H"
#include "uniformDimensionedFields.H"
#include "forces.H"
#include "mathematicalConstants.H"
#include "coordinateSystem.H"


using namespace Foam::constant::mathematical;

// * * * * * * * * * * *  Static Data Members * * * * * * * * *//

namespace Foam
{
namespace solidBodyMotionFunctions
{
    defineTypeNameAndDebug(nestedRotatingNonSmoothFSIMotion, 0);
    addToRunTimeSelectionTable
    (
        solidBodyMotionFunction,
        nestedRotatingNonSmoothFSIMotion,
        dictionary
    );
}
}


// * * * * * * * * * * * * * *  Constructors  * * * * * * * * * * * * * //

Foam::solidBodyMotionFunctions::nestedRotatingNonSmoothFSIMotion::
    ↪ nestedRotatingNonSmoothFSIMotion
(
    const dictionary& SBMFCoeffs,
    const Time& runTime
)
:
    solidBodyMotionFunction(SBMFCoeffs, runTime),
    outer_origin_(SBMFCoeffs_.lookup("outerOrigin")),
    nested_origin0_(SBMFCoeffs_.lookup("initialNestedOrigin")),
    nested_axis_(SBMFCoeffs_.lookup("nestedAxis")),
    outer_omega_(readScalar(SBMFCoeffs_.lookup("outerOmega"))),
```

```
    TCP0_(SBMFCoeffs_.lookup("initialTCP")),
    DT_(readScalar(SBMFCoeffs_.lookup("derivedTorque"))),
    TE0(SBMFCoeffs_.lookupOrDefault("initialTrailingEdge", vector::zero)),
    TE_US0(SBMFCoeffs_.lookupOrDefault("initialTrailingEdgeUnShifted",
        ↪ vector::zero)),

    mesh(time_.db().parent().lookupObject<fvMesh>("region0")),
    patches_(wordReList(SBMFCoeffs_.lookup("patches"))),
    patchSet_(mesh.boundaryMesh().patchSet(patches_)),
    rhoInf_(1.0),
    rhoName_(SBMFCoeffs_.lookupOrDefault<word>("rhoName", "rho")),

    stopConstraint(readScalar(SBMFCoeffs_.lookup("stopConstraint"))),
    proximityRegionSize(readScalar(SBMFCoeffs_.lookup("proximityRegionSize
        ↪ "))),
    proximityBoundaryPos(readScalar(SBMFCoeffs_.lookup("
        ↪ proximityBoundaryPos"))),
    proximityBoundaryNeg(readScalar(SBMFCoeffs_.lookup("
        ↪ proximityBoundaryNeg"))),
    curTimeIndex_(-1),
    motion_(SBMFCoeffs_, SBMFCoeffs_)
{
  if (rhoName_ == "rhoInf")
    {
        rhoInf_ = readScalar(SBMFCoeffs_.lookup("rhoInf"));
    }

  tau = vector::zero;
  TE = vector::zero;
  TE_US = vector::zero;
  TCP_ = vector::zero;
  flagPos = 0;
  flagNeg = 0;
}


// * * * * * * * * * * * * * * * Destructors * * * * * * * * * * * * * //

Foam::solidBodyMotionFunctions::nestedRotatingNonSmoothFSIMotion::~
    ↪ nestedRotatingNonSmoothFSIMotion()
{}


// * * * * * * * * * * * * * * Member Functions * * * * * * * * * * * * //

Foam::septernion
Foam::solidBodyMotionFunctions::nestedRotatingNonSmoothFSIMotion::
    ↪ transformation()
{
    const Time& t1 = mesh.time();
    Info<<"_____________________________________________________"<<nl;

    // Store the motion state at the beginning of the time-step
    if (curTimeIndex_ != this->time_.timeIndex())
    {
        motion_.newTime();
```

```
        curTimeIndex_ = this->time_.timeIndex();
    }

    scalar theta = motion_.rotationalMatrixToAngle();
    //Info<<"Angular Position in Degrees : "<<theta<<nl;

    // Proximity Check for Non-Smooth Dynamics

if(stopConstraint == 1)
{
    Info<<"Non-Smooth Dynamics Under Consideration"<<nl;

    if(proximityBoundaryPos - proximityRegionSize < 0)
 {
  if(theta > 360 - proximityRegionSize )
  {
   theta = theta - 360;
   flagPos = 1;
  }
 }

    if((tau.z() > 0 || motion_.pi().z() > motion_.momentOfInertia_.zz()*
        ↪ outer_omega_) && (theta >= (proximityBoundaryPos -
        ↪ proximityRegionSize) && theta <= proximityBoundaryPos ))
    {
      motion_.stopped = 1;
      motion_.torqueCalibrate = 0;
      Info<<"CCW Constraint Event : POS "<<nl;
      Info<<"Torque : "<<tau.z()<<nl<<"Angular Momentum : "<<motion_.pi()
          ↪ .z()<<nl;

      if(flagPos == 1)
 {
   Info<<"Angular Position : "<<theta+360<<nl;
   flagPos = 0;
 }
      else
 {
   Info<<"Angular Position : "<<theta<<nl;
 }

       Info<<"Current Proximity Boundaries : "<<proximityBoundaryPos<<" "
          ↪ <<proximityBoundaryNeg<<nl;

    }

    else
    {
 if(proximityBoundaryNeg + proximityRegionSize > 360)
 {
  if(theta < proximityRegionSize )
  {
   theta = theta+360;
   flagNeg = 1;
  }
 }
```

```
      if((tau.z() < 0 || motion_.pi().z() < motion_.momentOfInertia_.zz()*
         ↪ outer_omega_) && (theta <= (proximityBoundaryNeg +
         ↪ proximityRegionSize) && theta >= proximityBoundaryNeg ))
        {
           motion_.stopped = 1;
    motion_.torqueCalibrate = 0;
    Info<<"CW Constraint Event : NEG "<<nl;
           Info<<"Torque : "<<tau.z()<<nl<<"Angular Momentum : "<<motion_.
              ↪ pi().z()<<nl;

    if(flagNeg == 1)
     {
     Info<<"Angular Position : "<<theta-360<<nl;
     flagNeg = 0;
     }
    else
    {
     Info<<"Angular Position : "<<theta<<nl;
    }

    Info<<"Current Proximity Boundaries : "<<proximityBoundaryPos<<" "<<
       ↪ proximityBoundaryNeg<<nl;

        }


        else
        {
    motion_.torqueCalibrate = 1;
    motion_.stopped = 0;
    Info<<"Non-Constraint Event "<<nl;
    Info<<"Torque : "<<tau.z()<<nl<<"Angular Momentum : "<<motion_.pi().z
       ↪ ()<<nl<<"Angular Position : "<<theta<<nl;
    Info<<"Current Proximity Boundaries : "<<proximityBoundaryPos<<" "<<
       ↪ proximityBoundaryNeg<<nl;

        }
     }
}

    // Outer Angular Position
    scalar t = time_.value();
    scalar outer_angle = outer_omega_ * t;

    // Update Proximity Boundaries
    scalar outer_angle_update_deg = (outer_omega_ * t1.deltaTValue())
       ↪ *(180.0/constant::mathematical::pi);
    proximityBoundaryPos = motion_.angleCheck(proximityBoundaryPos +
       ↪ outer_angle_update_deg);
    proximityBoundaryNeg = motion_.angleCheck(proximityBoundaryNeg +
       ↪ outer_angle_update_deg);
    Info<<"Updated Proximity Boundaries : "<<proximityBoundaryPos<<"    "<<
       ↪ proximityBoundaryNeg<<nl;

    // Orientation Tensor Update corresponding to Proximity Boundary
```

```
    ↪ Update / Orientation Tensor Update corresponding to Angular
    ↪ momentum
// Velocity Verlet Position Update
motion_.updatePosition(t1.deltaTValue(), t1.deltaT0Value(),
    ↪ outer_omega_ );

// Rotation Matrix about z-axis
vector rel_nested_origin0_ = nested_origin0_ - outer_origin_;
tensor RM_(cos(outer_angle), -sin(outer_angle), 0, sin(outer_angle),
    ↪ cos(outer_angle), 0, 0, 0, 1);
vector rel_nested_origin_ = RM_ & rel_nested_origin0_;
Info<<"Linear Motion of NBO from : "<<rel_nested_origin0_<<" to : "<<
    ↪ rel_nested_origin_<<nl;

septernion s = motion_.transformation(rel_nested_origin_);
Info<<"<<-------------------------->>"<<nl;

TCP_ = s.transform(TCP0_); // Update Current Time Step TCP
Info<<"TCP : "<<TCP_;


// Torque/Force Dictionaries
TE = s.transform(TE0);
TE_US = s.transform(TE_US0);  // For Torque Check in DT_ = 0

vector GTC = vector::zero;
vector GTC_f = vector::zero;
vector GTC_t = vector::zero;

vector T_c;
vector F_c;

dictionary forcesDict;
forcesDict.add("type", forces::typeName);  // Center or Pivot Point
forcesDict.add("patches", patches_);
forcesDict.add("rhoInf", rhoInf_);
forcesDict.add("rhoName", rhoName_);
forcesDict.add("CofR", rel_nested_origin_);

forces f("forces",mesh, forcesDict);
f.calcForcesMoment();
T_c = f.momentEff();
F_c = f.forceEff();

vector T_TE;
vector F_TE;

dictionary forcesDict__;
forcesDict__.add("type", forces::typeName);  // Opposite Point of TCP
forcesDict__.add("patches", patches_);
forcesDict__.add("rhoInf", rhoInf_);
forcesDict__.add("rhoName", rhoName_);
forcesDict__.add("CofR", TE);

forces f__("forces",mesh, forcesDict__);
f__.calcForcesMoment();
```

```
    T_TE = f___.momentEff();
    F_TE = f___.forceEff();

    vector T_TE_US;
    vector F_TE_US;

    dictionary forcesDict_;
    forcesDict_.add("type", forces::typeName); // Rather Pointless
        ↪ Dictionary
    forcesDict_.add("patches", patches_);
    forcesDict_.add("rhoInf", rhoInf_);
    forcesDict_.add("rhoName", rhoName_);
    forcesDict_.add("CofR", TE_US);

    forces f_("forces",mesh, forcesDict_);
    f_.calcForcesMoment();
    T_TE_US = f_.momentEff();
    F_TE_US = f_.forceEff();

    vector T_TCP;
    vector F_TCP;

    dictionary forcesDict____;
    forcesDict____.add("type", forces::typeName);  // Torque Calculation
        ↪ Point
    forcesDict____.add("patches", patches_);
    forcesDict____.add("rhoInf", rhoInf_);
    forcesDict____.add("rhoName", rhoName_);
    forcesDict____.add("CofR", TCP_);

    forces f____("forces",mesh, forcesDict____);
    f____.calcForcesMoment();
    T_TCP = f____.momentEff();
    F_TCP = f____.forceEff();

    if(DT_ == 1)
    {
//Shift from Center to TCP
        //Info<<"Torque Derived from Net Torque and Net Force at Pivot
            ↪ Point(Center of Rotation)"<<nl;
     //tau = T_c + ((rel_nested_origin_ - TCP_) ^ F_c);

// Shift from TE to TCP
Info<<"Torque at TCP, Derived from Net Torque and Net Force at Trailing
    ↪ Edge"<<nl; // GTC_Verification
     tau = T_TE + ((TE - TCP_) ^ F_TE);

Info<<"Radius Vector from NO to TCP : "<< mag(rel_nested_origin_ - TCP_ )
    ↪  <<nl;
Info<<"Radius Vector from TCP to TE : "<< mag(TE - TCP_ ) <<nl;
Info<<"Effective Torque at TCP - Calculated from LE Data : "<<tau<<nl;
Info<<"Effective Force at NO : "<<F_c<<nl;
     Info<<"Effective Torque at NO : "<<T_c<<nl;
Info<<"Effective Force at TCP : "<<F_TCP<<nl;
     Info<<"Effective Torque at TCP : "<<T_TCP<<nl;
    }
```

```
    else
    {
if(DT_ ==2)
{
 // Shift from TE to TCP
 Info<<"Torque Derived from Net Torque and Net Force at Trailing Edge"<<
     ↪ nl;
      tau = T_TE + ((TE - TCP_) ^ F_TE);
 Info<<"Radius Vector from TCP to TE : "<< mag(TE - TCP_ ) <<nl;
 Info<<"Effective Force at TE : "<<F_TE<<nl;
     Info<<"Effective Torque at TE : "<<T_TE<<nl;
     Info<<"Net Torque at TCP : "<<tau<<nl;
 // Direct Torque at TCP Determination
 Info<<"Net Torque at TCP derived from forces class Calculator : "<<T_TCP
     ↪ <<nl;
 Info<<"Net Force at TCP derived from forces class Calculator : "<<F_TCP
     ↪ <<nl;
}
else
{

 Info<<"Torque Output from forces class Calculator"<<nl;
 tau = T_c;
 Info<<"Torque at CofR : "<<tau<<nl;

 //Shift from TE_US to Center
 vector checkTau = T_TE_US + ((TE_US - rel_nested_origin_ ) ^ F_TE_US);
 Info<<"Radius Vector from TE_US to NO : "<< mag(TE_US -
     ↪ rel_nested_origin_) <<nl;    // Not applicable except for checking
     ↪ formulation
 Info<<"Effective Force at TE_US : "<<F_TE_US<<nl;
     Info<<"Effective Torque at TE_US : "<<T_TE_US<<nl;
 Info<<"Torque at CofR Check : "<<checkTau<<nl;
 Info<<"Trailing Edge - Unshifted : "<<TE_US<<nl;
}
    }

    Info<<"Torque Calibrate : "<<motion_.torqueCalibrate<<nl;
    if(motion_.torqueCalibrate == 1)
    {
      //Calibrating torqe for use at Center of Rotation to match motion of
          ↪  Physical Case
 Info<<"Ratio Numerator : "<<motion_.momentOfInertia_.zz()<<nl;
 Info<<"Ratio Denominator : "<<motion_.momentOfInertia_Comp<<nl;
     tau = (motion_.momentOfInertia_.zz()/motion_.momentOfInertia_Comp)*
         ↪ tau;
 Info<<"Calibrated Torque : "<<tau<<nl;

 GTC_f = ((rel_nested_origin_) ^ F_c);
 Info<<"Only GTC_f contributes to GTC"<<nl;
 Info<<"GTC_f : "<<GTC_f<<nl;
 Info<<"NO : "<<rel_nested_origin_<<nl;
GTC = GTC_f;
 Info<<"GTC : "<<GTC<<nl;
    }
```

```
    else
    {


        // Calibrated Torque determined to use for Overall Turbine Torque
            ↪ calculation
  Info<<"Ratio Numerator : "<<motion_.momentOfInertia_.zz()<<nl;
  Info<<"Ratio Denominator : "<<motion_.momentOfInertia_Comp<<nl;

  GTC_t = ((motion_.momentOfInertia_.zz()/motion_.momentOfInertia_Comp)*
      ↪ tau);
  GTC_f = ((rel_nested_origin_) ^ F_c);
  Info<<"Both GTC_f and GTC_t contributes to GTC"<<nl;
  Info<<"GTC_t : "<<GTC_t<<nl;
  Info<<"GTC_f : "<<GTC_f<<nl;
  Info<<"NO : "<<rel_nested_origin_<<nl;
  GTC = GTC_t + GTC_f;
   Info<<"GTC : "<<GTC<<nl;
    }

    dimensionedVector g("g", dimAcceleration, vector::zero);

    if (time_.db().parent().foundObject<uniformDimensionedVectorField>("g"
        ↪ ))
    {
        g = time_.db().parent().lookupObject<uniformDimensionedVectorField
            ↪ >("g");
    }
    else if (SBMFCoeffs_.found("g"))
    {
        SBMFCoeffs_.lookup("g") >> g;
    }

    scalar ramp = 1.0;

    // Velocity Verlet Acceleration Update
    motion_.updateAcceleration
    (
        ramp*(tau + motion_.mass()*(motion_.momentArm() ^ g.value())),
        t1.deltaTValue(),
TCP_
    );

    return s;
}


bool Foam::solidBodyMotionFunctions::nestedRotatingNonSmoothFSIMotion::
    ↪ read
(
    const dictionary& SBMFCoeffs
)
{
    solidBodyMotionFunction::read(SBMFCoeffs);

    //outer_omega_.reset
```

```
    //(
    //    DataEntry<scalar>::New("outerOmega", SBMFCoeffs_).ptr()
    //);

    //nested_omega_.reset
    //(
    //    DataEntry<scalar>::New("nestedOmega", SBMFCoeffs_).ptr()
    //);

    return true;
}


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

B

Solver Class Dictionary File

The dictionary file needed to setup the case for the Oryon Watermill structural solver for the $RPS = 0.6$ simulation instance is the following dynamicMeshDict file defiition.

B.1. dynamicMeshDict

```
/*------ --------- --------*- C++ -*-------- --------- --------- *\
| ========                 |
    ↪                                                    |
| \\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
    ↪           |
|  \\    /   O peration    | Version:  2.4.0
    ↪                                          |
|   \\  /    A nd          | Web:      www.OpenFOAM.org
    ↪                      |
|    \\/     M anipulation |
    ↪                                                    |
\*-------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dynamicFvMesh    multiSolidBodyMotionFvMesh;

dynamicFvMeshLibs ( "libdynamicFvMeshVin.so" );

multiSolidBodyMotionFvMeshCoeffs
{
 FLUID_LAM_1
 {
  solidBodyMotionFunction   nestedRotatingNonSmoothFSIMotion;
  nestedRotatingNonSmoothFSIMotionCoeffs
  {
```

```
outerOrigin (0 0 0); // Center of Revolution of Nested Body
outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

initialNestedOrigin (0.03535909 -0.06123851 0); // Initial Center of
    ↪ Rotation of Nested Body
initialTCP (0.04757500 -0.08240231 0); // Initial Torque Calculation
    ↪ Point of Nested Body
nestedAxis (0 0 1); // Axis of Rotation of Nested Body
patches (WALL_LAM_1);
derivedTorque 2;
centreOfMass (0.03535909 -0.06123851 0);
centreOfRotation (0.03535909 -0.06123851 0); // To establish the
    ↪ distance between centerOfMass and centerOfRotation (MI
    ↪ calculation)

initialTrailingEdge (0.01625699 -0.02815793 0);
torqueCalibrate 1;
momentOfInertiaComparative  5.445e-06;

//Setting Constraint Conditions
stopConstraint 1; // Enter or exit constraint loop
proximityBoundaryPos 360;
proximityBoundaryNeg  270;
proximityRegionSize 1;
mass 4.984e-03;
momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
orientation //Oriented at 0 Degrees
(
   1 0 0
   0 1 0
   0 0 1
);
velocity (0 0 0);
acceleration (0 0 0);
angularMomentum (0 0 0);
rhoName rhoInf;
rhoInf 1000;
report on;
accelerationRelaxation 0.5;
accelerationDamping 1;
value uniform (0 0 0);
solver
{
 type symplectic;
}
constraints
{
 zAxis
 {
  sixDoFRigidBodyMotionConstraint axis;
  axis           (0 0 1);
 }
}
restraints
{
 translationDamper
```

```
    {
     sixDoFRigidBodyMotionRestraint linearDamper;
     coeff           0.001;
    }
   }
  }
 }

FLUID_LAM_2
{
 solidBodyMotionFunction  nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin  (0.07561991 -0.1309797 0); // Initial Center of
     ↪ Rotation of Nested Body
  initialTCP   (0.08783914 -0.1521419 0); // Initial Torque Calculation
     ↪ Point of Nested Body
  nestedAxis          (0 0 1); // Axis of Rotation of Nested Body
  patches           (WALL_LAM_2);
  derivedTorque   2;
  centreOfMass   (0.07561991 -0.1309797 0);
  centreOfRotation  (0.07561991 -0.1309797 0); // To establish the
     ↪ distance between centerOfMass and centerOfRotation (MI
     ↪ calculation)

  initialTrailingEdge  (0.05652113 -0.09789747 0);
  torqueCalibrate  1;
  momentOfInertiaComparative  5.445e-06;

  //Setting Constraint Conditions
  stopConstraint 1; // Enter or exit constraint loop
  proximityBoundaryPos 360;
  proximityBoundaryNeg  270;
  proximityRegionSize 1;
  mass 4.984e-03;
  momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
  orientation //Oriented at 0 Degrees
  (
     1 0 0
     0 1 0
     0 0 1
  );
  velocity (0 0 0);
  acceleration (0 0 0);
  angularMomentum (0 0 0);
  rhoName rhoInf;
  rhoInf 1000;
  report on;
  accelerationRelaxation 0.5;
  accelerationDamping 1;
  value uniform (0 0 0);
  solver
  {
```

```
    type symplectic;
   }
   constraints
   {
    zAxis
    {
     sixDoFRigidBodyMotionConstraint axis;
     axis           (0 0 1);
    }
   }
   restraints
   {
    translationDamper
    {
     sixDoFRigidBodyMotionRestraint linearDamper;
     coeff          0.001;
    }
   }
  }
}

FLUID_LAM_3
{
 solidBodyMotionFunction   nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin  (0.1158844 -0.2007153 0); // Initial Center of
     ↪ Rotation of Nested Body
  initialTCP    (0.1281033 -0.2218814 0); // Initial Torque Calculation
     ↪ Point of Nested Body
  nestedAxis           (0 0 1); // Axis of Rotation of Nested Body
  patches            (WALL_LAM_3);
  derivedTorque    2;
  centreOfMass    (0.1158844 -0.2007153 0);
  centreOfRotation  (0.1158844 -0.2007153 0); // To establish the
     ↪ distance between centerOfMass and centerOfRotation (MI
     ↪ calculation)

  initialTrailingEdge  (0.09678527 -0.1676370 0);
  torqueCalibrate  1;
  momentOfInertiaComparative  5.445e-06;

  //Setting Constraint Conditions
  stopConstraint 1; // Enter or exit constraint loop
  proximityBoundaryPos 360;
  proximityBoundaryNeg  270;
  proximityRegionSize 1;
  mass 4.984e-03;
  momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
  orientation //Oriented at 0 Degrees
  (
     1 0 0
     0 1 0
```

```
    0 0 1
  );
  velocity (0 0 0);
  acceleration (0 0 0);
  angularMomentum (0 0 0);
  rhoName rhoInf;
  rhoInf 1000;
  report on;
  accelerationRelaxation 0.5;
  accelerationDamping 1;
  value uniform (0 0 0);
  solver
  {
   type symplectic;
  }
  constraints
  {
   zAxis
   {
    sixDoFRigidBodyMotionConstraint axis;
    axis          (0 0 1);
   }
  }
  restraints
  {
   translationDamper
   {
    sixDoFRigidBodyMotionRestraint linearDamper;
    coeff         0.001;
   }
  }
 }
}

FLUID_LAM_4
{
 solidBodyMotionFunction  nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin (0.1561442 -0.2704536 0); // Initial Center of
     ↪ Rotation of Nested Body
  initialTCP (0.1683674 -0.2916209 0); // Initial Torque Calculation
     ↪ Point of Nested Body
  nestedAxis (0 0 1); // Axis of Rotation of Nested Body
  patches (WALL_LAM_4);
  derivedTorque 2;
  centreOfMass (0.1561442 -0.2704536 0);
  centreOfRotation (0.1561442 -0.2704536 0); // To establish the distance
     ↪  between centerOfMass and centerOfRotation (MI calculation)

  initialTrailingEdge (0.1370494 -0.2373766 0);
  torqueCalibrate 1;
  momentOfInertiaComparative 5.445e-06;
```

```
    //Setting Constraint Conditions
    stopConstraint 1; // Enter or exit constraint loop
    proximityBoundaryPos 360;
    proximityBoundaryNeg  270;
    proximityRegionSize 1;
    mass 4.984e-03;
    momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
    orientation //Oriented at 0 Degrees
    (
       1 0 0
       0 1 0
       0 0 1
    );
    velocity (0 0 0);
    acceleration (0 0 0);
    angularMomentum (0 0 0);
    rhoName rhoInf;
    rhoInf 1000;
    report on;
    accelerationRelaxation 0.5;
    accelerationDamping 1;
    value uniform (0 0 0);
    solver
    {
     type symplectic;
    }
    constraints
    {
     zAxis
     {
      sixDoFRigidBodyMotionConstraint axis;
      axis          (0 0 1);
     }
    }
    restraints
    {
     translationDamper
     {
      sixDoFRigidBodyMotionRestraint linearDamper;
      coeff         0.001;
     }
    }
   }
}

FLUID_LAM_5
{
 solidBodyMotionFunction   nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin (-0.07072272 0 0); // Initial Center of Rotation of
     ↪ Nested Body
```

```
    initialTCP (-0.09515 0 0); // Initial Torque Calculation Point of
        ↪ Nested Body
    nestedAxis (0 0 1); // Axis of Rotation of Nested Body
    patches (WALL_LAM_5);
    derivedTorque 2;
    centreOfMass (-0.07072272 0 0);
    centreOfRotation (-0.07072272 0 0); // To establish the distance
        ↪ between centerOfMass and centerOfRotation (MI calculation)

    initialTrailingEdge (-0.03251398 0 0);
    torqueCalibrate 1;
    momentOfInertiaComparative 5.445e-06;

    //Setting Constraint Conditions
    stopConstraint 1; // Enter or exit constraint loop
    proximityBoundaryPos 360;
    proximityBoundaryNeg  270;
    proximityRegionSize 1;
    mass 4.984e-03;
    momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
    orientation //Oriented at 0 Degrees
    (
       1 0 0
       0 1 0
       0 0 1
    );
    velocity (0 0 0);
    acceleration (0 0 0);
    angularMomentum (0 0 0);
    rhoName rhoInf;
    rhoInf 1000;
    report on;
    accelerationRelaxation 0.5;
    accelerationDamping 1;
    value uniform (0 0 0);
    solver
    {
     type symplectic;
    }
    constraints
    {
     zAxis
     {
      sixDoFRigidBodyMotionConstraint axis;
      axis           (0 0 1);
     }
    }
    restraints
    {
     translationDamper
     {
      sixDoFRigidBodyMotionRestraint linearDamper;
      coeff          0.001;
     }
    }
}
```

```
}

FLUID_LAM_6
{
 solidBodyMotionFunction   nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin (-0.1512394 0 0); // Initial Center of Rotation of
     ↪ Nested Body
  initialTCP (-0.1756783 0 0); // Initial Torque Calculation Point of
     ↪ Nested Body
  nestedAxis (0 0 1); // Axis of Rotation of Nested Body
  patches (WALL_LAM_6);
  derivedTorque 2;
  centreOfMass (-0.1512394 0 0);
  centreOfRotation (-0.1512394 0 0); // To establish the distance between
     ↪  centerOfMass and centerOfRotation (MI calculation)

  initialTrailingEdge (-0.1130423 0 0);
  torqueCalibrate 1;
  momentOfInertiaComparative 5.445e-06;

  //Setting Constraint Conditions
  stopConstraint 1; // Enter or exit constraint loop
  proximityBoundaryPos 360;
  proximityBoundaryNeg  270;
  proximityRegionSize 1;
  mass 4.984e-03;
  momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
  orientation //Oriented at 0 Degrees
  (
     1 0 0
     0 1 0
     0 0 1
  );
  velocity (0 0 0);
  acceleration (0 0 0);
  angularMomentum (0 0 0);
  rhoName rhoInf;
  rhoInf 1000;
  report on;
  accelerationRelaxation 0.5;
  accelerationDamping 1;
  value uniform (0 0 0);
  solver
  {
   type symplectic;
  }
  constraints
  {
   zAxis
   {
    sixDoFRigidBodyMotionConstraint axis;
```

```
    axis           (0 0 1);
   }
  }
  restraints
  {
   translationDamper
   {
    sixDoFRigidBodyMotionRestraint linearDamper;
    coeff          0.001;
   }
  }
 }
}

FLUID_LAM_7
{
 solidBodyMotionFunction  nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin (-0.2317725 0 0); // Initial Center of Rotation of
     ↪ Nested Body
  initialTCP (-0.2562066 0 0); // Initial Torque Calculation Point of
     ↪ Nested Body
  nestedAxis (0 0 1); // Axis of Rotation of Nested Body
  patches (WALL_LAM_7);
  derivedTorque 2;
  centreOfMass (-0.2317725 0 0);
  centreOfRotation (-0.2317725 0 0); // To establish the distance between
     ↪  centerOfMass and centerOfRotation (MI calculation)

  initialTrailingEdge (-0.1935705 0 0);
  torqueCalibrate 1;
  momentOfInertiaComparative 5.445e-06;

  //Setting Constraint Conditions
  stopConstraint 1; // Enter or exit constraint loop
  proximityBoundaryPos 360;
  proximityBoundaryNeg  270;
  proximityRegionSize 1;
  mass 4.984e-03;
  momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
  orientation //Oriented at 0 Degrees
  (
     1 0 0
     0 1 0
     0 0 1
  );
  velocity (0 0 0);
  acceleration (0 0 0);
  angularMomentum (0 0 0);
  rhoName rhoInf;
  rhoInf 1000;
  report on;
```

```
    accelerationRelaxation 0.5;
    accelerationDamping 1;
    value uniform (0 0 0);
    solver
    {
     type symplectic;
    }
    constraints
    {
     zAxis
     {
      sixDoFRigidBodyMotionConstraint axis;
      axis           (0 0 1);
     }
    }
    restraints
    {
     translationDamper
     {
      sixDoFRigidBodyMotionRestraint linearDamper;
      coeff          0.001;
     }
    }
   }
}

FLUID_LAM_8
{
 solidBodyMotionFunction  nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin (-0.3122962 0 0); // Initial Center of Rotation of
     ↪ Nested Body
  initialTCP (-0.3367349 0 0); // Initial Torque Calculation Point of
     ↪ Nested Body
  nestedAxis (0 0 1); // Axis of Rotation of Nested Body
  patches (WALL_LAM_8);
  derivedTorque 2;
  centreOfMass (-0.3122962 0 0);
  centreOfRotation (-0.3122962 0 0); // To establish the distance between
     ↪  centerOfMass and centerOfRotation (MI calculation)

  initialTrailingEdge (-0.2740988 0 0);
  torqueCalibrate 1;
  momentOfInertiaComparative 5.445e-06;

  //Setting Constraint Conditions
  stopConstraint 1; // Enter or exit constraint loop
  proximityBoundaryPos 360;
  proximityBoundaryNeg  270;
  proximityRegionSize 1;
  mass 4.984e-03;
  momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
```

```
    orientation //Oriented at 0 Degrees
    (
        1 0 0
        0 1 0
        0 0 1
    );
    velocity (0 0 0);
    acceleration (0 0 0);
    angularMomentum (0 0 0);
    rhoName rhoInf;
    rhoInf 1000;
    report on;
    accelerationRelaxation 0.5;
    accelerationDamping 1;
    value uniform (0 0 0);
    solver
    {
     type symplectic;
    }
    constraints
    {
     zAxis
     {
      sixDoFRigidBodyMotionConstraint axis;
      axis           (0 0 1);
     }
    }
    restraints
    {
     translationDamper
     {
      sixDoFRigidBodyMotionRestraint linearDamper;
      coeff          0.001;
     }
    }
  }
}

FLUID_LAM_9
{
 solidBodyMotionFunction   nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin (0.03536044 0.06124303 0); // Initial Center of
     ↪ Rotation of Nested Body
  initialTCP (0.047575 0.08240232 0); // Initial Torque Calculation Point
     ↪  of Nested Body
  nestedAxis (0 0 1); // Axis of Rotation of Nested Body
  patches (WALL_LAM_9);
  derivedTorque 2;
  centreOfMass (0.03536044 0.06124303 0);
  centreOfRotation (0.03536044 0.06124303 0); // To establish the
     ↪ distance between centerOfMass and centerOfRotation (MI
```

```
    ↪ calculation)

  initialTrailingEdge (0.01625699 0.02815794 0);
  torqueCalibrate 1;
  momentOfInertiaComparative 5.445e-06;

  //Setting Constraint Conditions
  stopConstraint 1; // Enter or exit constraint loop
  proximityBoundaryPos 360;
  proximityBoundaryNeg  270;
  proximityRegionSize 1;
  mass 4.984e-03;
  momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
  orientation //Oriented at 0 Degrees
  (
     1 0 0
     0 1 0
     0 0 1
  );
  velocity (0 0 0);
  acceleration (0 0 0);
  angularMomentum (0 0 0);
  rhoName rhoInf;
  rhoInf 1000;
  report on;
  accelerationRelaxation 0.5;
  accelerationDamping 1;
  value uniform (0 0 0);
  solver
  {
   type symplectic;
  }
  constraints
  {
   zAxis
   {
    sixDoFRigidBodyMotionConstraint axis;
    axis           (0 0 1);
   }
  }
  restraints
  {
   translationDamper
   {
    sixDoFRigidBodyMotionRestraint linearDamper;
    coeff          0.001;
   }
  }
 }
}

FLUID_LAM_10
{
 solidBodyMotionFunction   nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
```

```
outerOrigin (0 0 0); // Center of Revolution of Nested Body
outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

initialNestedOrigin (0.07561909 0.1309770 0); // Initial Center of
    ↪ Rotation of Nested Body
initialTCP (0.08783914 0.1521419 0); // Initial Torque Calculation
    ↪ Point of Nested Body
nestedAxis (0 0 1); // Axis of Rotation of Nested Body
patches (WALL_LAM_10);
derivedTorque 2;
centreOfMass (0.07561909 0.1309770 0);
centreOfRotation (0.07561909 0.1309770 0); // To establish the distance
    ↪ between centerOfMass and centerOfRotation (MI calculation)

initialTrailingEdge (0.05652113 0.09789748 0);
torqueCalibrate 1;
momentOfInertiaComparative 5.445e-06;

//Setting Constraint Conditions
stopConstraint 1; // Enter or exit constraint loop
proximityBoundaryPos 360;
proximityBoundaryNeg  270;
proximityRegionSize 1;
mass 4.984e-03;
momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
orientation //Oriented at 0 Degrees
(
   1 0 0
   0 1 0
   0 0 1
);
velocity (0 0 0);
acceleration (0 0 0);
angularMomentum (0 0 0);
rhoName rhoInf;
rhoInf 1000;
report on;
accelerationRelaxation 0.5;
accelerationDamping 1;
value uniform (0 0 0);
solver
{
 type symplectic;
}
constraints
{
 zAxis
 {
  sixDoFRigidBodyMotionConstraint axis;
  axis           (0 0 1);
 }
}
restraints
{
 translationDamper
 {
```

```
      sixDoFRigidBodyMotionRestraint linearDamper;
      coeff           0.001;
     }
    }
   }
  }
}

FLUID_LAM_11
{
 solidBodyMotionFunction  nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin (0.1158842 0.2007224 0); // Initial Center of
     ↪ Rotation of Nested Body
  initialTCP (0.1281033 0.2218814 0); // Initial Torque Calculation Point
     ↪  of Nested Body
  nestedAxis (0 0 1); // Axis of Rotation of Nested Body
  patches (WALL_LAM_11);
  derivedTorque 2;
  centreOfMass (0.1158842 0.2007224 0);
  centreOfRotation (0.1158842 0.2007224 0); // To establish the distance
     ↪ between centerOfMass and centerOfRotation (MI calculation)

  initialTrailingEdge (0.09678526 0.1676370 0);
  torqueCalibrate 1;
  momentOfInertiaComparative 5.445e-06;

  //Setting Constraint Conditions
  stopConstraint 1; // Enter or exit constraint loop
  proximityBoundaryPos 360;
  proximityBoundaryNeg  270;
  proximityRegionSize 1;
  mass 4.984e-03;
  momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
  orientation //Oriented at 0 Degrees
  (
     1 0 0
     0 1 0
     0 0 1
  );
  velocity (0 0 0);
  acceleration (0 0 0);
  angularMomentum (0 0 0);
  rhoName rhoInf;
  rhoInf 1000;
  report on;
  accelerationRelaxation 0.5;
  accelerationDamping 1;
  value uniform (0 0 0);
  solver
  {
   type symplectic;
  }
```

```
    constraints
    {
     zAxis
     {
      sixDoFRigidBodyMotionConstraint axis;
      axis            (0 0 1);
     }
    }
    restraints
    {
     translationDamper
     {
      sixDoFRigidBodyMotionRestraint linearDamper;
      coeff           0.001;
     }
    }
   }
}

FLUID_LAM_12
{
 solidBodyMotionFunction   nestedRotatingNonSmoothFSIMotion;
 nestedRotatingNonSmoothFSIMotionCoeffs
 {
  outerOrigin (0 0 0); // Center of Revolution of Nested Body
  outerOmega -3.769911184; // Revolution Speed of Nested Origin (rad/s)

  initialNestedOrigin (0.1561469 0.270464 0); // Initial Center of
     ↪ Rotation of Nested Body
  initialTCP (0.1683674 0.2916209 0); // Initial Torque Calculation Point
     ↪  of Nested Body
  nestedAxis (0 0 1); // Axis of Rotation of Nested Body
  patches (WALL_LAM_12);
  derivedTorque 2;
  centreOfMass (0.1561469 0.270464 0);
  centreOfRotation (0.1561469 0.270464 0); // To establish the distance
     ↪ between centerOfMass and centerOfRotation (MI calculation)

  initialTrailingEdge (0.1370494 0.2373766 0);
  torqueCalibrate 1;
  momentOfInertiaComparative 5.445e-06;

  //Setting Constraint Conditions
  stopConstraint 1; // Enter or exit constraint loop
  proximityBoundaryPos 360;
  proximityBoundaryNeg  270;
  proximityRegionSize 1;
  mass 4.984e-03;
  momentOfInertia (2.471e-06 2.471e-06 2.471e-06);
  orientation //Oriented at 0 Degrees
  (
     1 0 0
     0 1 0
     0 0 1
  );
  velocity (0 0 0);
```

```
    acceleration (0 0 0);
    angularMomentum (0 0 0);
    rhoName rhoInf;
    rhoInf 1000;
    report on;
    accelerationRelaxation 0.5;
    accelerationDamping 1;
    value uniform (0 0 0);
    solver
    {
     type symplectic;
    }
    constraints
    {
     zAxis
     {
      sixDoFRigidBodyMotionConstraint axis;
      axis           (0 0 1);
     }
    }
    restraints
    {
     translationDamper
     {
      sixDoFRigidBodyMotionRestraint linearDamper;
      coeff          0.001;
     }
    }
   }
  }

  FLUID_ROTOR
  {
   solidBodyMotionFunction  rotatingMotionVin;
   rotatingMotionVinCoeffs
   {
    origin (0 0 0);
    axis (0 0 1);
    omega -3.769911184; // rad/s
   }
  }
}


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

Bibliography

[1] Maritime Research Institute, Netherlands (MARIN), *Report no: 28537-1-bt : Oryon water mill power generation tests in calm water,* (2016), Confidential Test Report.

[2] E. Lefrancois and J.-P. Boufflet, *An introduction to fluid structure interaction: Application to the piston problem,* SIAM (2010).

[3] J. D. Anderson Jr., *Computational Fluid Dynamics: The Basics with Aplications* (McGraw-Hill. Inc., 1995).

[4] Y. Bazilevs, K. Takizawa, and T. E.Tezduyar, *Computational Fluid-Structure Interaction: Methods and Applications* (John Wiley & Sons, Ltd., 2013).

[5] Gerhard A. Holzapfel, *Nonlinear Solid Mechanics : A Continuum Approach for Engineering* (John Wiley and Sons, Ltd., 2000).

[6] J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics* (Springer, 2002).

[7] *Systems Engineering Fundamentals* (Defense Acquisition University Press, 2001).

[8] Herbert Goldstein, *Classical Mechanics* (Addison–Wesley, 1980).

[9] *ANSYS ICEM CFD Tutorial Manual*, ANSYS Inc.

[10] J. F. Thompson, B. K. Soni, and N. P. Weatherill, *Handbook of Grid Generation* (CRC Press, 1998).

[11] Tomislav Maric and Jens Hopken and Kyle Mooney, *The OpenFOAM Technology Primer* (McGraw-Hill. Inc., 2014).

[12] J. D. Anderson Jr., *Fundamentals of Aerodynamics* (McGraw-Hill. Inc., 2014).