



Swansea University
Prifysgol Abertawe

**Numerical Simulation of Selected Two-Dimensional
and Three-Dimensional Fluid-Structure Interaction
Problems Using OpenFOAM Technology**

Maimouna Said Al Manthari

*Zienkiewicz Centre for Computational Engineering, College of Engineering,
Swansea University, Swansea, Bay Campus, Swansea, SA1 8EN*

Thesis submitted to Swansea University in fulfillment of the requirements
for the Degree of Doctor of Philosophy

June 2018

Summary

Fluid-structure interaction (FSI) problems are increasing in various engineering fields. In this thesis, different cases of FSI in two- and three-dimensions (2D and 3D) are simulated using OpenFOAM and foam-extend. These packages have been used to create a coupling between fluid and solid.

The vortex-induced vibration (VIV) phenomenon of flow past a circular cylinder is studied using PIMPLE algorithm for pressure-velocity coupling. This VIV study is restricted to incompressible flow simulation at a Reynolds number (Re) of 100. The changes of drag and lift coefficient values depend on the study case and the spring-mass-damper system for the flow past a free oscillatory cylinder. The free vibrating cylinder examined in one-degree-of-freedom (1DOF) and two-degrees-of-freedom (2DOF) systems with linear damping and spring properties. Both will affect the behaviour of the cylinder within the flow with some noticeable differences. The response time of the cylinder and the drag coefficient are the most affected by the spring and damper.

Besides the vortex-induced vibration test cases, the two-dimensional and three-dimensional fluid-structure interaction benchmarking is also studied. A partitioned solution method for strongly coupled solver with independent fluid and solid meshes for transient simulation has been applied. The fluid domain dynamics is governed by the incompressible Navier-Stokes equations; however, the structural field is described by the nonlinear elastodynamic equations. Fluid and solid domains are discretised by finite volume method (FVM) in space and time.

A strong coupling scheme for partitioned analysis of the thin-walled shell structure exposed to wind-induced vibration (WIV) is presented. The achievement of the 3D membrane roof coupling scheme is studied by applying the 2D model. Additionally, numerical models for the slender shell structures coupling and the 3D flows indicate possible applications of the presented work. The computational fluid dynamics (CFD) simulation results revealed that even the flow is considered as a laminar, turbulence modelling or more refined meshes should be used to capture the generation and release of vortices.

A partitioned solution procedure for FSI problems in the building aeroelasticity area is also studied. An illustrative real-world model on the coupled behaviour of membrane structure under wind flow influence is given. A four-point tent subjected to wind motion is a typical application of this work applying with various physical factors that are a necessity for the thin membrane structure. The fluid domain is described by the incompressible Navier-Stokes equations at a Reynolds number of $Re = 3,750$. However, the motion of the solid field is modeled by total Lagrangian strategy for nonlinear elastic deformation.

The FSI simulation, particularly 3D problems require in very long calculation time. Some limitations of the FSI solver in foam-extend package called `fsiFoam` is discussed.

All solvers that used in this thesis are considered to be applied to a wide use of the implementation of FSI models, despite some problems in parallelisation, particularly in the latest FSI solver version. The analysis results are presented to demonstrate accuracy, convergence, and stability.

Declaration and Statements

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed (candidate)

Date

STATEMENT 1

This thesis is the result of my own work and investigation, except where otherwise stated. Where correction services have been used, the extent and nature of the correction is clearly marked in a footnote(s).

Other sources have been acknowledged by giving explicit references. A bibliography is appended.

Signed (candidate)

Date

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed (candidate)

Date

NB: Candidates on whose behalf a bar on access has been approved by the University (see Note 7), should use the following version of Statement 2.

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loans **after expiry of a bar on access approved by the Swansea University.**

Signed (candidate)

Date

Table of Contents

Acknowledgements.....	ix
List of Figures.....	x
List of Table.....	xiii
Abbreviations.....	xiv
Nomenclature.....	xvi
1. Introduction.....	1
1.1 Introduction to Fluid-Structure Interaction.....	1
1.2 Introduction to OpenFOAM	2
1.3 Flow-Induced Vibration.....	3
1.4 Fluid-Flexible Structure Interaction.....	6
1.5 Fluid-Membrane Structure Interaction	8
1.6 Aim of the Thesis.....	11
1.7 Layout of the Thesis.....	14
2. Equations, Discretisation and Methodologies.....	16
2.1 Governing Equations	16
2.1.1 Flow Equations ..	16

2.1.2 Structural Equations.....	17
2.2 Introduction to Finite Volume Discretisation	18
2.2.1 Discretisation of the Computational Domain	18
2.2.2 Transport Equation Discretisation	19
2.2.3 Face interpolation	21
2.2.4 Discretisation of Spatial Terms.....	22
2.2.5 Temporal Discretisation.....	26
2.2.6 Pressure-Velocity Coupling.....	27
2.3 Fluid-Structure Interaction Problems in OpenFOAM and Foam-Extend.....	37
2.4 Finite Volume Method for Fluid-Structure Interaction with Large Structural Displacements	39
2.5 OpenFOAM Library for Fluid-Structure Interaction.....	40
2.6 Partitioned Solver for Strongly Coupled Fluid-Structure Interaction.....	42
2.7 Arbitrary Lagrangian Eulerian Mapping	44
2.8 Fluid-Structure Coupling and Boundary Conditions	46
2.9 Numerical Method in FSI Library	47
2.10 Implicit Coupling Approach with Second-Order Predictor	48
2.11 Boundary Conditions Definitions in OpenFOAM.....	50
2.12 Chapter Summary	52

3. A Flow past a Two-Dimensional Cylinder and Vortex-Induced Vibration.....	54
3.1 Introduction to Flow around a Cylindrical Structure	54
3.2 Flow Regimes	55
3.3 Vortex Shedding	57
3.4 Hydrodynamic Forces	59
3.5 Vortex-Induced Vibration.....	61
3.6 Dynamics of One Degree of Freedom System and Solution to Vibration Equation	62
3.7 Reduced Velocity.....	63
3.8 Mesh Generation in OpenFOAM	65
3.9 Test Cases	65
3.9.1 2D Example of Flow past a Stationary Circular Cylinder	65
3.9.2 Vortex-Induced Vibration of the Circular Cylinder	75
3.9.3 Non-Resonance VIV Schematic Description	79
3.9.4 Forced Oscillation Cylindrical Structure	92
3.10 Chapter Summary	96
4. Two- and Three-Dimensional Benchmarking Models.....	99
4.1 2D Example of Flow-Induced Oscillations of a Flexible Tail behind a Block.....	99
4.1.1 Geometry and Boundary Conditions	99
4.1.2 Problem Definition	100
4.1.3 Mesh Generation.....	101

4.1.4 Spatial and Temporal Discretisation.....	101
4.1.5 Post-Processing, Results and Discussions	103
4.2 3D Elastic Cantilever Plate Attached to a Solid Block.....	106
4.2.1 Model Geometry and Boundary Conditions	108
4.2.2 Mesh Generation.....	109
4.2.3 Case Implementation	112
4.2.4 Simulation Problems.....	113
4.2.5 Results and Discussions.....	113
4.3 2D Model of Hanging Membrane Roof Subjected to the Wind	118
4.3.1 Geometry, Material Properties, and Boundary Conditions.....	118
4.3.2 Mesh Generation.....	119
4.3.3 Post-Processing, Results and Discussion.....	119
4.4 3D Membrane Roof Benchmark.....	121
4.4.1 Model Geometry	123
4.4.2 Material Properties.....	123
4.4.3 Mesh Generation.....	123
4.4.4 Discretisation, Boundary Conditions, and Simulation Results.....	124
4.5 Chapter Summary	130
5. Wind-Membrane Interaction.....	132
5.1 Numerical Example of Four-Point Tent Structure Subjected to the Wind.....	132

5.1.1 Meshing Set-Up	132
5.1.2 Important Logic Notes	134
5.1.3 Simulation Set-Up.....	135
5.1.4 Tent Case Limitations	138
5.2 Learning from Membrane Structure Failure Cases.....	143
5.3 Chapter Summary	144
6. Conclusions and Future Research.....	147
6.1 Achievements Summary	147
6.2 Suggestions for Future Work.....	151
Bibliography.....	152
Appendices.....	174
Appendix 2.A: OpenFOAM Computational Pointers.....	174
Appendix 3.A: Velocity (U) Boundary Conditions of the Circular Cylinder Case ..	177
Appendix 3.B: Pressure (p) Boundary Conditions of the Circular Cylinder Case ...	179
Appendix 3.C: controlDict File of the Circular Cylinder Case	181
Appendix 3.D: fvScheme File of the Circular Cylinder Case	185
Appendix 3.E: fvSolution File of the Circular Cylinder Case	187
Appendix 3.F: dynamicMeshDict File.....	190
Appendix 3.G: Free Vibration Case - Scenario 1	191
Appendix 3.H: Free Vibration Case - Scenario 2	196

Appendix 3.I: Free Vibration Case - Scenario 3.....	201
Appendix 3.J: Forced Vibration Case.....	206
Appendix 4.A: Mesh Generation- blockMeshDict of the 3D Elastic Cantilever Plate Attached to a Solid Block Case	208
Appendix 4.B: Implementation structure of 3D Elastic Cantilever Plate Attached to a solid Block.....	220
Appendix 4.C: oscillatingInlet of the 3D Elastic Cantilever Plate Attached to a Solid Block Case	222
Appendix 4.D: changeDictionaryDict of the 3D Elastic Cantilever Plate Attached to a Solid Block Case.....	224
Appendix 4.E: mappingFields File	225
Appendix 4.F: Allclean Script File	227
Appendix 4.G: Allrun Script File	228
Appendix 4.H: controlDict File	229
Appendix 5.A: Tent Case Structure	233
Appendix 5.B: Steps of Meshing the Fluid Region	234
Appendix 5.C: Steps of Meshing the Solid Region	243
Appendix 5.D: Allrun Script File	248

Acknowledgements

I would like to thank Prof. Perumal Nithiarasu, my supervisor, for his invaluable guidance, support and encouragement during all stages of this work.

I would like to thank the Zienkiewicz Centre for Computational Engineering, of Swansea University as a whole for providing a very supportive and encouraging research environment.

I would like to thank my husband Sulaiman sincerely for his support, understanding, and patience, and for all the memorable days, which we spent in the beautiful countries of United Kingdom.

I would like to thank my children, Aljuman, Mohammed, and Alreem for their support and making life in Swansea more beautiful.

I am very grateful to my parents, brothers, and sisters for their unconditional and constant support.

I would also like to send my special thanks to Bruno Santos for always guiding me with the OpenFOAM and always tries his best to answer all of my questions.

Finally, the financial support, which I received from the Ministry of Higher Education of Oman, is gratefully acknowledged.

List of Figures

Figure 2.1: Control volume.....	20
Figure 2.2: Storing variables options, (a) monolithic method and (b) partitioned method.....	22
Figure 2.3: Face interpolation.....	25
Figure 2.4: Arbitrary 2D finite volume grid with notations.....	29
Figure 2.5: Simple flow field by steady conditions.....	30
Figure 2.6: Values at faces determined from the values at the cell centers.....	31
Figure 2.7: Solving process of FSI using partitioned approach with weak and strong coupling.....	38
Figure 2.8: Flowchart of algorithm of fsiFoam solver.....	43
Figure 2.9: Partitioned solver for strongly coupled nonlinear fluid-structure interaction flowchart.....	51
Figure 3.1: Vortex shedding phenomenon behind a cylindrical structure	54
Figure 3.2: The shear layer of the flow over a cylinder	58
Figure 3.3: Strouhal number – Reynolds number relationship for a circular cylinder... ..	60
Figure 3.4: One-degree-of-freedom system models in cylinder	62
Figure 3.5: Feng’s experiment model	64
Figure 3.6: Geometry and boundary conditions of flow over a cylinder	67
Figure 3.7: Circular cylinder mesh visualized in ParaView	67
Figure 3.8: Schematic illustration for boundaries in laminar flow	69
Figure 3.9: Velocity profile for laminar flow with $Re = 100$ in ParaView	73
Figure 3.10: Pressure profile for laminar flow with $Re = 100$ in ParaView	73
Figure 3.11: Drag coefficient of 2D cylinder at laminar flow of $Re = 100$	74
Figure 3.12: Lift coefficient of 2D cylinder at laminar flow of $Re = 100$	75
Figure 3.13: Drag coefficient of laminar flow at $Re = 100$, $Re = 200$, and $Re = 1000$	76
Figure 3.14: Lift coefficient of laminar flow at $Re = 100$, $Re = 200$, and $Re = 1000$	77
Figure 3.15: Directories and files for vortex-induced vibration case	77
Figure 3.16: Schematic for the VIV non-resonance case	80
Figure 3.17: Geometry and flow conditions of case with 4-springs	81
Figure 3.18: Schematic of 2-springs and 2-dampers	81

Figure 3.19: (a) Drag and (b) lift coefficients for 1DOF and 2DOF	83
Figure 3.20: (a) Drag and (b) lift coefficients for 1DOF system	85
Figure 3.21: (a) Drag and (b) lift coefficients for 2DOF system	88
Figure 3.22: (a) Drag and (b) lift coefficients for reduced damping case for 1DOF and 2DOF.....	90
Figure 3.23: (a) Drag and (b) lift coefficients for reduced spring stiffness case for 1DOF and 2DOF	91
Figure 3.24: (a) Drag and (b) lift coefficients for 2DOF systems	93
Figure 3.25: Force coefficient of forced oscillation cylinder case at laminar flow with $Re=$ 100	95
Figure 3.26 Vertical displacement and lift coefficient time histories of the forced oscillation case	96
Figure 4.1: Computational domain and boundary conditions of tail behind block.....	100
Figure 4.2: 2D tail attached to solid support computational domain for refined mesh at (a) $t = 0s$ and (b) $t = 6s$	102
Figure 4.3: Oscillation of a tail attached to a solid block at $t = 6s$: (a) velocity and (b) pressure/density.....	104
Figure 4.4: Comparison results on the tip displacement, (a) coarse mesh (b) fine mesh.....	107
Figure 4.5: 3D flexible plate model: geometry and boundary conditions.....	108
Figure 4.6: 3D elastic plate case block levels.....	110
Figure 4.7: 3D tail attached to solid block mesh in ParaView.....	110
Figure 4.8: Meshing representation of the plate and the solid block in ParaView.....	111
Figure 4.9: 3D plate in wind: structural motion and stream-tube snapshots for some simulation time.....	114
Figure 4.10: Perpendicular displacement in meter of flexible plate over simulation time in second for points $A = (0.10, 0.055, 0.07)$, $B = (0.10, 0.055, 0.055)$ and $C =$ $(0.10, 0.055, 0.04)$	116
Figure 4.11: 2D membrane roof model: geometry, material properties, and boundary conditions.....	118
Figure 4.12: Flow velocity in the 2D membrane roof.....	120
Figure 4.13: Meshing details: (a) coarse mesh (b) fine mesh.....	120
Figure 4.14: Solution for the displacement at the interface: (a) coarse (b) fine.....	121

Figure 4.15: The maximum inflow speed for the 2D membrane roof model.....	121
Figure 4.16: Membrane roof deformation in x-direction.....	122
Figure 4.17 Membrane roof deformation in y-direction.....	122
Figure 4.18: Geometrical properties and boundary conditions of 3D membrane roof...	125
Figure 4.19: Meshing results for 3D membrane roof model.....	125
Figure 4.20: Maximum flow velocity for 3D membrane roof benchmark Figure 4.23: 3D flexible membrane roof deformation over z-direction.....	126
Figure 4.21: 3D membrane roof: (a) spatial (b) temporal inflow boundary conditions variations.....	127
Figure 4.22: 3D flexible membrane roof deformation over z-direction.....	128
Figure 4.23: Deformed structure of the 3D membrane roof case in z-direction.....	129
Figure 5.1: Geometry and dimension of the four-point tent structure.....	133
Figure 5.2: Maximum inflow velocity over simulation time.....	137
Figure 5.3: Response of the structure at different times with counter plot of the displacement dz in meter.....	139
Figure 5.4: Deformation results along z-direction of four-point tent case in 6s simulation time.....	140

List of Tables

Table 3.1: Flow regimes around a smooth circular cylinder in steady current.....	56
Table 3.2: Comparison of force coefficients from the simulation results with other studies.....	74
Table 3.3: Flow regimes of forced vibration cylinder case using ParaView.....	97
Table 4.1: Physical properties of the tail behind block model.....	101
Table 4.2: Comparison present work results with other literature for the elastic tail attached to the solid block.....	106
Table 4.3: Material properties of the 3D flexible plate attached to solid block.....	109
Table 4.4: Comparing of fluid mesh properties.....	110
Table 4.5: Comparing of structure mesh properties.....	112
Table 4.6: Results comparison for the 3D flexible plate attached to solid block model.....	117
Table 4.7: 2D hanging roof meshing details.....	119
Table 4.8: Material properties of the 3D membrane roof model.....	123
Table 5.1: Material properties of the 3D four-point tent case.....	136

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
ADI	Alternating Direction Implicit
ALE	Arbitrary Lagrangian-Eulerian
BGS	Block Gauss-Seidel
BN	Block Newton
CCM	Computational Continuum Mechanics
CFD	Computational Fluid Dynamics
CM	Control Mass
Co	Courant number
CV	Control Volume
CWE	Computational Wind Engineering
DES	Detached-Eddy Simulation
DIC	Diagonal-based Incomplete Cholesky preconditioner
DILU	Diagonal Incomplete Cholesky preconditioner for asymmetric matrices.
DOF	Degree-of-Freedom
GAMG	Generalized Geometric-Algebraic Multi-Grid or Geometric Agglomerated Algebraic Multi-Grid
GGI	General Grid Interface
FDIC	Faster version of the DIC preconditioner
FEM	Finite Element Method
FIV	Flow-Induced Vibration

FOAM	Field Operation and Manipulation
FSI	Fluid-Structure Interaction
FV	Finite Volume
FVM	Finite Volume Method
ICC	Incomplete Cholesky Preconditioned Conjugate Gradient
IQN-ILS	Interface Quasi-Newton iterations based on the Inverse Least Squares approximation of the Jacobian
LES	Large Eddy Simulation
MPI	Message Passing Interface
NR	Newton-Raphson
PbiCG	Preconditioned Biconjugate Gradient
PCG	Preconditioned Conjugate Gradient
PDEs	Partial Differential Equations
PIMPLE	PISO and SIMPLE algorithms
PISO	Pressure Implicit with Splitting of Operator
PVC	Polyvinylchloride
Re	Reynolds number
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
St	Strouhal Number
STL	Stereolithography
VIV	Vortex-Induced Vibration
WIV	Wind-Induced Vibration

Nomenclatures

Latin Characters

\mathbf{a}	General vector property
a_K	Matrix coefficient corresponding to neighbouring cell K
a_P	Central coefficient
A/D	Oscillating amplitude constant
C_D	Drag coefficient
\overline{C}_D	Mean drag coefficient
C_L	Lift coefficient
Co	Courant number
c	Linear-viscous damping
D	Cylinder diameter
$d\mathbf{S}$	Infinitesimal surface element with the normal of the associated outward points on ∂V
$\mathbf{d}_{m,f}$	Fluid domain displacement
$\mathbf{d}_s^{\Gamma_0}$	The solid displacement that formed by fluid-solid interface from the fluid domain displacement
$\mathbf{d}_{i,j}^{\Gamma}$	Displacement interface at the time-step i and the iteration j
$\tilde{\mathbf{d}}^{\Gamma}$	Interface displacement predictor
E_s	Young Modulus
\mathbf{F}_b	Body force
F_D	Drag force

F_L	Lift force
F_r	Frequency ratio
f	control volume faces
f_n	Cylinder frequency
f_v	Vortex shedding frequency
f_x	The interpolation factor that is defined as the distance ratio between \overline{fK} and \overline{PK}
\mathbf{I}	Identity matrix
i,j	Unit vectors
K	Neighbour cell centre
k	Spring stiffness
M	Mesh motion solver
m	Cylinder mass
P	Computational point placed in the centre of the control volume
p	Pressure
Q	The mass flux through the face
Re	Reynolds number
$\mathbf{r}_{i+1,j+1}^\Gamma$	Residual of the interface
\mathbf{S}	Outward-pointing face area vector
St	Strouhal number
\mathbf{S}_f	Face area vector
S_\emptyset	Source term
Sp	Linear part of the source term
Su	Constant part of the source term

T_v	Vortex shedding period
t	Time
tr	Tensor trace
\mathbf{U}, \mathbf{U}_f	Flow velocity
\mathbf{U}_s	Solid displacement
u_∞	Stream-wise velocity
V	Volume
∂V	The closed surface that bounding the volume V
V_P	control volume around the point P
V_R	Reduced velocity
\mathbf{x}	Position vector
y	Vertical displacement of the mass centre of a moving cylinder

Greek Characters

α_p	Pressure based under-relaxation factor
α_U	Under-relaxation factor
$\alpha_{i+1,j+1}$	Relaxation parameter
Γ	Diffusion coefficient
Γ_0	Fluid-solid interface
Δt	The time-step
Δx	The mesh size
ε	Strain tensor
λ_s, μ_s	Lame' constants

μ_f	Dynamic viscosity
ν_f	Kinematic viscosity ($\nu_f = \mu_f / \rho_f$)
ρ	Density
ρ_f	Fluid density
ρ_s	Solid density
σ_s	Cauchy stress tensor
τ	Equilibrium traction
ν_s	Poisson's ratio
ϕ	Scalar variable
$\Omega_{f,0}$	Internal fluid reference

Chapter 1

Introduction

1.1 Introduction to Fluid-Structure Interaction

Fluid-structure interaction (FSI) is very popular across different fields of engineering such as civil, mechanical, biomedical, and aerospace engineering. In the following paragraphs, some physical problems involving FSI are listed.

Civil engineering: The wind-induced vibration over thin-walled structures such as bridges, membrane structures, and lattice towers often experience large deformation. Some fatal failures can sometimes occur due to the oscillation at the natural frequency. The 1940 Tacoma Narrows suspension bridge in the USA is considered a critical example of the catastrophic failure of wind-induced vibration on unsuitable design. It was caused by rotational and large transverse oscillations, which caused to collapse [1].

Mechanical engineering: The applications for mechanical engineering include engineering, physics, the principles of material science, mechanical design, manufacture, analyse, and maintain mechanical systems. In the design stage, some internal flow models required a very accurate simulation in order to achieve a very high performance.

Biomedical engineering: Blood flow is one example of FSI in the human body. The blood flow circulation through the heart, arteries, and veins shows the fluid motion with moving boundaries. The design of medical devices such as micro-pumps depends on an understanding of the fluid flow mechanisms.

Aerospace engineering: The vortex-induced vibration study and flutter are crucial in understanding aircraft stability in flight.

Research interests in the field of FSI range from linear to the non-linear, and steady-state to transient interactions. The underlying FSI concept is that when a flow pass a body, it causes exchange of energy between the fluid and the structure, which in turn causes significant changes in the physical behaviour of both. The physical changes are evidenced from explosive structural loading, acoustics, liquid sloshing in open and closed vessels,

fluid-induced vibrations, building wind loads, bridge deck, and aerodynamic fluttering, etc. [2]. A review of FSI is important because it enhances knowledge in engineering design against wave stress effects on solids that may result in large deformations and damages [3]. The extent and type of failures depend on velocity, density and size of fluid. From an engineering perspective, the effects of wave stress greatly influence the solid design and material selection. For instance, the choice of materials for turbines, pumps and piping prone to cavitation requires an understanding of pressure fluctuations during design and material strength to avoid collapse, erosion of movable parts, and structural damages. Recent studies [4-5] show that more work is still needed in computational FSI.

This thesis investigates fluid-structure interaction in two- and three-dimensional problems. Body geometry is often considered trivial, but fluid flow over a bluff body remains a major challenging problem in a wide range of engineering applications. For instance, cylindrical geometry is responsible for complex flow phenomena that are largely Reynolds number (Re) dependent and generates unsteady vortex shedding (wake), oscillation separation and stagnation points, and turbulence transitions of different flow regimes that are Re -dependent [6].

In the last decade, many numerical and computational techniques for FSI simulation have been developed. This work focuses on the implementation of two-dimensional and three-dimensional FSI problems, using open-source software techniques; namely OpenFOAM [7].

1.2 Introduction to OpenFOAM

The software used in the current study is OpenFOAM (Open Field Operation and Manipulation). OpenFOAM is an open source object-oriented program for Computational Fluid Dynamics (CFD) package. OpenFOAM libraries are written in C++ language and used to build the executable files. The applications of these libraries are divided into two groups: solvers and utilities. Solvers provide the numeric and solutions of the mechanical problems, and utilities work with other tasks that include meshing, visualisation, and data manipulation [7].

Additionally, it has a general purpose of finite-volume simulation structure organized by OpenCFD Ltd at an ESI Group [7], and it is distributed by OpenFOAM Foundation in GNU General Public License. It usually covers a wide range of structures in geometrical and physical modelling, and includes laminar and turbulence model varieties and several functionalities of mesh modification.

The CFD tool is designed to solve continuum mechanics problems that are based on the finite volume method (FVM). The FV discretisation approach is the essential procedure for solving partial differential equations (PDEs) governing both the continuity and momentum conservation laws. The continuous PDE variables must be transformed into algebraic equation sets using finite volume procedure in order to describe the numerical or discrete solution of the flow governing equations [8-9].

The parallelisation uses only MPI as it is based upon the domain decomposition strategy. Both the computational structural mechanics (CSM) and the computational fluid dynamics (CFD) are implemented in OpenFOAM.

In addition, foam-extend [10] version of OpenFOAM will be used in this thesis. It has a branch of solid mechanics with implemented FSI and structural solvers along with the original OpenFOAM version that is released for meshing and post-processing.

The main purpose of the Extend-Project is to integrate contributions from OpenFOAM users and developers. In the late 1990s, early FSI work in OpenFOAM was carried out at the Imperial College London. However, FSI solver in OpenFOAM became much easier with the registration of the mesh-based field and multi-zonal support introduction.

1.3 Flow-Induced Vibration

Vortex-induced vibration (VIV) is a motion induced on a bluff body facing an external fluid flow by periodical irregularities on this flow. Vortex-induced vibration phenomenon of bluff bodies has been investigated for a long time because of its importance in both engineering applications and academic research. In order to gain a better understanding of this phenomenon, numerous computational and experimental studies have been

accomplished on this fluid-structure interaction problem covering from rigid cylinder(s) to elastic cylinder(s).

Vortex-induced vibration for elastically or rigidly mounted circular cylinder in cross flow has been subject to different investigations over the past years. The two scenarios are flow-induced forces caused by vortices, and forced cylinder vibrations. These will be the subjects of investigation.

Anagnostopoulos [11-12] focused on investigating the VIV by using a numerical method. He studied the VIV behaviour in the 2D circular cylinder with two-degrees-of-freedom at $Re = 200$. In 2D, the space-time finite element approach is used to solve the incompressible Navier-Stokes equations, while the explicit integral method is used to solve the motion of the cylinder. The VIV results demonstrated that there were different trends of lift and drag coefficients at low mass damping.

Placzek *et al.* [13] studied the numerical simulation of free and forced vibrating cylinder at $Re = 100$. A preliminary work of the study has exposed that for a stationary cylinder to check the characteristics of the wake. The simulation of the forced oscillating cylinder have been studied in the cross-flow direction to analyse different phenomena. The simulation of the forced oscillating cylinder is characterised by the amplitude A and by the frequency ratio ($F_r = f_n / f_v$), where f_n is the cylinder frequency and f_v is the vortex shedding frequency. Then, the free vibrating cylinder simulation carried out. The cylinder vibration is excited by the vortex shedding in the wake. Thus, in order to observe different behaviours, the frequency and amplitude are studied in a wide range of reduced velocities.

In a different experimental study, Govardhan and Williamson [14] investigated response profiles of free oscillating in terms of lift forces and near wake vorticity on a cylinder. Deductions from this study further contributed towards an understanding of FSI and how body motion is controlled, allowing wake response of the motion to be evaluated separately.

In another numerical study, Williamson and Govardhan [29] presented new vortex wake states in the map of vortex modes framework assembled from studies of a forced vibration that caused a free vibration. The discussion and analysis focused on the relationship between forced and free vibrations, and the relevance of the elastically mounted cylinder flow to more complex systems.

A freely oscillating cylinder in the inline and cross-flow directions display altered shedding patterns pointing to the viewpoint that variations in overall structural response depend on the cylinder vibration directions [15-17,30]. The findings by Williamson and Jauvtis [15] strongly indicated possible contributions of high harmonics to the lift force. An investigation by Dahl *et al.* [18-19] using phase differences between inline and cross-flow oscillations revealed a strong influence on cylinder trajectory regularity.

In the forced oscillating cylinder case, upstream vortices are responsible for the induced vibration by simply applying mechanical force to induce cylinder vibration [20-21]. On the other hand, imposed oscillation frequency dominates a perfectly rigid cylinder exposed to forced vibration [22].

The forced oscillatory case uses analogous, but with different sets of parameters. For example, a forced oscillating cylinder has oscillating amplitude constant (A/D where D is a cylinder diameter), and its frequency (f_o) vary at points where the ratio of $f_o/f_s = 1$, where f_o represents the natural frequency and f_s shows the oscillation frequency. Nonetheless, predicting flow-induced motion using forced oscillation still has several unresolved challenges. The near wake structure and forces acting on a cylinder under forced oscillation has been researched extensively [13,24-26].

The wake states for the forced oscillatory cylinder at low amplitude exhibits low and high frequency depending on force properties and wake structures. As the oscillation frequency passes through the natural Von Karman frequency (f_o), a transition occurs between two different wake modes namely, the low and high frequency. Transitions between low and high frequency states cause a “jump” in both vortex lift forces acting on the cylinder and the overall amplitude phase [23,27].

At higher oscillation amplitudes, a cylinder exposed to forced oscillations exhibits a third wake state between low and high frequencies. Few studies had explored the existence of a forced wake state at high amplitude until recently. This third forced wake which is also known as “intermediate”, occurs at $A/D = 0.5$ to 0.6 (oscillation amplitudes). Despite the limited availability of literature on intermediate branch existence, the large jumps occurring at different phase transitions explains the vortex shedding problem in the forced oscillating cylinder at higher mass-damping [28].

Conventionally, experiments using forced oscillation to investigate a broad array of engineering problems assumed flow-induced oscillation could be represented adequately by the sinusoidal oscillations, at a constant frequency and oscillation amplitude. However, failure to establish the link between freely oscillating and forced oscillatory cases rendered the historical approaches unsuitable in predicting flow-induced vibration conclusively [31].

Moreover, a large body of literature on the vortex-induced vibration field covering experimental and numerical investigations, from one to multi-degrees of freedom motion, flexible and rigid motions, VIV phenomenon in water and air, are available [32-35].

Zhou *et.al* [36] studied a flow past a flexible circular cylinder at $Re = 200$. The motion of the cylinder is modeled by a system of spring-damper-mass. One-degree-of-freedom and two-degrees-of-freedom models were examined. The results obtained were compared with previous computational and experimental results. The study results show that the VIV for 1DOF system show only a good qualitative agreement with the 2DOF system.

The study by Li *et al.* [37] focused on studying the characteristics of VIV of a flexible circular cylinder with one- and two-degrees-of-freedom models at a Reynolds number of 200. The results showed that there were similar trends between both models.

Moreover, the experimental results of 1DOF at a range of $Re = [90, 150]$ were captured by Anagnostopoulos and Bearman [38]. Additionally, the same experiment was conducted by Kalak and Williamson [39]. For large density ratios, the results showed that the vortex shedding frequency (f_v) for the static cylinder and the cylinder frequency (f_n) at resonance were very close ($f_n / f_v = 1$). However, this result was not applicable to small density ratios.

1.4 Fluid-Flexible Structure Interaction

Fluid-structure interaction (FSI) is defined as the interaction of some deformable or moveable structures with a surrounding or internal fluid. Studying the interaction between flow and flexible structures is an important stage towards the understanding of several physical and engineering problems. Simulations of such models are computationally

challenging. Thus, attempts to develop more efficient and more sophisticated numerical FSI models are relevant.

Many interesting FSI models are associated with fluid-structure coupling. Modelling the coupled fluid and solid dynamics is a very complex topic and has attracted a lot of attention. Coupling strategy is central in numerical analysis of flow over the bluff body in FSI applications. Most commercially available computational software programs use solvers in simulating fluid and solid interactions by imposing boundary conditions along their interface. The velocity or displacement by solid domain serves as a fluid solver boundary condition for the solid domain, while fluid domain relies on generated pressure along the interface as the force boundary condition by the solid solver [40]. Governing equations in each scenario cover mathematical variables such as density, pressure, velocity, stress, etc.

The variables are classified into coupled and uncoupled variables. Coupled variables refer to interface parameters that can be distinctively determined from both fluid and solid. On the other hand, uncoupled variables cannot be distinctively determined at fluid-solid interface hence require interfacial prediction. The main challenge is setting internal boundary conditions for uncoupled variables. For example, for two-dimensional FSI problems, normal pressure and velocity in the fluid is coupled with normal stress and velocity of solid because it is assumed to be continuous across the interface [3]. However, prediction of uncoupled variables such as tangential velocity and density of fluid, and shear stress and tangential velocity of solid, is necessary in order to set internal boundary conditions.

Generally, FSI problems define the strongly coupled fluid flow and flexible structure in which traction forces are expended on the solid and then cause a body deflection which finally affects the flow over moving boundaries. In spite of fluid and solid dynamics governed by individual equations, are commonly used, the strongly coupled FSI solution is currently an important topic of research.

In addition, the fluid and solid equations are solved separately in the implicit segregated approach, and the strongly coupled FSI between flow and flexible structure is limited only to their interface. Therefore, in order to handle the interaction between the fluid and solid solvers, an iterative algorithm should be used. Additionally, this iterative algorithm

is also required to enforce the equilibrium on their interface. Thus, the flow and the solid deformation are solved inside an interacting loop until the difference between the solutions of fluid and structure, such as the displacement of the interface, is less than a convergence criterion. The block Gauss-Seidel method also known the fixed-point method is the most commonly used coupling method [41]. This fluid-flexible structure interaction problem has been studied and discussed in a number of studies such as Dettmer and Peric' [42-43], Scheven and Ramm study [44], and Habchi *et al.*[45].

1.5 Fluid-Membrane Structure Interaction

Membrane structure designs are becoming considerably popular in modern engineering. These structures are also called tensile structures or fabric structures. They are considered a modern structural system. They were developed in the middle of the 20th century with a flexible and slender surface, that carries loads through tensile stresses without bending or compression. Membrane structures are good examples of extremely and highly optimised light-weight buildings.

The applications and demands for sustainable construction and the techniques of using new building materials, gave improvement to slender and light-weight structures in the field of civil engineering. With new materials being developed, the membranes have become wider reaching from mobile structures and large span roofs to cladding materials. As a result, for instance, the Millennium Dome (The O2 Arena) in London has become the focus of public interest in spectacular public constructions.

Designing these types of structures require that they should resist external loadings. In addition, more effort is also required to examine the structure's behaviour, as it becomes slender and lighter [46-47].

The strength of the material is used in an optimum way because of the steady stress condition over the thickness. Membrane structures are considered as special types of buildings due to their properties. They have very small or no bending stiffness. In addition, membranes are high strength, durable, self-cleaning, sound insulation, heat insulation, and low rate flammable.

The behaviour of the load carrying the membrane structures depends on tangential tension stresses to their surface. In compression stressed tangential case, membranes lose their stiffness and start to wrinkle. The external loading which is not tangential to the membrane surface leads relatively to large deformation. Therefore, membranes are designed to have doubly curved geometries, and pre-stress is applied for stabilisation to prevent the existence of large deformations even for little external loads.

The flexibility and slenderness in membrane structure constructions and material bring onward a high responsiveness to external loads. Membrane structures display high susceptibility, particularly for wind loading. In contrast to other load cases on membrane structures, such as snow or dead load, the wind load case cannot be assumed as a static load, but in some cases, the loading dynamics and the structural response have to be presumed. The wind load analysis on membrane structures is complicated in aeroelastic behaviour cases, in which large deformations of structures can cause an interaction between the structures and the wind flows [193-197].

In structural engineering, the typical approach to reducing the problem of the membrane structure being subjected to wind includes the risk of neglecting main effects, which result from the strong coupling of the wind and structural interaction [198-199].

Simiu and Scanlan [48] indicated that highly optimised buildings are an engineering task designed to ensure the performance of membrane structures subjected to wind and it will adequate from the first point during their anticipated life of serviceability and safety. Consequently, membranes are using materials that are the most efficient because of their load carrying behaviour [49-50].

Many studies have been carried out on the influence of wind flow on membrane structures and thin-walled shells. Yang *et al.* [51] classified the interaction between wind flow and structure into static and dynamic structures. The static interaction refers to the interaction between the static air and the vibrating membrane, while the dynamic interaction refers to the interaction between the wind flow and wind-induced vibrating structure. The static interaction can be considered as a special and simple case of the dynamic interaction.

In an experimental study, Yang *et al.* [61] experimentally evaluated the static and dynamic interaction by two parameters, aerodynamic damping and added mass. The study includes the static and dynamic interaction and its effects on structural dynamics. For the

static interaction, it includes the relationship between the covered membrane structure area and interaction parameters, while it contains the relationship between the wind direction and speed, and the interaction parameters, for the dynamic interaction. Experimental data represents that the dynamic interaction effect is substantial in the analysis of wind-induced response, and cannot be ignored. Moreover, the study concluded that the natural frequency of the structure is remarkably affected by the dynamic interaction.

For certain types of construction, the safety proof under extreme wind-loads is considered as one of the most demanding tasks in the field of construction engineering. For a simple case, a rigid construction affects only the wind flow direction. The pressure distribution performs a load on the structure surface [52]. The prediction of a dynamic interaction is considered as the most difficult issue. The structure starts fluttering due to fluid flow. This interaction can completely lead to structural failure.

Aerodynamic parameters have been studied by many researchers; however, additional work is still required for better understanding. Elashkar and Novak [53] investigated the requirements of aerodynamic parameters and its role in both free vibration and wind-induced vibration. Some models on the free vibration characteristics of cable-membrane structures have been carried out by Kawamura and Kiuchi [54], Takeda *et al.* [55], and Ishii [56]. According to Daw and Davenport study [57], the model of a forced flexible semi-cylindrical shell has been tested in a wind tunnel. This work concluded that the coefficients of aerodynamics are dependent on the structural shape and the amplitude.

Novak and Kassem [58] experimentally studied the free vibration of light-weight, self-supported large-span roofs supported by cavities with wall openings. The study investigated the wall openings influence on the frequency and the total structural damping and then compared it with a theory that they proposed. Finally, the results represented that the motion of air through the wall openings related to the roof motion. In addition, the air mass at the openings has a significant effect on the model damping and roof natural frequencies.

The free vibration of membrane to an inflow stream is studied by Il'chenko and Temnenko [59], where a numerical hydrodynamic system was used to analyse the effects of aerodynamic damping and the structural response. Kawai *et al.* [60] presented the flutter

such as a cantilever roof vibration based on a test of an aeroelastic wind tunnel. It was observed that the oscillations resulted in a particular wind velocity because of the low natural frequency value, which was excited by the aerodynamic stiffness value and the vortex-induced damping.

Experimental simulations in wind tunnels are general tools to examine dynamic effects in wind engineering. For this experimental method, the correspondence of dynamic behaviour between model and reality is considered as a basic requirement. However, this requirement is difficult to meet for small-scale models, particularly for the aeroelasticity analysis. As a result, wind tunnel experiments are expensive and complicated. According to Williams [200], the state of the membrane structures analysis in the wind was described as extremely complex due to no satisfying method of design, and it is an open question of connecting between experience with similar structures and experiments with simple theories.

Besides wind tunnel experimental methods, the Computational Fluid Dynamics (CFD) numerical methods application becomes applicable for wind effects analysis. The use of CFD methods in wind engineering is called Computational Wind Engineering (CWE). Furthermore, the use of numerical simulations for the aeroelasticity phenomenon is a guaranteed complement and improvement of experimental methods. The simulation of aeroelastic effects is the proper combination of various numerical simulations which is done by fluid-structure interaction approach [201-202].

1.6 Aim of the Thesis

The objective of this work is to study the vortex-induced vibration phenomena in different fluid-structure interaction benchmarks, particularly in two-dimensional and three-dimensional flow domains. The current study covers four main topics: the free and forced vibration of a cylindrical structure, 2D and 3D models of flow-induced vibration of a flexible tail attached to a solid block, 2D and 3D models of the membrane roof, and finally wind- membrane interaction is studies.

The work done here is focused on the use of OpenFOAM and foam-extend, which are two sibling open source software projects which provide a general purpose toolbox for

solving continuum mechanics problems, although it is mostly oriented towards computational fluid dynamics, although it is mostly oriented towards Computational Fluid Dynamics. Associated with the foam-extend project is an additional toolkit for advanced fluid-solid interaction which was also used in this work [10].

The choice for these software projects was done based on the following criteria:

- The software had to be open source, in order to allow for future work on this topic to not have to rely on costly software licenses, as well as allowing for future work to freely perform open software development for research and development of wind-interacting membranes, for which closed source software often does not allow.
- To use software that was complete enough to model all of the fluid flow and solid structure phenomena that can occur in real life; for example, with OpenFOAM/foam-extend, it's possible to use Large Eddy Simulation (LES) modelling of vortexes, with a higher accuracy than conventional Finite Element Method software, such as Elmer-FEM which only has basic fluid flow modelling.
- The software should allow for work well beyond the original scope of wind-membrane modelling, such as the ability to account for real world environments, such as weather storms, heat waves, acoustic effects, wear and tear, as well as the respective response from electronic equipment and human presence to all of these.

For all of these, only OpenFOAM and foam-extend would fit the all of these criteria, which would otherwise require one or more commercial software applications to perform.

However, given that this specific open source software is fairly complex to use, this thesis provides the groundwork and necessary information on how the results were achieved, what were the limitations when using these software projects, along with an overview of how to conduct studies with them and what to expect as the work progresses.

This is due to the nature of these open source projects that were used during this thesis, given that they are in constant development, with new developments released each year that either change completely how a particular feature is used and/or fix critical bugs that were in previous versions. The following list provides an overview of the versions that were used:

- OpenFOAM 2.2 was used for the free and forced vibration of a cylindrical structure, because OpenFOAM 2.3 had changed how the controls for springs, dampers and restricted movement were implemented, making it not possible to use them when the study was done at the time. As for foam-extend, neither of the existing versions provided the same level of control as OpenFOAM 2.2.
- Two versions of foam-extend had to be used, namely 3.1 and 4.0, along with the respective Fluid-Solid/Structure Interaction toolkit (the toolkit name changed between versions), depending on the type of simulation that was conducted. More specifically, the following foam-extend versions were used for each study:
 - foam-extend 3.1 version had been used for 2D and 3D models of the membrane roof, and 2D model of flow-induced vibration of a flexible tail attached to a solid block.
 - However, foam-extend 4.0 version had been used to study 3D model of flow-induced vibration of a flexible tail attached to a solid block
- Furthermore, neither versions of foam-extend allowed for the creation of a complete mesh of the tent case (the last study); it was necessary to use OpenFOAM 2.4.0 for generating the mesh that was then used to run the simulations with foam-extend.

In addition, it establishes the need to rely on similar benchmark cases, hence starting with the vortex-induced vibration of circular cylinder cases and continuing with the more complex cases that exhibit an oscillatory interaction between the forces exerted by fluid flow and solid motion. However, this task revealed itself to not always be straight forward as well, given that:

- In some situations, the original benchmarks were not properly documented, where one failed to specify the units to be in CGS, while another mixed up units of both SI and CGS, therefore having to cross-reference with yet another study of the same cases (when possible) in which greater effort in unit documentation was upheld, which happened for the study on 2D and 3D models of flow-induced vibration of a flexible tail attached to a solid block.

- Another benchmark case was too complex to recreate with OpenFOAM/foam-extend, which happened for the final study.

Hopefully with the ground work done in this thesis, work on this topic can continue in the future, but it will also strongly depend on how these software projects continue to evolve in the future, which is expected to occur in the communities involved with them.

1.7 Layout of the Thesis

Chapter 2: All important equations, space and temporal discretisation, and methodology to use the FSI solver in OpenFOAM and foam-extend technology are introduced. The FSI solver used is based on a partitioned approach. The finite volume method (FVM) is applied for the fluid flow discretisation on a moving grid in an arbitrary Lagrangian–Eulerian (ALE) formulation. The St. Venant–Kirchhoff constitutive law is used to analyse the structural elastic deformation for large non-linear deformations in a Lagrangian formulation. The FVM is also used to discretise the solid structure in an iterative segregated approach. The algorithm of the pressure-velocity coupling is applied to the large time-steps in the moving boundary problem.

Chapter 3: A detailed description of the 2D vortex-induced vibration past a circular cylinder is presented. The simulation of uniform flow over a stationary circular cylinder is presented at different Reynolds numbers ($100 \leq Re \leq 1000$), and then the flow past a free and forced oscillation of a cylinder is investigated. VIV simulations for the naturally oscillating cylinder is studied with one- and two-degrees-of-freedom in a laminar flow ($Re = 100$). In the VIV simulation, stiffness and damping of springs are changed to investigate its effects on the VIV behaviour.

Chapter 4: The 2D and 3D space models and benchmarking of FSI examples are presented. The strong coupling between fluid and solid solvers is performed, and the equilibrium on the fluid-structure interface is achieved by using a fixed-point algorithm with Aitken’s under-relaxation method or IQN-ILS. The solver of the automatic mesh motion depends on the Laplace smoothing equation with mesh diffusion variable. The fsiFoam solver is implemented for all cases in this chapter. This solver is validated on

two benchmarks in the 2D space and their extension to the 3D space. Several benchmark and practical examples are presented.

Chapter 5: A 3D space membrane structure application is presented. This example is more complex than the one discussed in the previous chapter due to the tension of the structure. The tension structures cover many categories such as fabric membranes. In this chapter, the four-point tent structure is a similar study to the behaviour of the thin membrane subjected to wind flow. Designing the thin-walled membrane in such a way, particularly in civil engineering, requires more effort to put into analysing the structural behaviour, as it becomes lighter and slenderer.

Chapter 6: The summary of results and achievements obtained from the previous chapters and finally suggestions for future work are discussed.

Chapter 2

Equations, Discretisation and Methodologies

In the fluid mechanics field, numerical procedures are employed to solve the Navier-Stokes equations. Briefly, the method consists of splitting the computational domain into finite discrete elements creating a mesh. The differential equations are discretised, over these elements to produce a set of algebraic equations. The solution of the algebraic system provides a set of values of variables at some determined locations in time and space. The discretisation procedure can be classified into the solution domain discretisation and equation discretisation [63,132].

The details of the flow and structural equations are described in Section 2.1. Section 2.2 presents the discretisation practice of the finite volume modelling and the classifications of the discretisation procedure are discussed in detail. The details of OpenFOAM library for fluid-structure interaction and coupling procedure are presented in the remainder of this Chapter from Section 2.3 – 2.11.

2.1 Governing Equations

2.1.1 Flow Equations

The Navier-Stokes equations governed the flow field for incompressible viscous flow. The continuity and momentum equations are

$$\nabla \cdot \mathbf{U}_f = 0, \quad (2.1)$$

$$\frac{\partial \mathbf{U}_f}{\partial t} + (\nabla \cdot \mathbf{U}_f) \mathbf{U}_f = -\frac{\nabla p}{\rho_f} + \nabla \cdot (\nu_f \nabla \mathbf{U}_f) \quad (2.2)$$

where \mathbf{U}_f is flow velocity, ρ_f is fluid density, p is pressure, ν_f is the kinematic viscosity ($\nu_f = \mu_f / \rho_f$), μ_f is the dynamic viscosity.

The mass and momentum conservation equations (2.1) and (2.2) are both satisfied in the fluid reference domain $\Omega_{f,0}$. Such governing equations for the flows without moving mesh can be discretised by considering the Eulerian description in which the mesh is fixed [64]. However, in the Lagrangian formulation, the mesh moves with the flow. Such types of strategies are invalid for the computational domains that deform extremely in time. Often, the Arbitrary Lagrangian- Eulerian (ALE) formulation [64] is utilised for handling the equations of flow on a deformed mesh, as presented in Section 2.7.

2.1.2 Structural Equations

The balance of momentum for the structural body is

$$\frac{\partial^2(\rho_s \mathbf{U}_s)}{\partial t^2} - \nabla \cdot \boldsymbol{\sigma}_s = \rho_s \mathbf{F}_b. \quad (2.3)$$

Where \mathbf{U}_s is the displacement of the solid, ρ_s is the solid density. The force of the body and the Cauchy stress tensor are denoted with \mathbf{F}_b and $\boldsymbol{\sigma}_s$, respectively.

The strain tensor $\boldsymbol{\varepsilon}$ in terms of \mathbf{U}_s is given by

$$\boldsymbol{\varepsilon} = \frac{1}{2} [\nabla \mathbf{U}_s + (\nabla \mathbf{U}_s)^T], \quad (2.4)$$

The Hook's law in terms of stress and strain tensors is defining by the following system [54,66,81]

$$\boldsymbol{\sigma}_s = 2\mu_s \boldsymbol{\varepsilon} + \lambda_s \text{tr}(\boldsymbol{\varepsilon})\mathbf{I}, \quad (2.5)$$

where \mathbf{I} is the identity matrix, tr is the tensor trace, μ_s and λ_s are the Lamé constants which represent the elastic material characteristics. These constants are interlinked to the Poisson's ratio ν_s and the Young Modulus E_s by the following

$$\lambda_s = \frac{\nu_s E_s}{(1 + \nu_s)(1 - 2\nu_s)}, \quad (2.6)$$

$$\mu_s = \frac{E_s}{2(1+\nu_s)} \quad (2.7)$$

By substituting into Equation (2.3) yields

$$\frac{\partial^2(\rho_s \mathbf{U}_s)}{\partial t^2} - \nabla \cdot [\mu_s \nabla \mathbf{U}_s + \mu_s (\nabla \mathbf{U}_s)^T + \lambda_s \mathbf{I} \text{tr}(\nabla \mathbf{U}_s)] = \rho_s \mathbf{F}_b. \quad (2.8)$$

2.2 Introduction to Finite Volume Discretisation

The aim of any discretisation method is to transform partial differential equations (PDEs) into a system of algebraic expressions. This system obtains a set of values that correspond to the solution of original equations solution at some determined locations in space and time. Equation discretisation derives a system of algebraic equations from the differential equations.

The first step in all computational fluid dynamics (CFD) procedures is splitting the computational domain into a finite number of cells called mesh.

These discretisation procedures are further discussed in the sections below. As OpenFOAM depends on the finite volume method (FVM), the discretisation that will discuss in this chapter will follow the FV procedure.

2.2.1 Discretisation of the Computational Domain

The discretisation process of the solution domain can be subdivided into spatial discretisation and temporal discretisation [68-69]. Spatial discretisation consists of dividing the computational domain into a finite number of elements constituting a mesh called control volumes (CVs). Control volumes are completely fill the solution domain. Temporal discretisation is only applied to transient problems. It contains dividing the time into finite intervals called time-steps.

Figure 2.1 shows a typical control volume where the computational point P placed in the centre of the CVs, such that:

$$\int_{V_P} (\mathbf{x} - \mathbf{x}_P) dV = \mathbf{0}. \quad (2.9)$$

In the control volume, the cell faces can be divided into internal faces which are between two control volumes, and boundary faces that coincide with the domain boundaries. The vector of the face area \mathbf{S}_f is created for each face in a way that it points outward from the control volume with the lower label. It is normal to the face, and its magnitude is equal to the face area. In Figure (2.1), the owner and neighbour cell centres are respectively marked with P and K for the shaded face, as the vector of the face area \mathbf{S}_f points outward from the P cell. For simplicity, all control volume faces will be denoted by f , which also shows the point in the centre of the face.

In OpenFOAM, variables are principally stored at the centre of the element, although they might be stored on vertices or faces. Generally, the size of the mesh is an important process in CFD procedures. The fine mesh leads to high computational recourses, while, the too coarse mesh may produce to wrong results.

2.2.2 Transport Equation Discretisation

A scalar variable is a variable or a field that holds only one variable at the time. It contains a single component that assumes a range of values. The transport equation form for a scalar variable ϕ reads:

$$\underbrace{\frac{\partial \rho \phi}{\partial t}}_{\text{Temporal derivative}} + \underbrace{\nabla \cdot (\rho \mathbf{U} \phi)}_{\text{Convection term}} = \underbrace{\nabla \cdot (\Gamma \nabla \phi)}_{\text{Diffusion term}} + \underbrace{S(\phi)}_{\text{Source term}}, \quad (2.10)$$

where ρ is density, \mathbf{U} is velocity and Γ is the diffusion coefficient.

Since ρ and Γ are constants, and the source term is not essential in this study, Equation (2.10) becomes

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{U} \phi) = \nabla \cdot (\nabla \phi) \quad (2.11)$$

This equation is a second-order equation because of the availability of the second derivative of ϕ in space in the diffusion term. Therefore, the discretisation that implemented in this work is a second-order accurate in space and time and will be illustrated in this Chapter. In addition, each term of the transport equation will be presented separately.

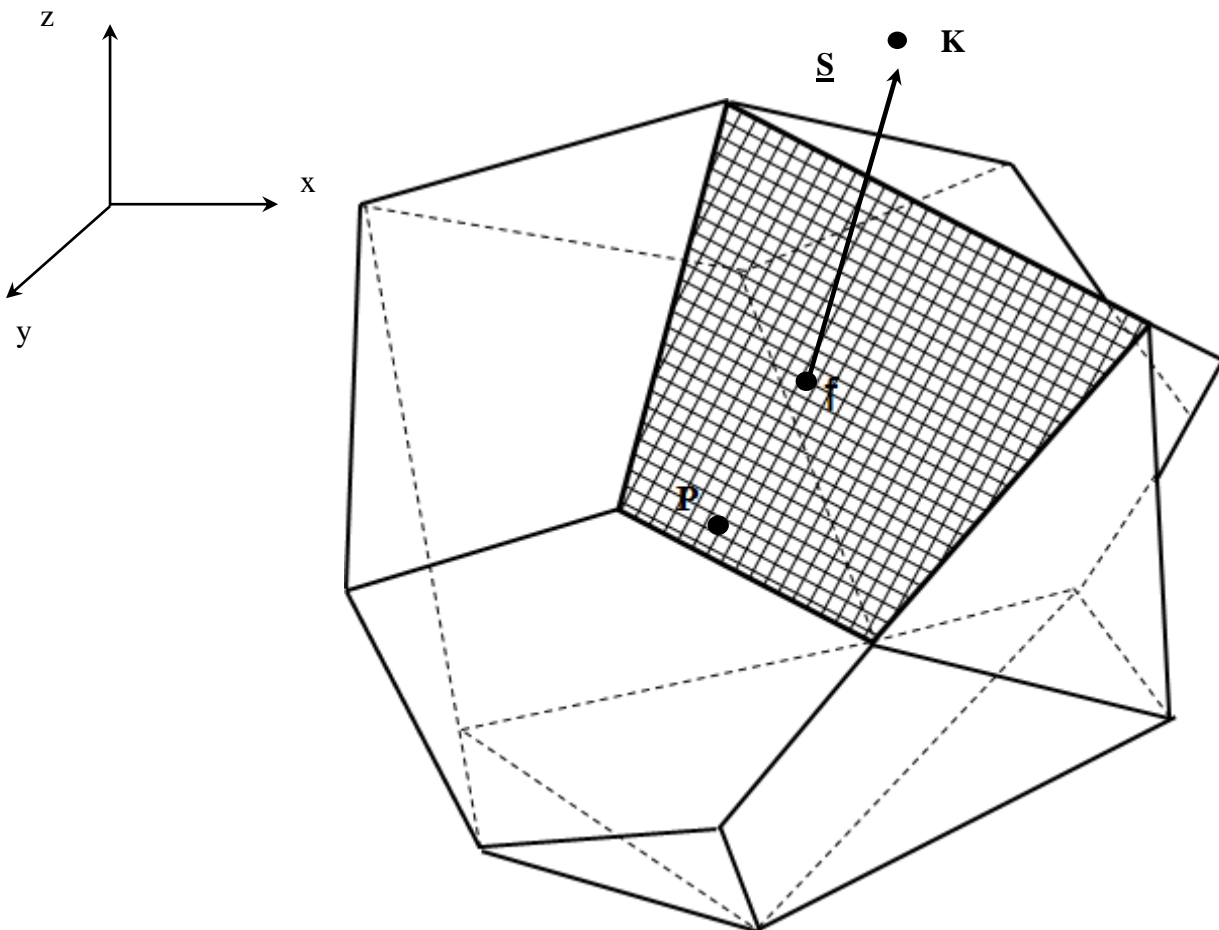


Figure 2.1: Control volume

To apply the finite volume method, Equation. (2.11) should be integrated into each control volume V_P around the point P as in the following expression:

$$\int_{V_P} \frac{\partial \phi}{\partial t} dV + \int_{V_P} \nabla \cdot (\mathbf{U} \phi) dV = \int_{V_P} \nabla \cdot (\nabla \phi) dV \quad (2.12)$$

For unsteady problems, the last equation must be integrated over the interval $[t, t + \Delta t]$ and then the integrated transport equation (2.11) will be expressed as

$$\int_t^{t+\Delta t} \left[\frac{\partial}{\partial t} \int_{V_P} \phi dV + \int_{V_P} \nabla \cdot (\mathbf{U} \phi) dV - \int_{V_P} \nabla \cdot (\nabla \phi) dV \right] dt = 0. \quad (2.13)$$

The discretisation method accuracy depends on space and time variation of the function $\phi = \phi(\mathbf{x}, t)$ around the point P . Therefore, the variation should be expressed in space and time as follows

$$\phi(\mathbf{x}) = \phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla \phi)_P, \quad (2.14)$$

$$\phi(t + \Delta t) = \phi^t + \Delta t \left(\frac{\partial \phi}{\partial t} \right)^t. \quad (2.15)$$

Where $\phi_P = \phi(\mathbf{x}_P)$ and $\phi^t = \phi(t)$.

2.2.3 Face interpolation

It is very important to choose the domain locations where the variable values will be stored before solving the governing equations. The most common selecting options are shown in Figures below. Figure (2.2a) presents the method of storing the variables at the centre of each cell and at the faces of the boundaries. This method is known as a collocated

method. While Figure (2.2b) illustrates the segregated method, which is storing the values of all variables in the centroid of each face.

OpenFOAM applies the collocated method for storing the variables, and the discretisation process of the Equation (2.13) will be discussed below.

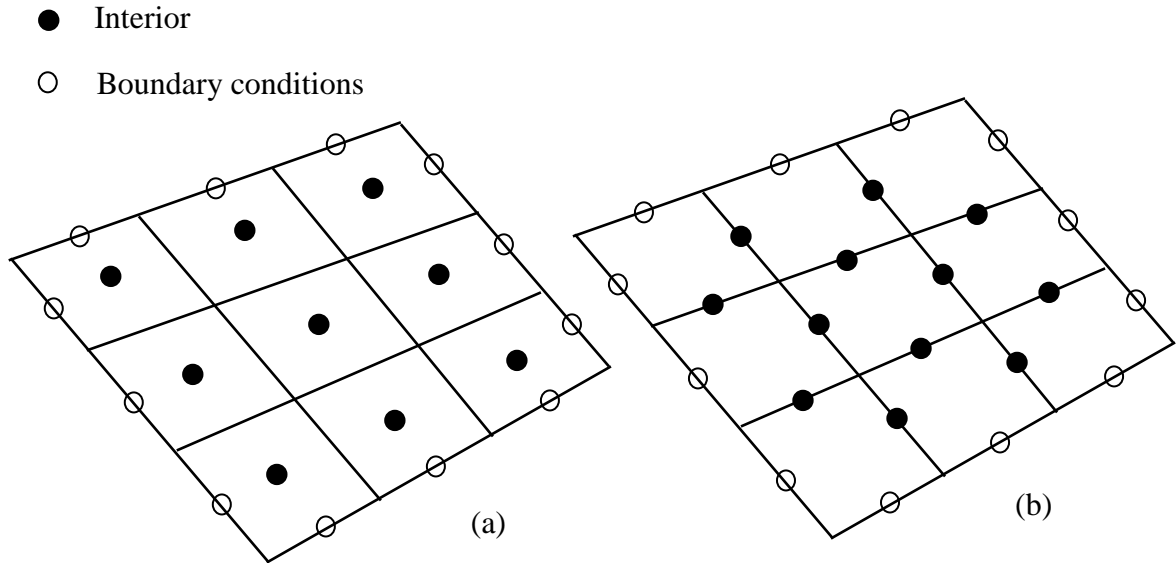


Figure 2.2: Storing variables options, (a) monolithic method and (b) partitioned method

2.2.4 Discretisation of Spatial Terms

For the spatial terms discretisation, the Gauss divergence theorem will be applied through the discretisation process involving the following identities

$$\int_V \nabla \cdot \mathbf{a} dV = \oint_{\partial V} d\mathbf{S} \cdot \mathbf{a}, \quad (2.16)$$

$$\int_V \nabla \phi dV = \oint_{\partial V} d\mathbf{S} \phi. \quad (2.17)$$

In which \mathbf{a} is a general vector property, ∂V represents the closed surface that bounding the volume V , and $d\mathbf{S}$ shows infinitesimal surface element with the normal of the associated outward points on ∂V .

Then, surface and volume integrals must be evaluated considering the prescribed ϕ variation over the control volume P as shown in Equation (2.17),

$$\int_{V_P} \phi(\mathbf{x}) dV = \int_{V_P} [\phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla \phi)_P] dV = \phi_P V_P, \quad (2.18)$$

where V_P is the cell volume. The second integral term of the last equation equals to zero because the point P located in the control volume centroid.

By considering terms under the divergence operator and the control volume is bounded by a set of flat faces, then Equation (2.16) can be expressed by a sum of integrals overall flat faces as

$$\begin{aligned} \int_{V_P} \nabla \cdot \mathbf{a} dV &= \oint_{\partial V_P} d\mathbf{S} \cdot \mathbf{a} \\ &= \sum_f \left(\int_f d\mathbf{S} \cdot \mathbf{a} \right). \end{aligned} \quad (2.19)$$

For the face integral in Equation (2.19), the linear variation assumption of the function ϕ gives

$$\begin{aligned} \int_f d\mathbf{S} \cdot \mathbf{a} &= \left(\int_f d\mathbf{S} \right) \cdot \mathbf{a}_f + \left[\int_f d\mathbf{S} (\mathbf{x} - \mathbf{x}_f) \right] \cdot (\nabla \mathbf{a})_f \\ &= \mathbf{S} \cdot \mathbf{a}_f. \end{aligned} \quad (2.20)$$

By combining Equations (2.18) – (2.20), the form of the second order accurate of the Gauss theorem is found

$$(\nabla \cdot \mathbf{a})V_P = \sum_f \mathbf{S} \cdot \mathbf{a}_f \quad (2.21)$$

In this case, f indicates the value of the variable in the middle of the face, and \mathbf{S} is the area vector of the outward-pointing face. In the structure of the current mesh, \mathbf{S}_f points outwards from P only if f is owner face to P , and for the neighbouring faces, \mathbf{S}_f points inward. Therefore, the sum over the faces is given by the following expression

$$\sum_f \mathbf{S} \cdot \mathbf{a}_f = \sum_{\text{owner}} \mathbf{S} \cdot \mathbf{a}_f - \sum_{\text{neighbour}} \mathbf{S} \cdot \mathbf{a}_f \quad (2.22)$$

2.2.4.1 Convection Term

By using Equation (2.21), the convection term discretisation is given as

$$\begin{aligned} \int_{VP} \nabla \cdot (\mathbf{U} \phi) dV &= \sum_f \mathbf{S} \cdot (\mathbf{U} \phi)_f \\ &= \sum_f Q \phi_f, \end{aligned} \quad (2.23)$$

$Q = \mathbf{S} \cdot (\mathbf{U})_f$ is the mass flux through the face and it can be calculated from the interpolated values of \mathbf{U} .

Then, the convection differencing scheme is used to determine the face value of ϕ from the values in the centres of the cell.

2.2.4.2 Convection Differencing Scheme

The objective of the convection differencing scheme is to calculate the value of the variable ϕ on the face from the values in the cell centres. Here, the differencing schemes are limited by using only the nearest neighbours of the control volume because, in unstructured meshes framework, it would not be possible to use any other values more than ϕ_P and ϕ_K due to the storage associated with the additional information.

By assuming the linear variation between P and K of the variable ϕ is calculated from

$$\phi_f = f_x \phi_P + (1 - f_x) \phi_K. \quad (2.24)$$

Where, f_x is the interpolation factor and is defined as the distance ratio $\frac{\overline{fK}}{\overline{PK}}$,

i.e. $f_x = \frac{\overline{fK}}{\overline{PK}}$.

Several variations have been developed for the central and upwind differencing schemes. The users can discover all accessible interpolation schemes of OpenFOAM from the user guide [125].

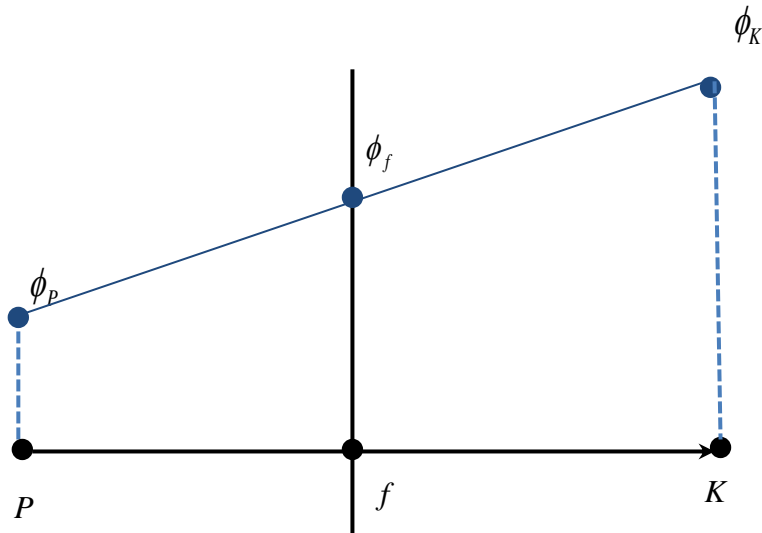


Figure 2.3: Face interpolation

2.2.4.3 Diffusion Term

By the same way applied for discretisation of the convective term, the discretisation of the diffusive term is implemented as

$$\begin{aligned} \int_{VP} \nabla \cdot (\nabla \phi) dV &= \sum_f \mathbf{S} \cdot (\nabla \phi)_f \\ &= \sum_f \mathbf{S} \cdot (\nabla \phi)_f. \end{aligned} \quad (2.25)$$

2.2.5 Temporal Discretisation

The explicit and implicit procedures are considered as main classifications for time derivative terms of discretisation method. The explicit method utilises the values from the previous time-step while the implicit method can be solved iteratively and it includes values from the next time-step. Therefore, the temporal integral can be performed by using the explicit (forward Euler) and implicit (backward Euler) methods.

Using Equations (2.23) and (2.25) to discretise the transport equation (2.13)

$$\int_t^{t+\Delta t} \left[\left(\frac{\partial \phi}{\partial t} \right) V_P + \sum_f Q \phi_f - \sum_f \mathbf{S} \cdot (\nabla \phi)_f \right] dt = 0. \quad (2.26)$$

This expression is called semi-discretised from the general form of the transport equation [68]. Then, from the variation of the function ϕ in time, Equation (2.15), the discretisation of the temporal integral and the time derivative can be calculated as:

$$\left(\frac{\partial \phi}{\partial t} \right)_P = \frac{\phi_P^n - \phi_P^{n-1}}{\Delta t}, \quad (2.27)$$

$$\int_t^{t+\Delta t} \phi(t) dt = \frac{1}{2} (\phi^{n-1} + \phi^n) \Delta t \quad (2.28)$$

Where $\phi^{n-1} = \phi(t)$ and $\phi^n = \phi(t + \Delta t)$.

In the explicit method, the temporal discretisation of the transport equation can be calculated using the old-time value.

$$\phi_P^n = \phi_P^{n-1} + \frac{\Delta t}{V_P} \left[\sum_f Q \phi_f^{n-1} - \sum_f \mathbf{S} \cdot (\nabla \phi)_f^{n-1} \right]. \quad (2.29)$$

All terms on the right-hand-side of Equation (2.29) depend on the old-time level. Therefore, the new value of ϕ_P can be directly calculated. This explicit procedure is

unstable when the Courant number value (Equation (2.30)) is larger than 1. The Courant number is defined as

$$Co = \frac{U \Delta t}{\Delta x}, \quad (2.30)$$

where U is the flow velocity, Δx is the size of the mesh, and Δt represents the time-step.

The implicit discretisation expresses values of the face in terms of the new-time level of the cell values as follows

$$\phi_p^n = \phi_p^{n-1} + \frac{\Delta t}{V_p} \left[\sum_f Q \phi_f^n - \sum_f \mathbf{S} \cdot (\nabla \phi)_f^n \right]. \quad (2.31)$$

This procedure is unconditionally stable [68].

2.2.6 Pressure-Velocity Coupling

The solution of the conservation of mass and momentum equations (2.1) and (2.2) displays the following issues:

- The convective term of the momentum equation contains non-linear quantities, \mathbf{U}^2 .
- Continuity and momentum equations are essentially coupled because the velocity component appears in both mass and momentum equations. However, the pressure appears only in the momentum equation and no transport equation or other for the pressure.

If the gradient of pressure is known, the method of discretizing equations for velocity from the momentum conservation equation is similar to that for other scalars. However, in flow calculations, it is essential to compute the pressure field as part of the solution. Thus, the pressure gradient is not known beforehand. The pressure can be obtained from the temperature and density by using the state equation $p = p(\rho, T)$. For the incompressible flow, the density is constant. Thus, this definition is not applicable.

Therefore, in this case, the coupling between velocity and pressure presents a limitation on the flow solution. Particularly, applying the correct pressure field component in the momentum equation leads to the velocity field satisfying the continuity equation. These algorithms are known as SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) and PISO (Pressure Implicit with Splitting of Operators) OpenFOAM usually employs a SIMPLE algorithm for the steady-state simulation and PISO algorithm for the transient simulation [70].

To solve the non-linearity problem either employ a non-linear equation solver or linearise the convection term. However, due to the difficulty of implementing the non-linear solver and the computational cost required, the convection term linearisation is used here. The convection term linearisation is given by the following expression

$$\begin{aligned}
\nabla \cdot (\rho \mathbf{U} \mathbf{U}) &= \sum_f \mathbf{S} \cdot \rho \mathbf{U}_f \mathbf{U}_f \\
&= \sum_f Q \mathbf{U}_f \\
&= a_p \mathbf{U}_p + \sum_K a_K \mathbf{U}_K.
\end{aligned} \tag{2.32}$$

In which the mass flow rate Q , a_K is a matrix coefficient corresponding to neighbouring cell K , a_p is a central coefficient, and \mathbf{S} is the outward-pointing area vector of the face. Note that Q , a_K and a_p are all functions of \mathbf{U} .

The fluxes Q should satisfy the mass conservation equation, and mass and momentum equations should be solved together. Linearisation of the convection term indicates that an existing velocity (flux) field that satisfies continuity, and hence will be applied to calculate a_p , and a_K .

The second part in the right-hand side given in the last expression in equation (2.32) is explained by the net flux over the control volume boundary equals the sum of integrals through the four control volume faces in 2D and six control volume faces in 3D. However, the interpolation is used to determine since the integrand value is not available at the control volume.

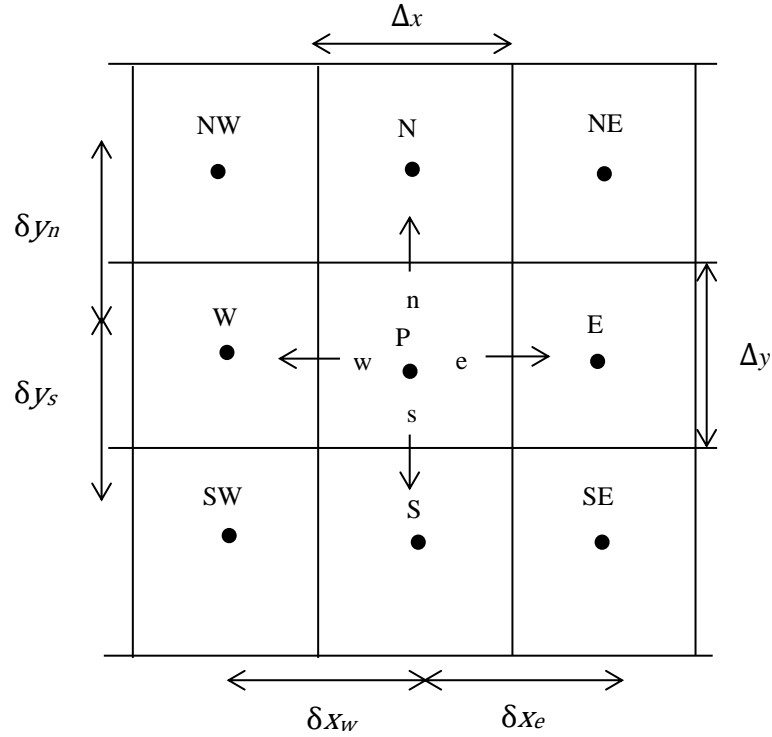


Figure 2.4: Arbitrary 2D finite volume grid with notations

To understand the discretisation of the conservation equations that used in computational fluid dynamics, the example of the transport equation will be involved (incompressible flow, constant density).

The transport equation form for a scalar variable ϕ in Equation (2.11) can be expressed as

$$\frac{\partial \phi}{\partial t} + \frac{\partial}{\partial x_i} (\mathbf{U}\phi) = \frac{\partial}{\partial x_i} \left(\frac{\partial \phi}{\partial x_i} \right) \quad (2.33)$$

This equation will be discretised for the simple flow field shown in the figure below by assuming steady state conditions.

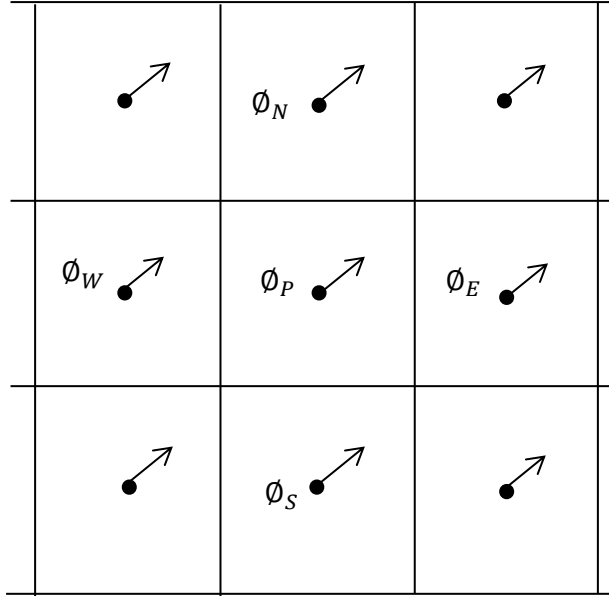


Figure 2.5: Simple flow field by steady conditions

To find the balance over the control volume, the Equation (2.33) is given by

$$\begin{aligned}
 & A_e u_e \phi_e - A_w u_w \phi_w + A_n v_n \phi_n - A_s v_s \phi_s \\
 & = A_e \left. \frac{d\phi}{dx} \right|_e - A_w \left. \frac{d\phi}{dx} \right|_w + A_n \left. \frac{d\phi}{dx} \right|_n - A_s \left. \frac{d\phi}{dx} \right|_s.
 \end{aligned} \tag{2.34}$$

Where (A_e, A_w, A_n, A_s) indicate the area of the faces, $(\phi_e, \phi_w, \phi_n, \phi_s)$ show the concentrations at the faces, $(\phi_E, \phi_W, \phi_N, \phi_S)$ are concentrations at the cell centres, $(u_e, u_w, u_n, u_s, v_e, v_w, v_n, v_s)$ are velocities at the faces, and $(u_E, u_W, u_N, u_S, v_E, v_W, v_N, v_S)$ present velocities at the cell centres.

The last equation contains values at faces which need to be determined from the interpolation form from the values placed at the cell centres.

Then, by using the first order upwind differencing to determine the values at the faces, let's assume that the value in the cell centre of the face is equal to the value in the centre.

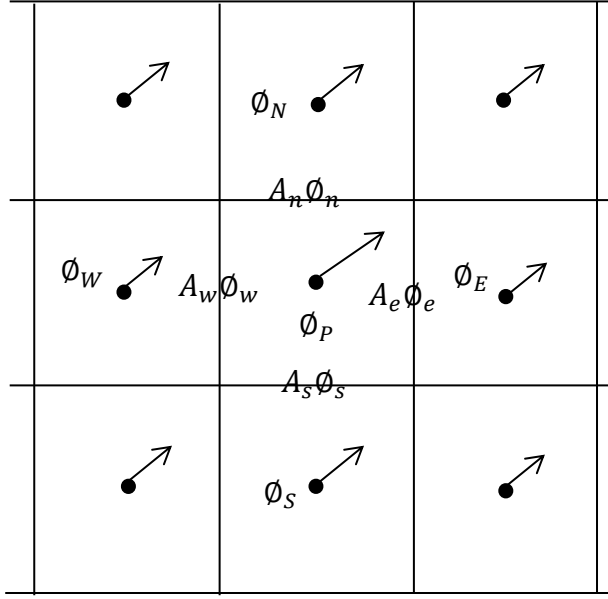


Figure 2.6: Values at the faces determined from interpolation from the values at the cell centres

Therefore, the result is

$$\begin{aligned}
 & A_e u_P \phi_P - A_w u_W \phi_W + A_n v_P \phi_P - A_s v_S \phi_S \\
 &= \frac{A_e(\phi_E - \phi_P)}{\delta x_e} - \frac{A_w(\phi_P - \phi_W)}{\delta x_w} + \frac{A_n(\phi_N - \phi_P)}{\delta y_n} - \frac{A_s(\phi_P - \phi_S)}{\delta y_s}.
 \end{aligned} \tag{2.35}$$

The last equation can be rearranged to show an expression at the cell centre P as a function in the surrounding cells, the grid, and the flow field as follows

$$\begin{aligned}
 & \phi_P \left(A_n v_P + A_e u_P + \frac{A_w}{\delta x_w} + \frac{A_n}{\delta y_n} + \frac{A_e}{\delta x_e} + \frac{A_s}{\delta y_s} \right) \\
 & \phi_W \left(A_w u_W + \frac{A_w}{\delta x_w} \right) + \phi_N \left(\frac{A_n}{\delta y_n} \right) + \phi_E \left(\frac{A_e}{\delta x_e} \right) + \phi_S \left(A_s v_S + \frac{A_s}{\delta y_s} \right).
 \end{aligned} \tag{2.36}$$

Now, this equation can be simplified to

$$\begin{aligned}
 a_P \phi_P &= a_W \phi_W + a_N \phi_N + a_E \phi_E + a_S \phi_S + b \\
 &= \sum_K a_K \phi_K + b
 \end{aligned} \tag{2.37}$$

Where K refers to the neighbouring cells. The coefficients a_K and b will be different at every iteration for every cell in the domain.

The linearisation process does not affect steady-state problems. However, for the unsteady problems, two approaches can be implemented either to neglect the effects of the lagged non-linearity or to use the iteration over non-linear terms. The iteration procedure can significantly improve the computational cost, but only for the large time-steps. Thus, the non-linear system is resolved for every time-step, where its size limitation comes from the temporal accuracy. However, small time-steps are needed to resolve. Thus, if the time-steps are small enough, the change between consecutive solutions will also be small, and therefore it is likely to lag the non-linearity of the system without any significant effect.

In OpenFOAM, the PISO algorithm was proposed by Issa [71] is usually used for pressure-velocity coupling in transient calculations. However, for steady-state calculations, the SIMPLE procedure developed by Patankar [72] is commonly employed for pressure-velocity coupling.

In this thesis, the PISO algorithm is used in all cases in Chapters 4 and 5. Besides the PISO algorithm, the PIMPLE procedure which is combined both PISO and SIMPLE algorithms is applied in the study cases in Chapter 3.

2.2.6.1 Derivation of the Pressure Equation

In order to find the pressure equation, a semi-discretised form of the momentum conservation equation will be used as follows [63]

$$a_p \mathbf{U}_p = \mathbf{H}(\mathbf{U}) - \nabla p . \quad (2.38)$$

Where $\mathbf{H}(\mathbf{U})$ contains the rest terms of the momentum equation that are not given in Equation (2.38). The pressure gradient term is not discretised at this stage [73]. Equation (2.38) is found from the integral form of the momentum equation by applying the discretisation process that described previously. It has been subsequently divided through

by the volume to allow face interpolation of the coefficients. $\mathbf{H}(\mathbf{U})$ in Equation (2.38) is given as

$$\mathbf{H}(\mathbf{U}) = -\sum_K a_K \mathbf{U}_K + \frac{\partial \mathbf{U}}{\partial t} - \nabla \cdot (\nu \nabla \mathbf{U}). \quad (2.39)$$

The $\mathbf{H}(\mathbf{U})$ term contains the transport part and the source part. The transport part consists of coefficients of the matrix for all neighbours multiplied by all corresponding velocities. The source part includes the source part of the unsteady term and all additional source terms besides the pressure gradient. However, in this case, there are no additional source terms.

The discretised form of the continuity Equation (2.1) gives

$$\nabla \cdot \mathbf{U} = \sum_f \mathbf{S} \cdot \mathbf{U}_f = 0. \quad (2.40)$$

To substitute momentum conservation equation into continuity equation, \mathbf{U}_P is evaluated by Equation (2.40) as

$$\mathbf{U}_P = \frac{\mathbf{H}(\mathbf{U})}{a_P} - \frac{1}{a_P} \nabla p. \quad (2.41)$$

Velocities on the faces are stated as the face interpolate of Equation (2.41)

$$\mathbf{U}_f = \left(\frac{\mathbf{H}(\mathbf{U})}{a_P} \right)_f - \left(\frac{1}{a_P} \right)_f (\nabla p)_f. \quad (2.42)$$

This expression will be applied later for the face flux calculation.

By substituting Equation (2.42) into Equation (2.40) the following pressure equation form is obtained:

$$\begin{aligned}\nabla \cdot \left(\frac{1}{a_p} \nabla p \right) &= \nabla \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_p} \right) \\ &= \sum_f \mathbf{S} \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_p} \right)_f.\end{aligned}\tag{2.43}$$

The final discretisation form of the incompressible Navier-Stokes equations reads

$$a_p \mathbf{U}_p = \mathbf{H}(\mathbf{U}) - \sum_f \mathbf{S}(p)_f,\tag{2.44}$$

$$\sum_f \mathbf{S} \cdot \left[\left(\frac{1}{a_p} \right)_f (\nabla p)_f \right] = \sum_f \mathbf{S} \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_p} \right)_f.\tag{2.45}$$

The face fluxes Q are calculated using Equation (2.42) as

$$Q = \mathbf{S} \cdot \mathbf{U}_f = \mathbf{S} \cdot \left[\left(\frac{\mathbf{H}(\mathbf{U})}{a_p} \right)_f - \left(\frac{1}{a_p} \right)_f (\nabla p)_f \right].\tag{2.46}$$

When Equation (2.43) is satisfied, then the conservation of the face fluxes is guaranteed.

2.2.6.2 PISO, SIMPLE, and PIMPLE Algorithms

The discretised form of the Navier-Stokes equations for an incompressible flow that is given by Equations (2.44) and (2.45) show a linear dependence of pressure on velocity and vice-versa. Thus, a special treatment should be involved in this inter-equation coupling.

- Simultaneous approach: this operates by solving the complete equation system over the whole domain simultaneously. This procedure may be considered when the computational points number is small, and the simultaneous equations number is not too large. Therefore, the resulting matrix consists of the inter-equation coupling, and it is larger than the computational points number. The computational cost of this algorithm is great for both the operations number and memory requirements.

- Segregated algorithm: the equations are solved sequentially. A special dealing is required to establish the inter-equation coupling system. SIMPLE [72], PISO [71] and their derivatives are the most common approaches to treat with the coupling of the inter-equation in the pressure-velocity system. Commonly, these are applied by OpenFOAM and will be described in the following paragraphs.

The PISO algorithm for transient flows can be described as follows:

- Firstly, the momentum equation is solved. In this stage, the pressure gradient source term is not defined. Thus, the pressure field component is used from the previous time-step instead. This step is called the momentum predictor. The momentum equation solution (Equation (2.44)) provides an approximation of the new velocity field.
- The equation of pressure can be formulated, and the $\mathbf{H}(\mathbf{U})$ operator can be assembled by using the predicted velocities. The pressure equation solution provides the first estimation of the new pressure field. This stage is called the pressure solution.
- Equation (2.46) presents a set of conservative fluxes that is consistent with the new pressure distribution. In addition, the velocity field component should be corrected as a result of the new pressure field. An explicit manner is used to accomplish the velocity correction using Equation. (2.41). This stage is called the explicit velocity correction.

The velocity field can be corrected explicitly by using Equation (2.41) due to having new pressure field. It can be seen by observing Equation (2.41) that the velocity field correction is depending on both the pressure gradient change term $\left(\frac{1}{a_p}\nabla p\right)$ and on the

corrected velocity into the neighbouring cells $\left(\frac{\mathbf{H}(\mathbf{U})}{a_p}\right)$. In this way, the explicit meaning

refers to use the predicted velocity field; \mathbf{U} , in order to calculate $\mathbf{H}(\mathbf{U})$ term in Equation (2.41) and the correction of the velocity is neglected. It is efficiently assumed that the error in the velocity term is a result of the pressure term error. This is not true, so it is important to correct $\mathbf{H}(\mathbf{U})$ term, the new pressure equation should be formulated and repeat the procedure. The loop of PISO algorithm contains explicit velocity corrections and an implicit momentum predictor that followed by pressure solutions. The PISO loop

is repeated until a determined tolerance is achieved. The number of the correction loops in OpenFOAM is specified in the file fvSolution by defining nCorrectors [70,74].

Moreover, a new conservative flux set is available after each pressure solution. Therefore, it would be possible to repeat the calculation of the coefficients in $\mathbf{H}(\mathbf{U})$ term. However, this is not done because it is assumed that the pressure-velocity coupling is more important than the nonlinear coupling, consistent with the momentum equation linearisation. Thus, the coefficients in $\mathbf{H}(\mathbf{U})$ term are stayed constant through whole correction stages and will be changed only in the following momentum predictor.

The SIMPLE algorithm is developed for the steady-state problems. If steady flow problems are iteratively solved, it is not necessary to solve the linear pressure-velocity coupling because the changes between the solutions are no longer small. In this case, the non-linearity of the coupling system becomes more significant because the time-step is much larger.

The SIMPLE approach by Patankar [72] is expressed for the following facts:

- The momentum equation (Equation (2.44)) is used to obtain the velocity field approximation. The pressure gradient is calculated by the pressure distribution from the previous time-step. The momentum equation is under-relaxed with the velocity based under-relaxation factor α_U .
- The pressure equation is formulated and resolved to find the new pressure distribution field by solving the mass equation (2.45).
- A set of conservative fluxes is obtained using Equation (2.46). It would be essential to resolve the pressure equation to obtain a better approximation of the corrected pressure field. However, the non-linearity effects are more significant than in the transient case. It is enough to find the pressure field approximation and calculate the $\mathbf{H}(\mathbf{U})$ coefficients using the new conservative fluxes set. Therefore, the pressure solution is under-relaxed in order to take the correction of the velocity field into account.

$$p^{\text{new}} = p^{\text{old}} + \alpha_p (p^p - p^{\text{old}}). \quad (2.47)$$

In which:

p^{old} represents the pressure field component that used in the momentum predictor.

p^{new} is the pressure field approximation that will be used in the next momentum predictor.

p^p indicates the pressure equation solution.

α_p is the pressure based under-relaxation factor, ($0 < \alpha_p < 1$).

According to Perić [75], the under-relaxation procedure analysis is discussed for the second corrector in the PISO algorithm, $\alpha_U = 0.8$ for momentum, and $\alpha_p = 0.2$ for the pressure are recommended.

In addition, the solution of the PIMPLE algorithm merged between both algorithms is used in `pimpleFoam` and `pimpleDyMFoam` solvers in Chapter 3 of this thesis. It can be summarised as follows:

- Momentum predictor: the momentum equation (2.44) is solved using the pressure filed from the previous time-step since the exact pressure gradient is not known.
- Pressure solution: the solution of the discretised form of the equation (2.45) is obtained using velocities found in the previous time-step, and this will be the first estimation of the new pressure field.
- Velocity correction: the velocity field can be explicitly corrected using equation (2.46) by applying the new pressure field.

2.3 Fluid-Structure Interaction Problems in OpenFOAM and Foam-Extend

OpenFOAM approaches for solving the interaction between the structure and the fluid were presented by Hua-Dong Yao in 2014 during the third occasion of the “CFD with OpenSource software” course [76].

The approaches of FSI simulation are also discussed in OpenFOAM course presentation by Tuković and Jasak [77]. The methods are split into monolithic and partitioned approaches. The solving procedure of the loosely and strongly coupling of the segregated approach is presented sequentially in Figure 2.7 from solid to fluid in every time-step.

Michler's *et al.* study [78] discussed the contrast between the numerical simulation for the monolithic and the partitioned processes of the FSI problem. In terms of accuracy computational cost, and stability, the two approaches were contrasted by using a simple numerical piston in the experimental benchmark.

Although the monolithic approach has been widely used in the past, most researches agree that it is impractical due to not only it is hard to implement but also it is a challenging task to maintain the solver up-to-date with the latest modification in each various research field [79-80]. Therefore, the partitioned scheme is developed to be the most popular approach in different research fields.

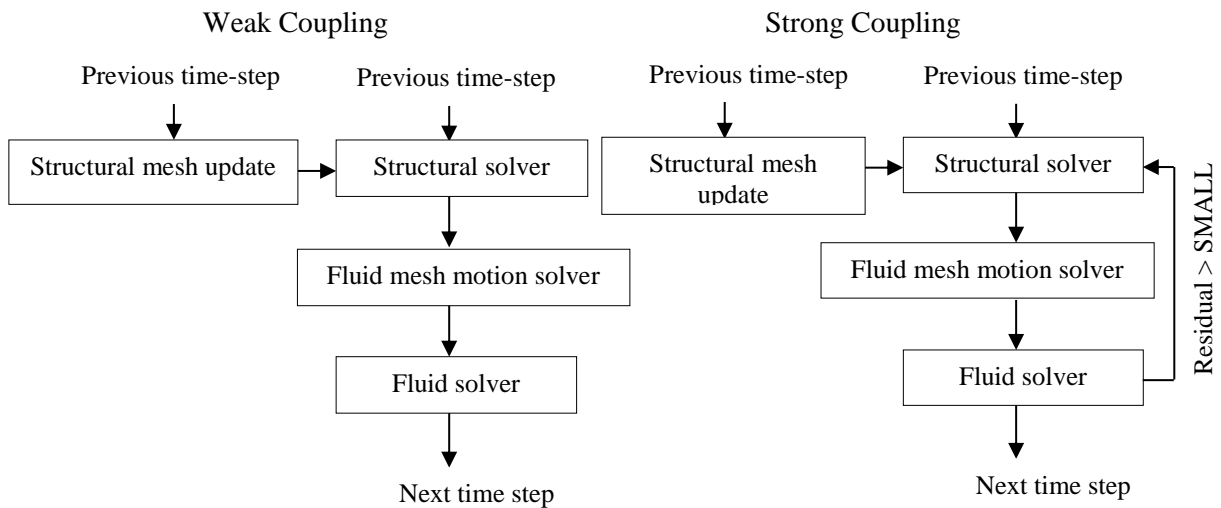


Figure 2.7: Solving process of FSI using partitioned approach with weak and strong coupling [76]

The FSI package has applied the partitioned approach for fluid-solid coupling in foam-extend. Here, there are two separate solvers for fluid and structure are required and a coupling algorithm to couple fluid and solid solvers at their interface in time and space by transferring data between that two solvers by using interpolation techniques. These two separate solvers can be developed independently thus making the partitioned scheme simpler and more agile.

The partitioned approach provides an implicit coupling method that is suitable for strong interaction instead of the explicit coupling that only accounts for the weak interaction. The major distinction among the explicit (weak) and implicit (strong) coupling is

described that the second has an outer (external) loop in solution procedure. This external loop is called fixed-point or sub-iterating.

In general, after each time-step, the mesh deformation (displacement) should be consistent with applied dynamic and kinematic conditions at the fluid-solid interface. This loop is called a sub-iteration or a fixed-point iteration. The difference between the two partitioned coupling approaches is demonstrated in Yao's work [76]. It is essential to highlight that the Aitken adaptive under-relaxation technique and IQN-ILS are utilised to accelerate the coupling procedure in this work.

2.4 Finite Volume Method for Fluid-Structure Interaction with Large Structural Displacements

A natural platform for fluid-structure interaction (FSI) in OpenFOAM is a generic Computational Continuum Mechanics (CCM) library in which both fluid and solid solvers exist. It can deal with the structural analysis and fluid flow simulation for the fluid-structure interaction (FSI). There is no further need for multithreaded simulation of software for software coupling as doing a simulation into a single software. OpenFOAM solvers and discretisation methods share the matrix support and the base mesh, and furthermore different implementation of mesh-to-mesh mapping tools are already used for further simplifying the related problems.

OpenFOAM utilised for building self-contained FSI solver that simulate the interaction among St. Venant-Kirchhoff elastic solid and an incompressible Newtonian fluid. Incompressible Navier-Stoke equations are used to model the fluid flow in Arbitrary Lagrangian- Eulerian (ALE) formulation (Section 2.7) whereas the large structure deformation is demonstrated by the geometrically nonlinear momentum equation in the Lagrangian formulation.

Spatial discretisation is performed for fluid and solid models by using the second-order accurate FVM, where the flow model is discretised on moving mesh [84-85], while the structure model is discretised on the fixed mesh and the displacement is updated from the previous time-step. The deformation of the fluid-solid interface is accommodated by using the automatic mesh motion solver [86-87].

Coupling among these two models is performed by utilising a strongly-coupled staggered algorithm in which the force is stimulated into one direction and the displacement in the opposite one.

According to Farhat *et al.* [88], the interface displacement predictor and force corrector is designed in order to preserve the coupled solution second-order temporal accuracy.

The surface mapping tools are also provided by OpenFOAM. It is used for the data mapping among surfaces directly or by using the second order interpolation. A numerical experiment conducted for demonstrating the density ratio of fluid-solid limit on which the strongly coupled algorithm is required [89].

2.5 OpenFOAM Library for Fluid-Structure Interaction

In the last twenty years, the finite volume method (FVM) has been established to be an alternate to the finite element method (FEM) which was widely used for the numerical stress analysis. The finite volume (FV) development, based on the stress analysis models, gives a possibility to solve the coupling problems of fluid-structure interaction (FSI) by using numerical approaches [90]. In this section, OpenFOAM library for partitioned fluid-structure interaction solver will be discussed.

This library contains three classes which are `flowModel`, `stressModel` and `fluidStructureInterface`. Both fluid and structure solvers are derived from `flowModel` and `stressModel` classes. In addition, the exchange information between the fluid and the solid solvers at the interface is defined by virtual functions. For Newtonian incompressible fluid, `icoFoam` solver is used for unsteady laminar flow.

The Lagrangian formulation is the basis of the structural solver's implementation in which `unsTotalLagrangianStress` solver for the total displacement and `unsIncrTotalLagrangianStress` for incremental displacement.

The `fluidStructureInterface` class is used for both weak and strong coupling for fluid and solid solvers. Fixed-point iteration scheme for the strong coupling will be performed. It is used with the convergence acceleration by either the Aitken under-relaxation or by the

interface quasi-Newton procedure with an approximation for the inverse of the Jacobian from least squares models (IQN-ILS) [79]. Furthermore, parallelisation of the fluid/solid coupling is accomplished by using the fluid-solid interface that presented by a couple of the general grid interface (GGI) zone to zone interpolation and the global face zones.

In addition, the latest foam-extend library for the FSI provides some numerical techniques which were not available in the previous one until now for general use. High accuracy of the face and cell displacement gradients are required in the finite volume stress analysis. In order to calculate gradients of cell centre in OpenFOAM and foam-extend, one can use either least-square method or cell-based Gauss method. The vertex-based Gauss method is known to outperform the least-square method, but it carries on the vertex field accuracy. It is used for polyhedral finite volume mesh calculation of face centre and cell centre. Philip Cardiff [91] noticed that the vertex field calculation is not good enough by using the simple weighted averaging. Thus the new parallel procedure called cell-to-vertex interpolation is implemented. That is based on both the linear shape function and least-square method.

Moreover, a new corrected scheme is performed for the face centre discretisation in the normal direction. This implemented scheme corrects skewness and non-orthogonality. In temporal discretisation of stress analysis, second-order displacement derivative is employed. Finally, the backward scheme of second-order accuracy is used to allow flow and structure models to discretise in time.

The fluid-structure interaction library numerical aspects are described by Tuković and Tuković *et al.* [92-94]. The FSI solver in the FSI package in the extend project named `fsiFoam` is used for strong fluid-solid coupling along with the implicit partitioned approach. In this solver, the laminar flow of incompressible fluid is used the PISO algorithm.

After creating a fluid and structural mesh, the next step is to set the case folder for the simulation. Two separate directories are required inside the original case folder, called `fluid` and `solid`. Both folders require all folders and files for the simulation. Additionally, extra properties files are required in the fluid directory, which include all necessary information for the mesh motion algorithm and the FSI. Furthermore, the faces that are belonging to the fluid-structure interface need to be explicitly defined for both fluid and

solid grids. Thus, the GGI utility has the ability to determine which cell faces have to interpolate and to where. This is by applying the utility tool, called setSet. This allows the large selection, manipulation, and handling of point, cells and surfaces within the computational domain.

Moreover, the dynamic mesh is applied due to the displacement of the fluid-solid interface. In addition, when the fluid-structure interface moves, the internal fluid grids adjust their position. Figure 2.8 illustrates the algorithm of the fsiFoam solver in foam-extend tool.

2.6 Partitioned Solver for Strongly Coupled Fluid-Structure Interaction

The monolithic and partitioned approaches are two main numerical methods used for solving FSI problems. The complete non-linear system in the monolithic approach regarding the solid displacement equation and fluid flow are discretised and simultaneously solved in the space and time using the similar manner [95-101]. Such a direct approach or fully coupled approach is remarkable due to its highly stable and robust for strong fluid-structure interaction [102-103]. In this way, the monolithic approach requires more coding and indicates less modularity as compared to the partitioned method that includes structure and flow equations solved through discretisation method and independently using suitable algorithms [104-109].

However, the fluid and solid equations are solved separately in the partitioned approach, and the coupling becomes limited to the fluid-solid interface [45]. For this reason, an iterative algorithm should be utilised for handling communication between the flow and structural solvers and for enforcing equilibrium onto the fluid-structure interface. This indicated that the fluid flow and the structural deformation are successively solved in an iterating loop until the fact that the difference among the structural and flow solution is smaller compared to the convergence criterion such as the interface displacement.

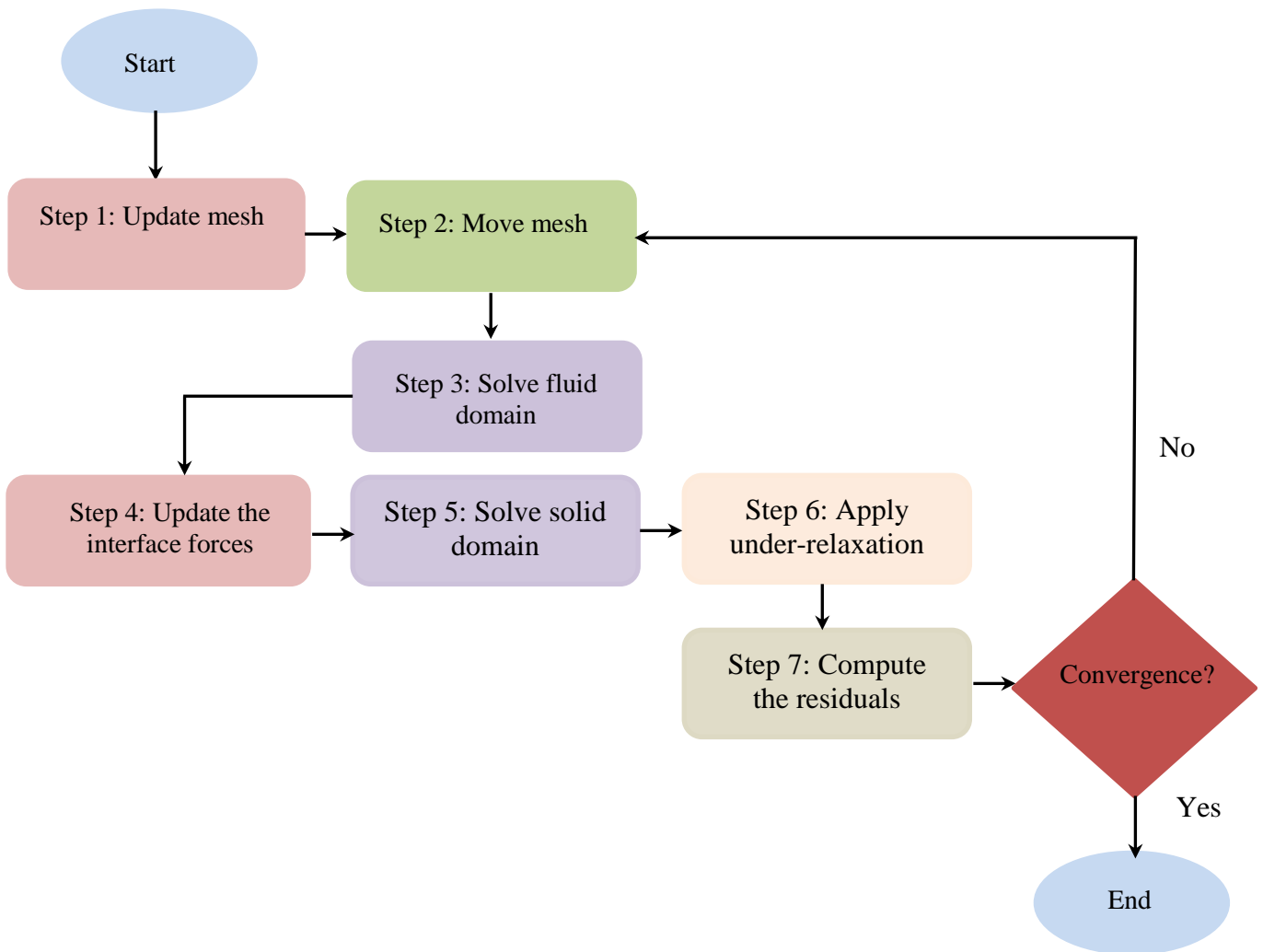


Figure 2.8: Flowchart of algorithm of fsiFoam solver

Their coupling procedure that is being used in this work is the block Gauss-Seidel method which also called fixed-point method [41,110-111]. Several studies have presented that the fixed-point method usage along with the dynamic under-relaxation is easy to implement into the partitioned approaches and it is highly efficient as compared to others [105-106,109-110,112-115]. If the relaxation value of the parameter is not well selected, then it might cause the divergence. This occurs due to the high-density ratio of fluid/solid or low solid stiffness occurring when the coupling among the structural deformation and fluid flow is strong. The Aitken under-relaxation method can be accelerated and stabilised the fixed-point iteration where the parameter of relaxation is adopted by using two previous iteration methods at each iteration [105,110,116].

The computational domain of fluid flow in the fluid-structure interaction problems is deformed along with the fluid-structure interface displacement. Therefore, the Navier-Stokes equations on that deforming mesh can be solved by the Arbitrary Lagrangian-Eulerian (ALE) formulation [64,117-119].

The elastic structure deformations are solved by considering the simple linear formulation through the constitutive model for Hookean structure [67,99]. Furthermore, it is appropriate to use nonlinear formulation where St. Venant-Kirchhoff constitutive model is used for structural analysis for larger displacement of the structure with high rates of strain [65-66,106]. Jasak and Weller [67] discretised the problem of linear elastic deformation as compared to the finite element method for the computational solid mechanics [120-122] by optimised the finite volume method in the Lagrangian formulation. Thus, the diffusion terms are discretised explicitly, and convection terms are implicitly discretised which accelerates the final solution convergence as given in Equation (2.8).

The third and last solver deals with internal mesh motion that might be computed by using several numerical procedures. Such approaches depend on rotation, translation motion or on both together as in [80,87,111,123-126]. The dynamic mesh motion solver is called *velocityLaplacian*. It calculates the points displacement in the computational mesh with a velocity that defined as the initial and boundary conditions of the Laplace smoothing equation [80,87]. This solver takes the unknown equation of the mesh motion in the internal fields and solves the boundary condition of the mesh motion at the fluid-solid interface. The good mesh quality is maintained by using the variable mesh diffusivity especially near moving boundary [45]. The diffusivity is defined to calculate the mesh stiffness to the solver of motion depends on the inverse distance from the fluid-solid interface.

2.7 Arbitrary Lagrangian Eulerian Mapping

In fluid mechanics, there are two algorithms for describing the motion of the fluid particles, the Lagrangian and the Eulerian formulation.

The Lagrangian algorithm is this in which each computational node of the mesh is associated with one or more fluid particles and follows the particle(s) motion during the simulation time. This formulation gives the possibility to track free surfaces or moving interfaces during the simulation time. However, it is unable to follow large distortions in the computational mesh [119].

On the other hand, the Eulerian formulation employs a computational domain that is fixed in space and enable large distortions in the motion at the expense of the interface and the tracking boundary [119].

For a wide range of applications, both descriptions have positive and negative characteristics since it is frequent that the situation demands positive aspects from both algorithms to be presented in a single computational problem. In this case, working with fluid-structure interaction, a solid domain has a displacement as a function of the original mesh and the flow domain has a solution with respect to the present grid, the Euler description. The no-slip condition is applied to the deformable domain like the Dirichlet boundary condition, which provides the additional condition that the cells of fluid near the interface, that share face with the solid cells, have the same velocity as the solid cells that they share a face with. Thus, the fluid grid points move with the solid points on the coupling interface.

The arbitrary Lagrangian-Eulerian (ALE) formulation [64,119] is widely used in the description of fluid-structure interaction that involves the arbitrary deformation of the boundary to avoid a drag and static pressure issues in the thin boundary layer between the fluid and the solid [117,127-129]. The Lagrangian description solved the convective flow that is imposed by the motion of the interface which is near the moving structure. The equations tend to be the Eulerian description because of the mesh velocity $\mathbf{u}_{m,f}$ becomes zero away from the structure.

The fluid-solid interface Γ_0 forms the solid displacement $\mathbf{d}_s^{\Gamma_0}$ from the fluid domain displacement $\mathbf{d}_{m,f}$ toward to the domain of the internal fluid reference $\Omega_{f,0}$. Hence, $\mathbf{d}_{m,f} = \mathbf{d}_s^{\Gamma_0}$ on Γ_0 and $\Delta \mathbf{d}_{m,f} = 0$ in $\Omega_{f,0}$. Crosetto *et al.* [98] define the ALE mapping as

$$\chi(t): \Omega_{f,0} \rightarrow \Omega_{f,t} \tag{2.48}$$

$$\mathbf{x}_0 \mapsto \chi_t(\mathbf{x}_0) = \mathbf{x}_0 + \mathbf{d}_{m,f}(\mathbf{x}_0).$$

The velocity of the fluid domain can be defined as

$$\mathbf{U}_{m,f} = \frac{\partial \mathbf{d}_{m,f}}{\partial t} \Big|_{\mathbf{x}_0} = \frac{\partial \chi_t}{\partial t} \Big|_{\mathbf{x}_0}. \quad (2.49)$$

The Navier-Stokes equations in the ALE formulation will be satisfied in the updated fluid domain $\Omega_{f,t}$ thus

$$\nabla \cdot \mathbf{U}_f = 0, \quad (2.50)$$

$$\frac{\partial \mathbf{U}_f}{\partial t} + (\mathbf{U}_f - \mathbf{U}_{m,f}) \cdot \nabla \mathbf{U}_f = -\frac{\nabla p}{\rho_f} + \nu_f \nabla^2 \mathbf{U}_f, \quad (2.51)$$

where the convective term is $(\mathbf{U}_f - \mathbf{U}_{m,f})$. The setting of $\mathbf{U}_{m,f} = \mathbf{U}_f$ is given the Lagrangian description, and the Eulerian description will be obtained when $\mathbf{U}_{m,f} = 0$ with $\mathbf{U}_{m,f}$ mesh velocity for the fluid domain.

Therefore, using the ALE formulation allows the computational mesh to move and preserve the cells orthogonality while keeping track of the free surfaces and moving boundaries. Briefly, this algorithm enables the deformation of fluid mesh in response to the structural mesh deformation. Thus, apply the traction vector on the structural domain and then impose the displacement of the flow boundary, re-mesh and interpolate the pressure and velocity on the new mesh.

2.8 Fluid-Structure Coupling and Boundary Conditions

Certain conditions should be followed in order to establish the equilibrium of the fluid-structure interface Γ ; which is the continuity of displacement \mathbf{d} , equilibrium traction $\boldsymbol{\tau}$ and mesh velocity \mathbf{U} .

$$\mathbf{d}_s^\Gamma = \mathbf{d}_f^\Gamma, \quad (2.52)$$

$$\mathbf{U}_s^\Gamma = \mathbf{U}_f^\Gamma, \quad (2.53)$$

$$\boldsymbol{\tau}_s^\Gamma + \boldsymbol{\tau}_f^\Gamma = 0. \quad (2.54)$$

The variables at the fluid-structure interface are denoted by the superscripts Γ where Equations (2.52) to (2.54) must be satisfied.

Furthermore, the traction $\boldsymbol{\tau}_f$ is given by $\boldsymbol{\tau}_f = p_f \mathbf{n} + \mu_f \nabla \mathbf{U}_f \cdot \mathbf{n}$ that represents the sum of the pressure and the viscous force [127-128]. Equations (2.52) and (2.53) indicate the fluid-structure interface quantities of the continuity of displacement and the velocity. While Equation (2.54) represents the action-reaction principle which is dual quantities equilibrium of the pressure and viscous stress.

The viscous forces and pressure of fluid flow are transferred toward solid at the fluid-structure interface as boundary conditions to compute the stress field and the displacement for the structural solver. The mesh motion is solved by the displacement and velocity that is used as the input. Moreover, in the solid domain, the stress field is transferred toward the flow field on the fluid-structure interface as a boundary condition.

Initially, both the solid and fluid domains are at the rest position and no-slip boundary conditions implemented on the walls or the fluid-solid interfaces. Moreover, the outlet conditions are set as zero for both Neumann for velocity and pressure, and the profile of the parabolic velocity is set at the inlet in the flow domain. It is important to note that the defined boundary conditions depends on the studied cases in Chapters 3-5.

2.9 Numerical Method in FSI Library

Moving solid boundaries with fluid-structure interaction problem require the third coupled solver for automatic internal mesh motion. Consequently, the dynamic mesh motion solver deforms internal fluid domain while maintaining deforming mesh validity and quality. The fluid-structure interface displacement in the result of the structural solver used for the mesh motion solver as a boundary condition. Many mesh motion solver can be found into open literature [80,87,117,127-128] like Laplace smoothing [130].

The mesh deterioration and distortion occur when using the Laplace equation because the mesh point movement will be large enough near to moving boundary [67,130]. The

quality of mesh for large boundary translation is maintained by the Laplace face decomposition method along with an inverse method of quadratic diffusion coefficient [87]. This method is used, and the diffusion coefficient depends toward on cell distance for the nearest moving boundary.

2.10 Implicit Coupling Approach with Second-Order Predictor

A linear patch-to-patch interpolation [66-67,87] used for exchanging the information between the fluid and structure domains at the fluid-structure interface. Furthermore, a finite volume method is utilised for both the fluid and solid solvers where the data are stored at the centre of the element of each field.

The communication among the three different solvers, i.e., mesh motion solver, fluid flow solver, and structural deformation solver results in using the partitioned approach. The dynamic relaxation [110,116] along with iterative implicit fixed-point algorithm is utilised for accurate coupling of different solvers and for enforcing the equilibrium onto the fluid-structure interface. Each time-step executed at the iterative algorithm, where every iteration j , the fields of structure and fluid both solved until the convergence criterion satisfied. The development of the flowchart of numerical procedures represented in Figure 2.9.

Now $\mathbf{d}_{i,j}^\Gamma$ is the displacement interface at the time-step i and the iteration j , the fluid solver F , the mesh motion solver M , and the solid solver S . The solver performance and convergence are improved by the help of interface displacement predictor $\tilde{\mathbf{d}}^\Gamma$ for $j=1$ and for each time-step.

$$\text{Order 0: } \tilde{\mathbf{d}}_{i+1,1}^\Gamma = \mathbf{d}_{i,N}^\Gamma \quad \text{for } i=1$$

$$\text{Order 1: } \tilde{\mathbf{d}}_{i+1,1}^\Gamma = \mathbf{d}_{i,N}^\Gamma + \delta t \mathbf{u}_{i,N}^\Gamma \quad \text{for } i=2 \quad (2.55)$$

$$\text{Order 2: } \tilde{\mathbf{d}}_{i+1,1}^\Gamma = \mathbf{d}_{i,N}^\Gamma + \frac{\delta t}{2} (3\mathbf{u}_{i,N}^\Gamma - \mathbf{u}_{i-1,N}^\Gamma) \quad \text{for } i \geq 3$$

where N denotes that the last iterations number ($j=1-N$). The second step in Figure 2.9 initialised these equations.

The predicted interface displacement is used for moving the mesh as a boundary condition, and then the new internal mesh motion is transferred to the fluid flow solver after obtaining its velocity. After that, the ALE formulation is utilised to solve the fluid flow problem for the pressure-velocity coupling by using PISO algorithm. The viscous stress and pressure are both computed and then transferred as a boundary condition into the fluid-structure interface for the solid solver. The predicted displacement $\tilde{\mathbf{d}}_{i+1,j+1}^\Gamma = \mathbf{S} \circ F(\tilde{\mathbf{d}}_{i+1,j}^\Gamma)$ is executed by the solid solver. Additionally, when the ratio of the fluid density to the solid density is very large and/ or the solid structure stiffness is small, then the fluid flow impact on the solid structure will be important. In this way, the displacement of the predicted interface does not relate to the result $\tilde{\mathbf{d}}_{i+1,j+1}^\Gamma \neq \mathbf{d}_{i+1,j}^\Gamma$. Hence, interface displacement should be used for iterative correction and the iterative approach which can be indicated as:

$$\mathbf{d}_{i+1,j+1}^\Gamma = \mathbf{d}_{i+1,j}^\Gamma + \alpha_{i+1,j+1} \mathbf{r}_{i+1,j+1}^\Gamma \quad (2.56)$$

where the relaxation parameter is $\alpha_{i+1,j+1}$ and $\mathbf{r}_{i+1,j+1}^\Gamma = \tilde{\mathbf{d}}_{i+1,j+1}^\Gamma - \mathbf{d}_{i+1,j}^\Gamma$ is the residual of the interface. The stopping criteria is required by the iterative procedure, then the interface displacement residual for length scaled Euclidian norm $\mathbf{r}_{i+1,j+1}^\Gamma$ as adopted from [106,110,131] is defined for the relative error for the outer-loop:

$$\xi_{i+1,j+1}^\Gamma = \frac{\|\mathbf{r}_{i+1,j+1}^\Gamma\|_{\max}}{\sqrt{n_q}}. \quad (2.57)$$

In which n_q is length of the vector $\mathbf{r}_{i+1,j+1}^\Gamma$. This type of equation is computed into the step that is just before the test of convergence as mentioned in Figure 2.6. The outer-loop iterations stopping value is given as fixed ($\xi_{i+1,j+1}^\Gamma = 10^{-6}$). A new time-step begins and the iterations stop below this value.

In order to re-write the Aitken under-relaxation factor, Irons and Tuck [116] introduced the Aitken method for evaluating the dynamic relaxation parameter $\alpha_{i+1,j+1}$ in every iteration j which is further revisited by Kuttler and Wall [110]. This particular method utilises the previous two iterations for determining the present solution by considering $\mathbf{r}_{i+1,j+1}^\Gamma = \tilde{\mathbf{d}}_{i+1,j+1}^\Gamma - \mathbf{d}_{i+1,j}^\Gamma$ and $\mathbf{r}_{i+1,j+2}^\Gamma = \tilde{\mathbf{d}}_{i+1,j+2}^\Gamma - \mathbf{d}_{i+1,j+1}^\Gamma$, thus the relaxation parameter is

$$\alpha_{i+1,j+1} = -\alpha_{i+1,j} \frac{(\mathbf{r}_{i+1,j+1}^\Gamma)^\top \cdot (\mathbf{r}_{i+1,j+2}^\Gamma - \mathbf{r}_{i+1,j+1}^\Gamma)}{\|\mathbf{r}_{i+1,j+2}^\Gamma - \mathbf{r}_{i+1,j+1}^\Gamma\|^2}. \quad (2.58)$$

The first under-relaxation cycle $\alpha_{i+1,1}$ for the relaxation parameter cannot be calculated as required by the previous two iterations. Furthermore, the previous time-step $\alpha_{i,N}$ of the last relaxation parameter might be too small for using as the first value for next time-step. It is suggested [110,116] to use α_{\max} as a constrained parameter

$$\alpha_{i+1,1} = \max(\alpha_{i,N}, \alpha_{\max}). \quad (2.59)$$

The first step in Figure 2.9 initialized that previous equation.

The partitioned fluid-structure interaction problems utilised the Aitken acceleration method because it is an easy and efficient way to show and produce accurate results and reduce the simulation time [110,104,114].

2.11 Boundary Conditions Definitions in OpenFOAM

Each of the boundary conditions which is described mathematically in an equation has a physical meaning, i.e., numerical methods have to be translated to the algebraic form. For example, the inlet boundary condition describes the behaviour of the flow by using the appropriate mathematical equation that expresses the physical conditions of the velocity and pressure. These conditions include Dirichlet and Neumann conditions, which are defined to connect to the boundary conditions and the mathematical model. Therefore,

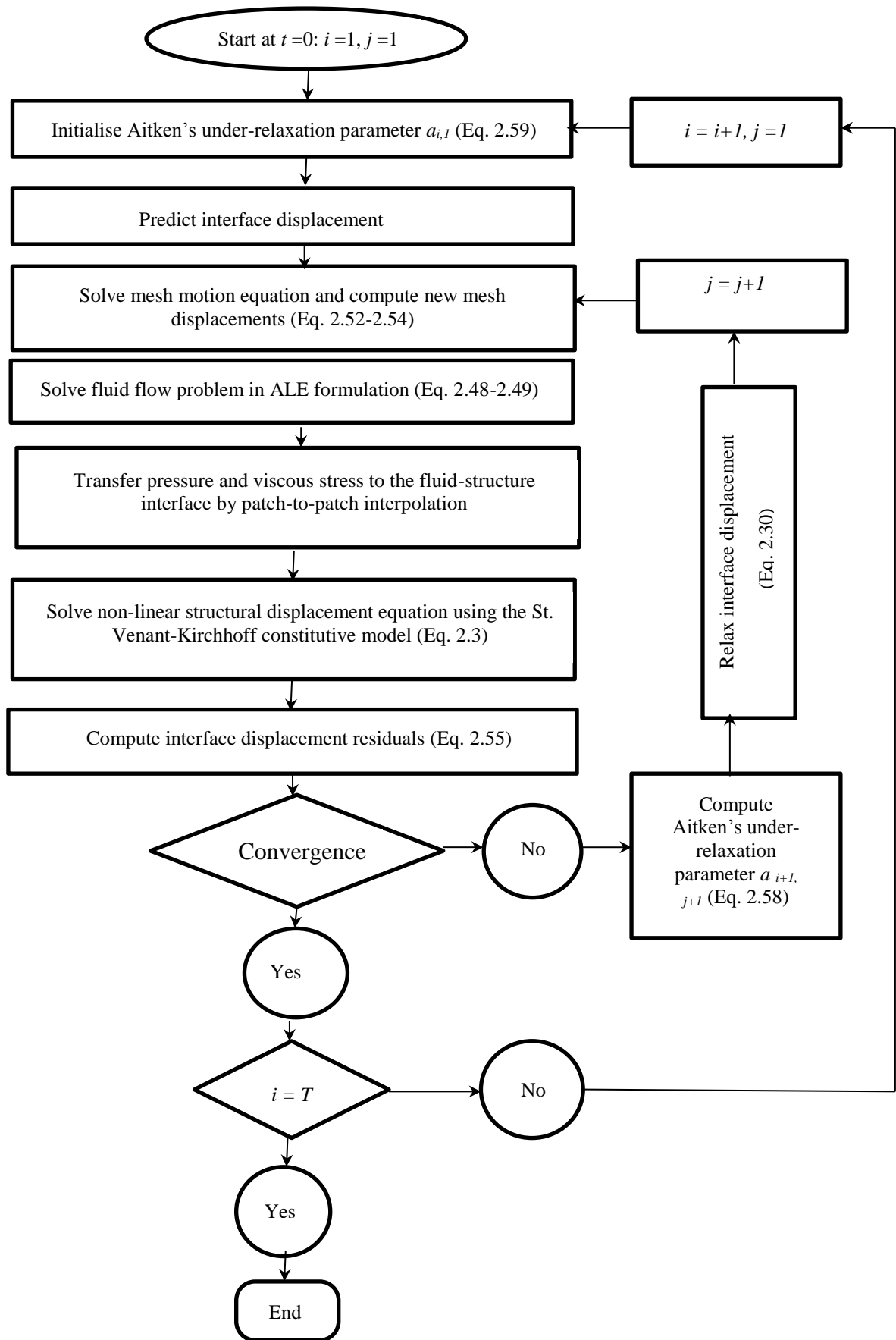


Figure 2.9: Partitioned solver for strongly coupled nonlinear fluid–structure interaction flowchart, i is time- step from 1 to T , and j is Aitken iteration from 1 to N [45].

the mathematical terms or operators such as gradient, divergence and Laplacian will be affected. In OpenFOAM, most of the boundary conditions definition saved into `src/finiteVolume/fields/fvPatchFields` directory. That also includes the main boundary conditions implemented “types” which are stored in `basic`, `derived`, `constraint` and `fvPatchField` sub-directories and the brief description of each will be given in the following paragraphs.

The first one is the `fvPatchField` directory which includes the general boundary conditions definition that represents the basic class. All basic functions and data structures will be defined, used and inherited by the certain classes.

The second is the `basic` directory which includes the basic boundary conditions that are defined mathematically. For instance, of such types and their boundary conditions, Dirichlet “`fixedValue`” Neumann “`zeroGradient` and `fixedGradient`” and Robin “`mixed`” boundary conditions. Moreover, “`coupled`” boundary condition which implemented to couple boundaries; i.e., two boundary patches are coupled together.

The next is the `derived` directory. This contains the derived boundary conditions from Dirichlet, Neumann and Robin boundary conditions and these considered as the simple types specialisations.

Finally, the `constraint` directory is implemented for the geometric boundary conditions which derived from the coupled boundary. In this directory, each of the cells is depending on the corresponding patch cell which treats to the boundary cells as the internal cells [74,125].

2.12 Chapter Summary

The technique of the finite volume discretisation has been described. It uses the control volume of the arbitrary topology, simplifying the mesh generation problem for complex geometries. The discretisation of the spatial and temporal terms based on the face interpolation procedure has been presented in Section 2.2. Moreover, in this section, the discretisation technique for coupled systems of equations has been discussed. The adopted treatment of the pressure-velocity system is based on the SIMPLE algorithm for

steady-state flows, the PISO approach for transient calculations, and PIMPLE algorithm which is combined both SIMPLE and PISO algorithms. In addition, a sequence of procedures for both steady and transient simulations have been summarised. PIMPLE and PISO are only used for this thesis.

A partitioned solver is discussed in this chapter considering strong coupling fluid–structure interaction problems by using the fixed-point implicit scheme with adaptive based on Aitken under-relaxation scheme and IQN-ILS. Finite volume method is applied to discretize the fluid flow and structural displacement in space and time. An Arbitrary Lagrangian–Eulerian formulation is used to solve Navier–Stokes equations as the internal mesh in the fluid flow region deforms with the flexible boundaries. The interaction between the fluid and solid solvers is achieved on their interface. The solver of the mesh motion uses the Laplace smoothing equation with mesh diffusivity. It takes the displacement at the fluid–solid interface as a boundary condition and then solves the mesh motion in the flow domain. This solver called `fsiFoam` which was developed by the open source C++ library for `foam-extend` project.

Chapter 3

A Flow past a Two-Dimensional Cylinder and Vortex-Induced Vibration

The objective of this chapter is to simulate a two-dimensional laminar flow past a circular cylinder and a vortex-induced vibration investigation using OpenFOAM tool. The `pimpleFoam` and `pimpleDyMFoam` solvers are used for stationary and moving cylinders, respectively. These two solvers are using PIMPLE algorithm for pressure-velocity coupling. There are three different cases which are studied in this chapter: flow past a static circular cylinder, flow past a free vibration cylinder and flow past a forced vibration cylinder. The results of the hydrodynamic forces (lift and drag) are discussed for each case. The Strouhal number is also shown in this study.

3.1 Introduction to Flow around a Cylindrical Structure

The flow of a fluid past a cylindrical structure can generate some vortices in the wake. These vortices will be shed later on as shown in Figure 3.1. This phenomenon is called vortex shedding (see Section 3.3 for further explanation).

The phenomena of the vortex shedding might cause the continuous vibration, and this is commonly called vortex-induced vibration (VIV) (see Section 3.5).

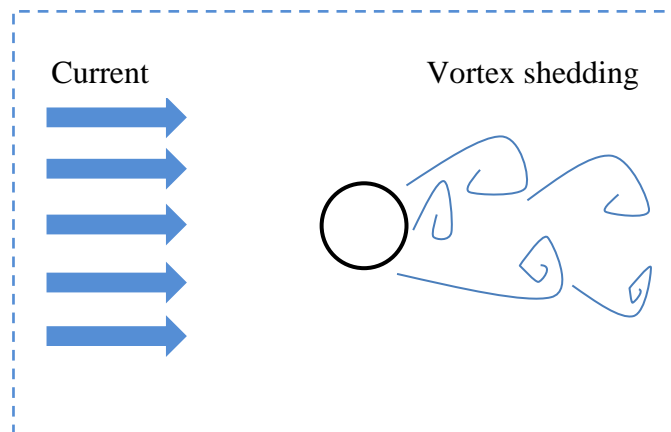


Figure 3.1: Vortex shedding phenomenon behind a cylindrical structure

The observations of the vortex shedding around the cylinder may result in better understanding the VIV as it is the phenomena that generate the VIV on the cylinder. Therefore, the observation of the vortex shedding will be studied for different flow characteristics, which is represented by the Reynolds number (see Section 3.2) [37].

The vortex shedding behind a cylinder could be studied by either experiments or simulations. The time consumption and financial cost of doing experiments are high, and thus simulation plays an important role in design.

3.2 Flow Regimes



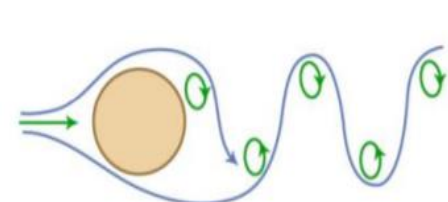
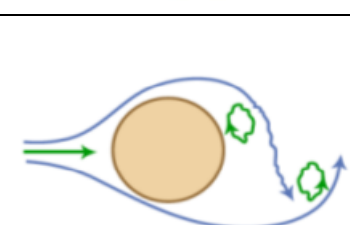
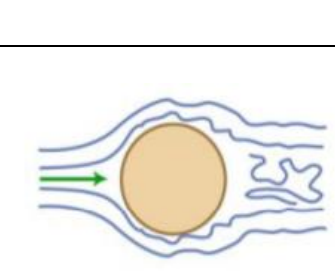
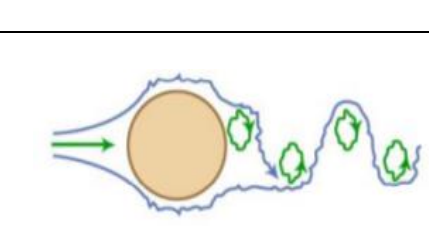
Stokes [133] was the first researcher to coin Reynolds number (Re) as a dimensionless number whose ratio measures the inertia to viscous fluid force. The Reynolds number has been widely used and many flow regimes will result in considerable Reynolds number changes. The Reynolds number changes create flow separation in the cylinder wake region and are named vortices. At low Reynolds number values ($Re < 5$), no separations occur. However, when the Reynolds number increases, then the separations occur, and flow becomes unstable and appears as vortex shedding phenomena, and that can be shown in Table 3.1. A separation behind a circular cylinder surface from a fixed pair of symmetrical vortices starts to appear as Re increases from 5 to 40 [134].

The separation at the cylinder surface, as Re gradually increases is characterised by transitional successions from laminar to turbulence in different regions. These regions include wake, boundary layer, and shear layer transitions. Usually, increase in Reynolds number causes the first transition to occur, between 200 and 300 Reynolds number, in the wake region as turbulence develops gradually spreading laterally across the wake. The second shear layer transition occurs between 300 and $3 \times 10^5 Re$.

The local effects of Reynolds number in the boundary layer regime can be categorised into three. The first has a smaller Reynolds number ($< 3 \times 10^5$) and is known as the subcritical range. The Re in this range features laminar boundary layer which is in the early stages of separation layer transition and has a fully turbulent wake. Second is the critical range with Re range from 3×10^5 to 3×10^6 . In this range, gradual separation of the boundary layer begins with reattachment of turbulent, development of separation bubble,

and eventually separation of the turbulent boundary layer. Increase in Reynolds numbers beyond 3×10^6 results to a supercritical region which is characterised by a transition from the laminar boundary layer to turbulent, then finally attaining the separation point.

Table 3.1: Flow regimes around a smooth circular cylinder in steady current. Figure by MIT OCW [135]

No.	Flow Regimes	Description	Range of Re
a.		Creeping flow - no separation	$Re < 5$
b.		Stable vortices pair in the wake	$5 < Re < 40$
c.		Laminar vortex street (Von Karman vortex street)	$40 < Re < 200$
d.		Laminar flow boundary layer up to the turbulent wake	$200 < Re < 300$ Transition to turbulence $300 < Re < 3 \times 10^5$
e.		Boundary layer transition to turbulence	$3 \times 10^5 < Re < 3 \times 10^6$
f.		Turbulent vortex street, but the wake is narrower than in the laminar case	$3 \times 10^6 < Re$

Computational simulations using numerical methods are mostly implemented at the subcritical Reynolds number range because at this transition stage as it is easy to resolve a relatively thick laminar flow for the attached boundary layer using grids [8]. As the Reynolds number increases to critical and supercritical regions, the boundary layer becomes relatively thinner by almost six times compared to the subcritical region, making it difficult to resolve for the attached turbulence [136]. This perhaps explains why there are fewer CFD studies for flow over a circular cylinder at higher Re compared to those of low Re.

It is critical to note that in OpenFOAM the Reynolds number is not defined itself, but it is calculated when specifying the kinematic viscosity; $\nu = \mu/\rho$. Thus the Reynolds number is calculated for simulation as

$$\text{Re} = \frac{\rho DU}{\mu}. \quad (3.1)$$

Where ρ is the fluid density, D is the cylinder diameter, U is the flow velocity, and μ is the dynamic viscosity.

3.3 Vortex Shedding

Vortex shedding or flow-induced vibration (FIV) is a phenomenon arising from FSIs and continues to pose challenges in a broad range of engineering fields. They include valves, piping systems, heat exchange tubing, stream generator tubing and other marine and civil engineering structural components. The FIV loads significantly affect normal operational functions of these critical engineering components. Most research interests in this area focus on finding solutions that eliminate or mitigate the FIV phenomena.

The phenomenon of vortex shedding occurs when the stable vortex pairs are exposed to very small disturbances and evolve into unstable at Reynolds number greater than 40. This is an important feature for flow past a circular cylinder because it is responsible for boundary layer separation. The separation is attributed to an imposed adverse pressure gradient as the flow environment undergoes divergent geometry on other side of the cylinder [134,137-139]. The divergent flow geometry generates vortices behind a circular

cylinder causing a periodic shear layer separation. The FIV problem occurs in almost all engineering applications with severe to catastrophic damages. It is thus a major concern to engineering designers in applications involving high fluid flow velocities and centrifugal compressor projects, e.g., marine cables, heat exchanger tubes, flexible risers used in the production of petroleum, chimney stacks, bridges, transmission lines, etc. [140].

Therefore, the shear and the boundary layers are formed over the cylinder [17]. The boundary layer consists of a vorticity, and this is fed into the shear layer that is formed on the downstream of the separation point (refer to Figure 3.2a). In addition, this leads to rolling up the shear layer into the vortex with an identical sign to the incoming vorticity (refer to Figure 3.2b) [134].

From the previous section, it could be observed that the vortex shedding could occur at a certain frequency. This is known as the vortex shedding frequency, f_v and it is given by

$$f_v = 1 / T_v, \quad (3.2)$$

where T_v is the vortex shedding period.

The vortex shedding or FIV is represented by the Strouhal number (St), which is a vortex shedding parameter indicating the conversion ratio from kinematic flow energy to oscillatory flow energy [141-142]. This is expressed as:

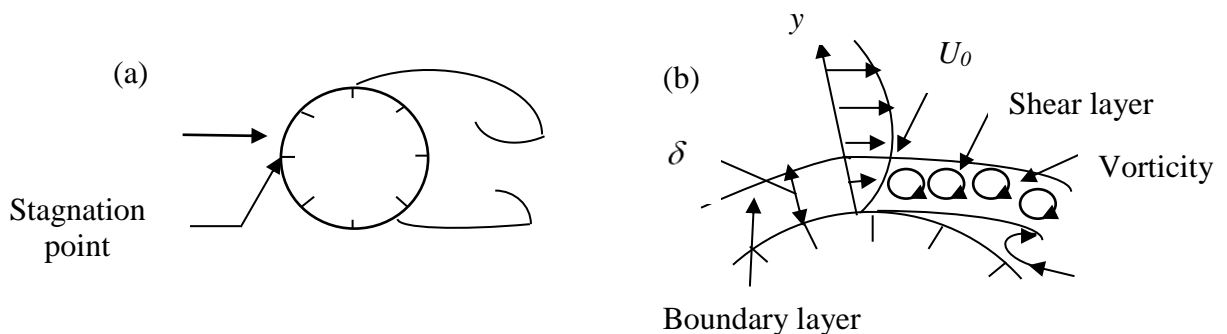


Figure 3.2: The shear layer of the flow over a cylinder

$$St = f_v \cdot D / U \quad (3.3)$$

According to Koushan [143], the Strouhal number reveals that there is a relation between the fixed cylinder or pipe vortex shedding frequency (f_v) and the value of the free stream velocity divided by the diameter of the cylinder or pipe (U/D). The relation between the Reynolds number and Strouhal number for a cylinder is presented in Figure 3.3.

3.4 Hydrodynamic Forces

Fluid flow past a cylinder creates vortex shedding behind the cylinder and these forces lead to the cylinder to vibrate in both the cross-flow and inline directions [143-147,149]. According to Feng's study [148], parameters such as flow velocity, nondimensional damping, non-dimensional mass, cylinder natural frequency, and support structures were found to influence responses of the elastically-mounted cylinder in free oscillation. This study revealed that when the vortex shedding frequency is very close to the natural frequency that would cause the cylinder to vibrate.

The vortex shedding phenomena are due to the hydrodynamic forces that are acting on the cylinder. The force in the cross-flow direction contains the lift force and added mass. The lift force results in the pressure differences at the top and the bottom of the cylinder. However, the added mass appears due to the body accelerating and deflecting volume of the surrounding fluid. Therefore, if the structure does not move then the added mass does not occur. On the other hand, the hydrodynamic force that occurs in the inline direction results in the drag force. Drag force appears because of the pressure differences that are induced between the upstream and downstream cylinder faces [143]. Drescher [208] performed an experiment where the drag force (F_D) and the lift force (F_L) are traced and measured from the pressure distribution, and that is cited by Sumer and Fredsøe [134].

According to Sumer and Fredsøe [134]

- The drag force that is acting in the inline direction of the cylinder has periodic changes in the oscillating time through the mean drag.
- In spite of the symmetric flow is caused with respect to the cylinder axis, there exists a nonzero force component with zero mean in the lift force (transverse

direction) and thus this force changes periodically with time.

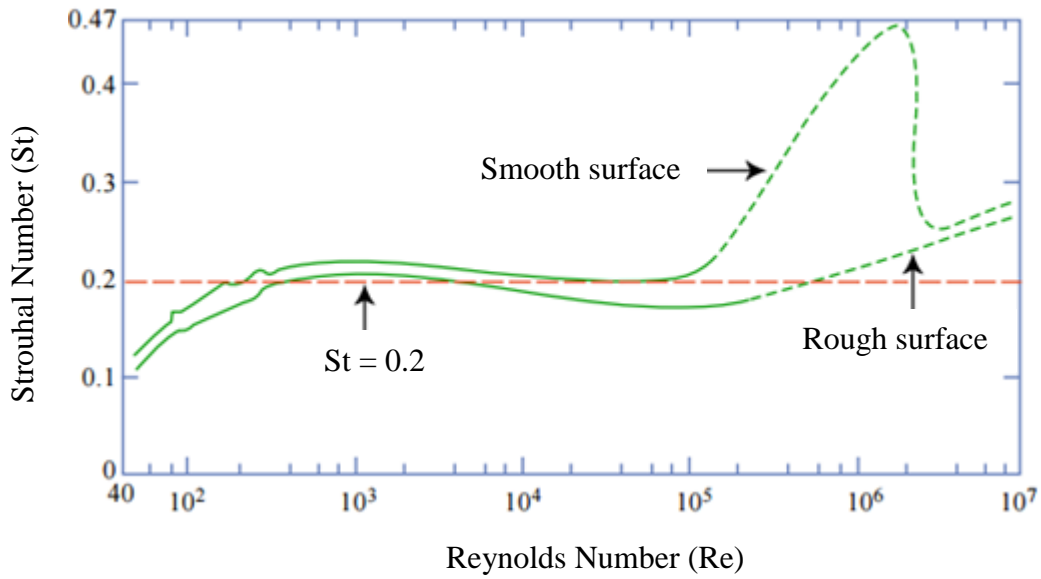


Figure 3.3: Strouhal number – Reynolds number relationship for a circular cylinder. Data from [11,150].
Figure by MIT OCW [135]

Drag, mean drag and lift coefficients (C_D , \bar{C}_D and C_L , respectively) are non-dimensional parameters for the drag and lift forces and are formulated as

$$C_D = \frac{F_D}{\frac{\rho}{2} U^2 L D} \quad (3.4)$$

$$\bar{C}_D = \frac{\bar{F}_D}{\frac{\rho}{2} L D U^2} \quad (3.5)$$

$$C_L = \frac{F_L}{\frac{\rho}{2} U^2 L D} \quad (3.6)$$

where U , D , L and ρ are the flow velocity, cylinder diameter, cylinder length and the fluid density, respectively.

3.5 Vortex-Induced Vibration

In vortex-induced vibration phenomenon, when the vortices are not symmetrically formed around the bluff body, different lift forces are developed on each side of the body; this leads to motion transverse to the flow. This motion changes the nature of the vortex formulation in such a way as to lead to a limited motion amplitude.

Apart from the FIV, vortex-induced vibrations are another major cause of engineering problems arising from FSI. It is caused by inherent self-excitation, self-regulation, and a self-limiting phenomenon that is largely nonlinear in nature [151]. The Tacoma-Narrow Bridge collapse (1940) resulting from wind-induced vibrations is a good example of a structural failure that arises from the vortex-induced vibrations. Therefore, the VIV phenomena have attracted a huge interest in both academia and industry for decades [152].

Generally, the VIV phenomenon is encountered in the hydrodynamic systems where the excitation occurs because of the vortex shedding that comes from bluff bodies. The process of the vortex shedding will create asymmetric pressure distribution over a circular cylinder, then eventually leads to the body movement. The body motion is nonlinear and occurs through a range of frequencies and thus leads to increases in the strength of the shed vorticity. Therefore, this phenomenon can increase the body fatigue and introduce potential damage [36-37,147,153].

Vortex-induced vibrations generate wake regime patterns that are largely dependent on vortex mode and oscillation amplitude [154]. Govardhan and Williamson [140,155] investigated various vortex patterns generated from a rigid-cylinder system in vortex-induced vibrations. The key parameters identified to influence vortex patterns include mass damping, density ratio, and degrees of freedom.

In this chapter, the vortex induced vibration theory for the cylindrical structure will be presented. It presents the solution to the forced oscillation equation and the free vibration equation.

3.6 Dynamics of One Degree of Freedom System and Solution to Vibration Equation

The one degree of freedom system is presented in Figure 3.4, and it is called a spring-mass-damper system. In which the spring has no mass or damping, the mass has no damping or stiffness, and the damper has no mass or stiffness [156-157]. Moreover, a free vibration of an elastically mounted circular cylinder is presented by a vibrating structure description. Additionally, in this thesis, the mass movement is allowed to be in one direction only and based on the vertical vibration of the cylinder is modeled as a one-degree-of-freedom system.

The equation of motion for Figure 3.4 is given by

$$m \ddot{y}(t) + c \dot{y}(t) + k y(t) = F(t) , \quad (3.7)$$

where m is the total cylinder mass, c indicates the linear-viscous damping, k is the spring stiffness, F shows the forces that are acting on the mass points, and y is the vertical displacement of the mass centre of a moving cylinder. The dots indicate the differentiation over y with respect to time. The solution to Equation (3.7) is given by the sum of the particular part (forced response) and a homogenous part (free response) [156]. For the free vibration system, there are no external forces ($F=0$), then the solutions can be differentiated into two conditions: with and without viscous damping.

For free vibrations with non-damped motion, i.e., no external forces applied, the equation of motion follows

$$m \ddot{y}(t) + k y(t) = 0 \quad (3.8)$$

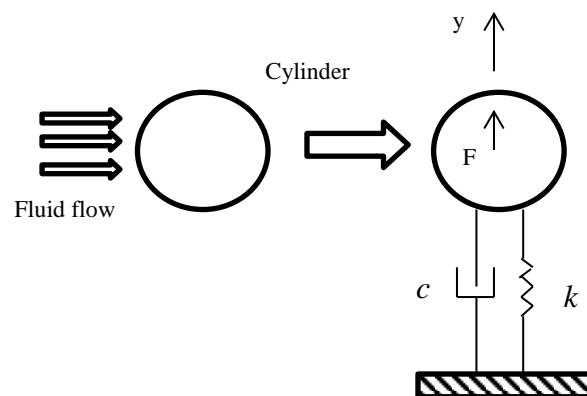


Figure 3.4: One-degree-of-freedom system models in cylinder

The non-damped system vibrates freely at its natural frequency ω_n , where its unit is radians per second, and it is formulated as

$$\omega_n = \sqrt{k/m} \quad (3.9)$$

$$\omega_n \cdot T_n = 2\pi \quad \text{OR} \quad T_n = \frac{2\pi}{\omega_n} \quad (3.10)$$

Then the natural frequency f_n is

$$f_n = \frac{1}{T_n} = \frac{\omega_n}{2\pi}. \quad (3.11)$$

By substituting Equation (3.9) into Equation (3.8) the following is obtained

$$\ddot{y}(t) + \omega_n^2 y(t) = 0. \quad (3.12)$$

The last equation has the following general solution since m and k are both positives,

$$y = A \cos(\omega_n t) + B \sin(\omega_n t). \quad (3.13)$$

In which A and B are determined by the initial condition: $y(0) = 0 \rightarrow A = 0$. Therefore, Equation (3.13) becomes

$$y = B \sin(\omega_n t). \quad (3.14)$$

3.7 Reduced Velocity

The undisturbed flow distance is traveling over one cycle, U/f , which defines the steady vibrations path length per cycle [143]. The reduced velocity, V_R , produces the ratio between the path length per cycle and the body width which is indicated by the cylinder diameter, D , in this work. The reduced velocity is given as follows

$$V_R = \frac{U}{f_0 \cdot D}. \quad (3.15)$$

Where U is the speed of flow, and D is the cylinder diameter, and f_0 indicates the cylinder Eigen frequency in still water.

Feng [148] did an experiment where a D-section cylinder on vertical springs was mounted, Figure 3.5 so that the system has only one-degree-of-freedom and then Feng [148] exposed it to air flow with very small increments. All parameters, the oscillation frequency f_{osc} , vortex shedding frequency f_v , the phase angle θ and the oscillation amplitude A were measured in the experiment.

According to Feng's study [148], the cross-flow vibration starts for the reduced velocity V_R is around 3, and it reaches a peak when V_R is about 5. Afterward, $5 < V_R < 7$, the vortex shedding frequency and Eigen frequency are equal; this phenomenon is called the "lock-in." At this range, the oscillation frequency and the vortex shedding frequency breakdown into the natural frequency system. Finally, the "lock-in" phenomenon is also defined as a response that displays the resonance.

In addition to "lock-in" phenomenon, it is important to note that despite the vortex shedding frequency and the Eigen frequency are equal, the Eigen frequency would be affected by the added mass change, and the vibration itself will affect the vortex shedding frequency.

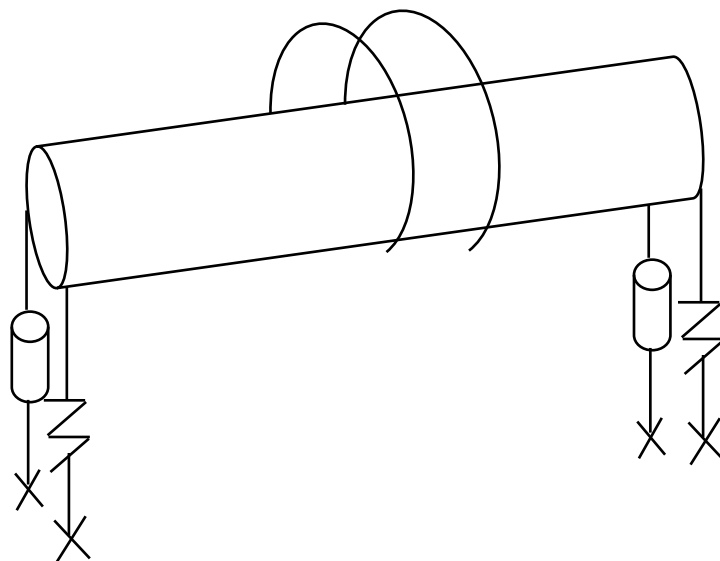


Figure 3.5: Feng's experiment model [148]

Furthermore, the inline vibration is beginning for the reduced velocity value is about 1.5, but it is usually small enough to be un concern.

3.8 Mesh Generation in OpenFOAM

In OpenFOAM, the mesh can be generated using many ways. One of them is using the dictionary file `blockMeshDict` which locates in the `constant/polyMesh` directory. After setting up the `blockMeshDict` file, the mesh will be generated by launching the command `blockMesh` in the terminal. Finally, it will interpret the dictionary to create the mesh and then write out the data of the mesh to boundary, cells, faces and points files into the `blockMeshDict` [158-159].

The `blockMesh` concept is to analyse the domain geometry into three-dimensional hexahedral blocks. The block edges can be set up as arcs, straight lines, or splines. There are eight vertices for each block one at each hexahedra corner. Later, they can be written as a list and assessed using their label by OpenFOAM [158-159].

3.9 Test Cases

3.9.1 2D Example of Flow past a Stationary Circular Cylinder

Flow over a stationary cylinder is subject to parameters such as free stream velocity, surface roughness, fluid density, etc. [160]. As previously mentioned in Section 3.2, the different flow regimes depend on the Reynolds number range. For instance, flow occurring between $0 < Re < 5$ show reattached streamlines similar to detached hence near balanced upstream and downstream pressure, a phenomenon referred to as “Stokes flow.” The next band $5 < Re < 40$ feature wakes and separation that is symmetrical to the fluid flow axis. Reynolds numbers higher than 40 leads to unsteady flow because the vortices that were initially in fixed positions begins shedding in an alternate but regular fashion, a condition known as “Von Karman Vortex Street” [161].

Simulation of flow past a stationary cylinder is mainly confined to numerical methods for two-dimensional flow at extremely low Reynolds numbers. Experimental simulations at

low Reynolds number provided detailed insights of vortex dynamics and flows instability [13,17,23,137,144,153,162].

3.9.1.1 Laminar Flow Domain

In this case, the circular cylinder with a diameter ($D = 1\text{m}$) is placed in the channel with length $20D$ and width $8D$. The flow velocity is 1 m/s at the inlet and is zero gradient at the outlet. Zero pressure also defined at the outlet as shown in Figure (3.6). The cylinder is assumed to be fixed. This means it will not move although it is subjected to some hydrodynamic forces, and its centre coordinates are $(0, 0, 0)$. This case is considered as laminar flow case with Reynolds number equal to 100. By following Equation (3.1), Re is defined by the kinematic viscosity value ν in the `transportProperties` file which is, in this case, $\nu=0.01\text{m}^2/\text{s}$.

The vortex-induced vibration (VIV) simulation of the static cylindrical structure was applied for the laminar flow. The one-degree-of-freedom system or the spring-mass-damper (refer to Figure 3.4) is represented [156]. In this case, there is no structural damping in the cylinder's motion, and it is considered that the damping is appeared due to the fluid viscosity.

3.9.1.2 Mesh Generation

As mentioned previously, in OpenFOAM, the mesh can be generated using many ways. In this case, the Gmsh software is used to produce this case mesh. After generating mesh using Gmsh, then the `gmshToFoam` utility will write the mesh from Gmsh form to OpenFOAM form, and the results will be shown in the `constant/polyMesh` directory [158]. After generating the mesh, it has 92976 nodes and 45940 elements.

In this case, the static mesh is defined by Gmsh software, and its geometry was given in Figure 3.7. The mesh that used in the above defined VIV case is described in the following illustration. As seen in Figure 3.7, the mesh near the wall of the cylinder is built to be fine enough and so that it could be easy to capture the vortex shedding that is important to

generate the cylinder's motion. Additionally, in the other regions near the wall, the mesh is generated to be coarse in order to save the simulation time.

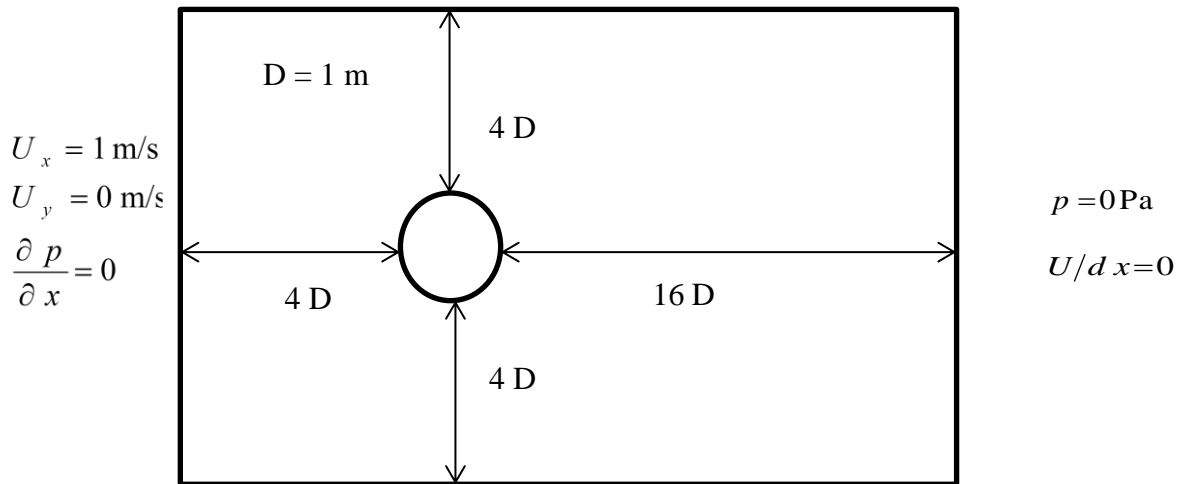


Figure 3.6: Geometry and boundary conditions of flow over a cylinder

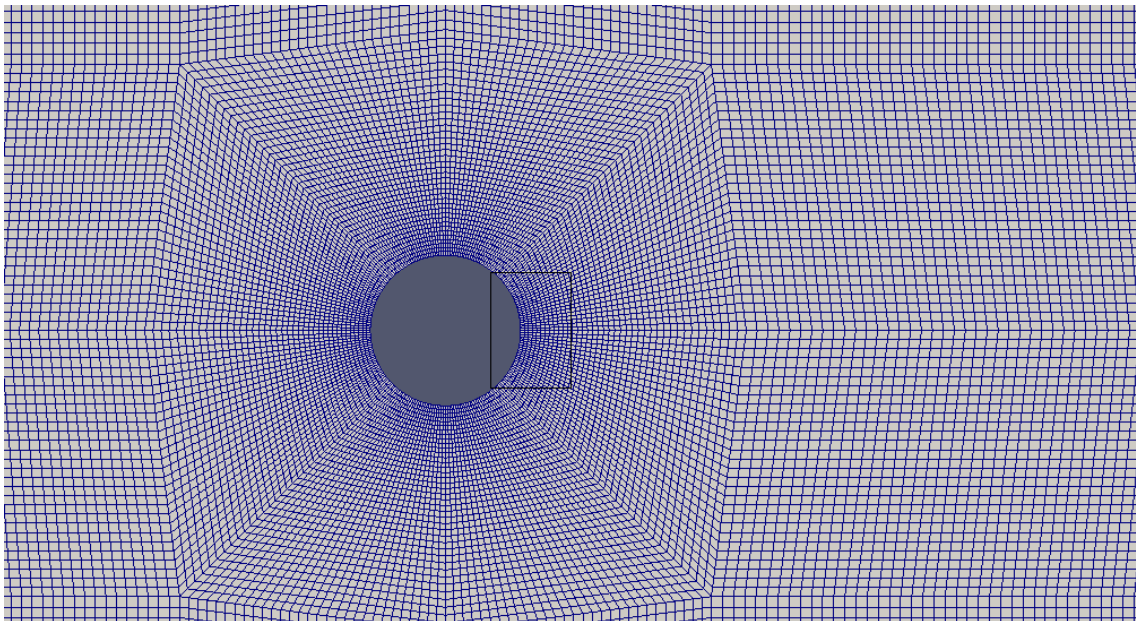


Figure 3.7: Circular cylinder mesh visualised in ParaView

3.9.1.3 Boundary Conditions

The static circular cylinder case is considered as the two-dimensional, incompressible and there is horizontally uniform flow past it. The flow conditions are made by assuming that the flow occurs in a rectangular channel with the circular cylinder which is symmetrically placed between the two plane walls with slip boundary conditions as represented in Figure 3.8.

The inflow velocity boundary conditions for the unconfined cylinder is given as follows:

- At the inlet (inflow), the uniform flow conditions are

$$U_x = U, U_y = 0 \text{ and } U_z = 0. \quad (3.16)$$

- At the outlet (outflow), the flow variables have zero diffusion which means that the boundary conditions at the outlet are extrapolated from the domain to result in the upstream flow conditions. The velocity and pressure are updated by the extrapolation in the outflow. That consists of the assumption of the fully developed flow which is no change in the area at the outlet boundary [163].

$$\frac{\partial p}{\partial t} + \bar{U}_n \frac{\partial p}{\partial n} = 0. \quad (3.17)$$

- At the upper and lower walls, the boundary slip conditions are assumed on the walls where the viscous effects are negligible.
- On the cylindrical structure wall, the no-slip condition is applied where all the velocity components on the cylinder surface will be zero due to the fluid viscous effects.

In OpenFOAM, the velocity and pressure boundary conditions located in the time directory named 0 and the examples of the velocity and pressure will be shown in Appendix 3.A and Appendix 3.B, respectively.

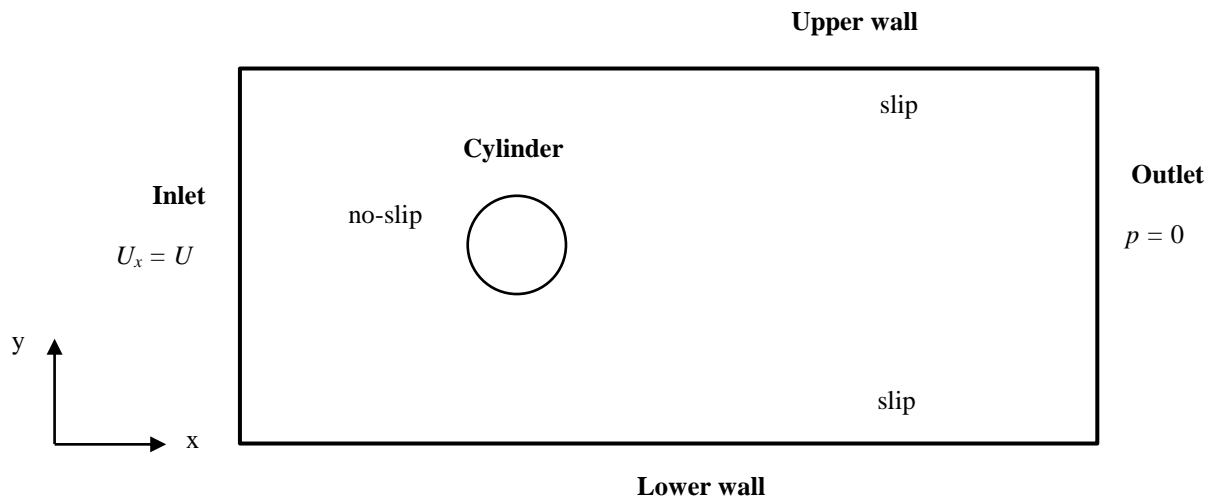


Figure 3.8: Schematic illustration for boundaries in laminar flow

3.9.1.4 Time and Data Input/Output Controls

OpenFOAM solvers are run by setting up the input/output data [158]. The input parameters such as time-step and output time interval are set up in the controlDict file. The time-step must be adapted to reach the low Courant number and provides an accurate solution, particularly for unsteady problems. When the flow velocity is increased, the simulation time-step should be decreased in order to adjust the Courant number (refer to Equation 2.30). In this work, the time-step is customized to the Courant number stays less than 0.8.

In addition, there are some functions can be defined in the controlDict file to print out some more information and results from the simulation. For example, forceCoeffs is one of that additional functions. It is important to extract the hydrodynamic forces (drag and lift) coefficients from the simulation. The controlDict directory for the laminar flow is presented in Appendix 3.C.

3.9.1.5 Discretisation Scheme

Discretisation is very important in order to solve all equation terms. There are many

discretisation scheme options are available in OpenFOAM. The `fvSchemes` dictionary placed in the `system` directory sets all the discretisation terms for the running applications. The `fvSchemes` directory consists of many sub-dictionaries for different discretisation terms as follows:

- `ddtSchemes` defines the terms of the first-time derivative. In this work, the backward time derivative scheme is set which is unsteady and implicit schemes that have second-order accuracy.
- `divSchemes` is considered as an essential discretisation scheme in the CFD and it includes the divergence terms. In this discretisation scheme, the `Gauss` scheme method is always used, and it requires the interpolation scheme selection for the dependent field. Here, the interpolation schemes are assumed to be `linear` and `limitedLinear` for the flux term and for the convection term, respectively. In one hand, both interpolation schemes have second-order accuracy. On the contrary, the `linear` scheme is unbounded while the `limitedLinear` scheme is bounded.
- `gradSchemes` estimates gradients at the cell centres. In this case, the `Gauss` discretisation method is used, and the `linear` interpolation scheme is employed.
- `snGradSchemes` discretises the terms of the surface normal gradient and it defines as the normal component to the cell face. In this work, the scheme is considered as a corrected `grad`.
- `laplacianSchemes` uses to discretise the Laplacian terms, and the interpolation scheme here should be determined by the diffusion coefficients. In this case, the `Gauss` discretisation method is applied for all variables, the `linear` interpolation scheme is also employed, and the surface normal gradient scheme is assumed as `limited` with a 0.5 corrected grade.
- `interpolationSchemes` uses to define the interpolation schemes at the cell faces. Here, the `linear` interpolation method for the velocity is applied which is unbounded and has the second order accuracy.
- `fluxRequired` defines the flux fields that are generated by the application. In this work, fluxes are evaluated from the pressure equation [164].

The fvSchemes dictionary of this case is presented in Appendix 3.D.

3.9.1.6 Solution and Algorithm Control

The equation's algorithms, solvers, and tolerances are all monitored in the fvSolution dictionary located in the system directory. In this case, the solver that is used is GAMG (Geometric-agglomerated Algebraic Multigrid, refer to Appendix 2.A) that requires the positive definite, diagonally matrix. In this solver, smoothing is an essential part of the multigrid method. The high-frequency error on the current mesh can be reduced as it solves for both symmetric and asymmetric matrices. The GaussSeidel smoothing scheme is applied in the solver which means that the smoothing depends on the Gauss-Seidel method.

The pressure-velocity coupling algorithm in the laminar cases is PIMPLE algorithm. The fvSolution directory for this example is shown in Appendix 3.E.

3.9.1.7 Laminar Flow Solver

Here, the pimpleFoam solver is chosen for the laminar flow simulation for the Newtonian fluid which is a transient solver for the incompressible flow with a large time-step using PIMPLE algorithm. The code is naturally transient and, initial and boundary conditions are requiring [158,165-166].

3.9.1.8 Post-processing

The simulation output variables will be the velocity and pressure time directories and the force coefficients. The drag and lift coefficients; the hydrodynamic force coefficients, can be extracted from the controlDict dictionary by adding that under the keyword forceCoeffs with some basic information such as cylinder diameter, free stream velocity, reference area, and others that should be mentioned in order to get the correct force coefficients (refer to Appendix 3.C). In addition to that, the simulation results visualisation such as

pressure, velocity, viscosity, etc. can be shown in ParaView by using the paraFoam utility. Figure 3.9 presents the velocity simulation result of the static circular cylinder using ParaView. The color difference describes the different velocities, blue presents the smallest velocity value and the darker colors in the red range represents the increased velocity values.

3.9.1.9 Laminar Flow with Re = 100 Simulation Results and Discussion

The velocity and pressure visualisation profiles for the laminar flow with $Re = 100$ are presented in Figures 3.9 and 3.10, respectively.

From both Figures 3.9 and 3.10, it can be seen that the vortex shedding has appeared and these results combine with the vortex shedding theories in Section 3.3 which stated that the vortex street started to occur for flow with $Re > 40$. In this range, the wake will become unstable, and ultimately it will result in the vortex shedding phenomena due to its oscillation at a certain frequency.

From Figure 3.12, a Strouhal number obtained is 0.167, which is very close to the Strouhal number obtained by other studies [22,167-169]. The lift coefficient in this simulation is 0.374, and the mean drag coefficient is 1.540.

The first part of the simulation time (<50s) shows the transient phase through the generated perturbations arrives at the cylinder and then causes the shedding. The drag coefficient oscillation is used to characterise the periodic state at twice of the lift coefficient.

3.9.1.10 Results Summary and Discussions of Laminar Flow with Re = 100, 200, and 1000

Some studies result for the mean drag and lift coefficients of $Re = 100, 200,$ and 1000 are summarised in Table 3.2. This table shows the comparison of hydrodynamic forces coefficients gained from the past literature with the present study. It can be noticed that the results are shown a big agreement with the previous literature at $Re = 100$ and 200 .

However, at $Re = 1000$, there is a remarkable difference between the force coefficients found by Frank *et al.* [172] and the current study. The significant difference between the value of lift coefficient at $Re = 1000$ and other two at $Re = 100$ and 200 is found. This may be due to transitional turbulence. Moreover, the drag coefficient at all studied cases is systematically higher than one in the literature this is might be due to the differences between the coarse and fine meshes.

In addition, Figures 3.13 and 3.14 presents the comparison results that are summarised in Table 3.2. It can be seen that the amplitude of lift coefficient increases exponentially as the Reynolds number increases.

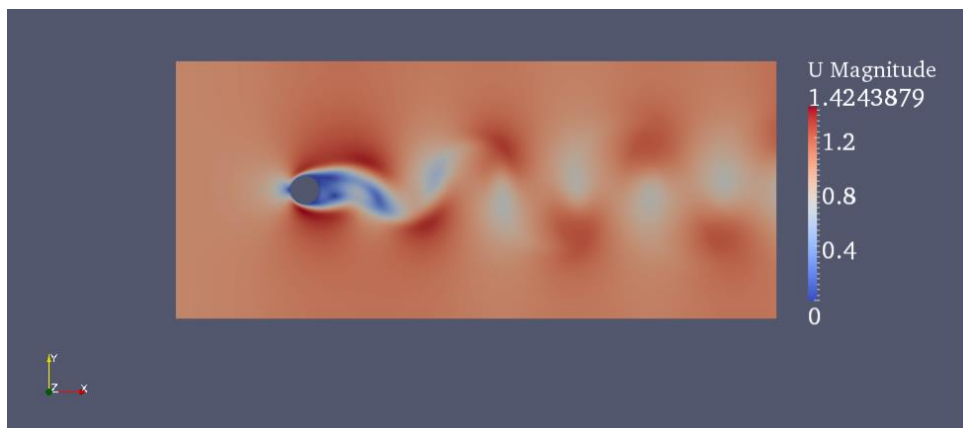


Figure 3.9: Velocity profile for laminar flow with $Re = 100$ in ParaView

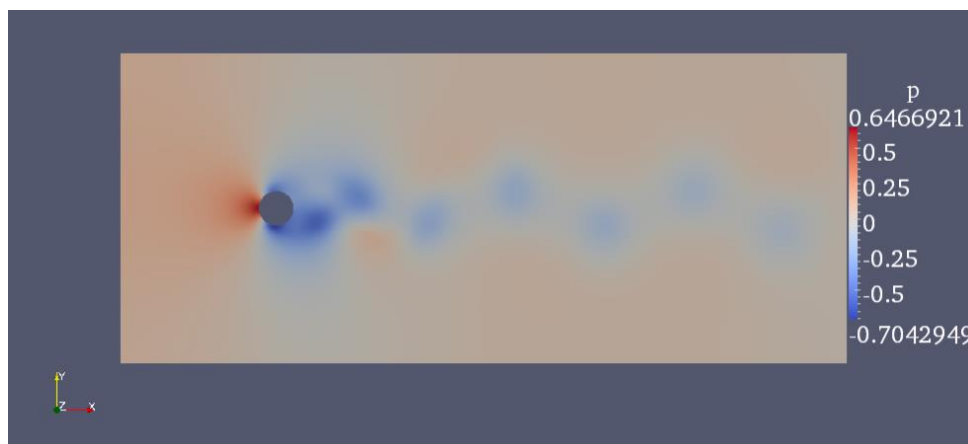


Figure 3.10: Pressure profile for laminar flow with $Re = 100$ in ParaView

Table 3.2: Comparison of force coefficients from the simulation results with other studies

Source	Re = 100		Re = 200		Re = 1000	
	$\overline{C_D}$	C_L	$\overline{C_D}$	C_L	$\overline{C_D}$	C_L
Mittal and Raghuvanshi [162]	1.402	0.355	-	-	-	-
Berthelsen and Faltinsen [170]	1.38	0.34	1.37	0.7	-	-
Calhoun [171]	1.33	0.298	1.17	0.668	-	-
Franke, <i>et al.</i> [172]	-	-	1.31	0.65	1.47	1.36
Herfjord [173]	1.36	0.34	1.35	0.70	-	-
Linnick and Fasel [174]	1.34	0.333	1.34	0.69	-	-
Rajani, <i>et al.</i> [175]	1.335	0.179	1.337	0.424	-	-
Russel and Wang [176]	1.38	0.30	1.29	0.5	-	-
Xu and Wang [177]	1.42	0.34	1.42	0.66	-	-
Tezduyar, <i>et al.</i> [178]	1.37	0.371	-	-	-	-
Present work	1.540	0.374	1.560	0.727	1.693	1.123

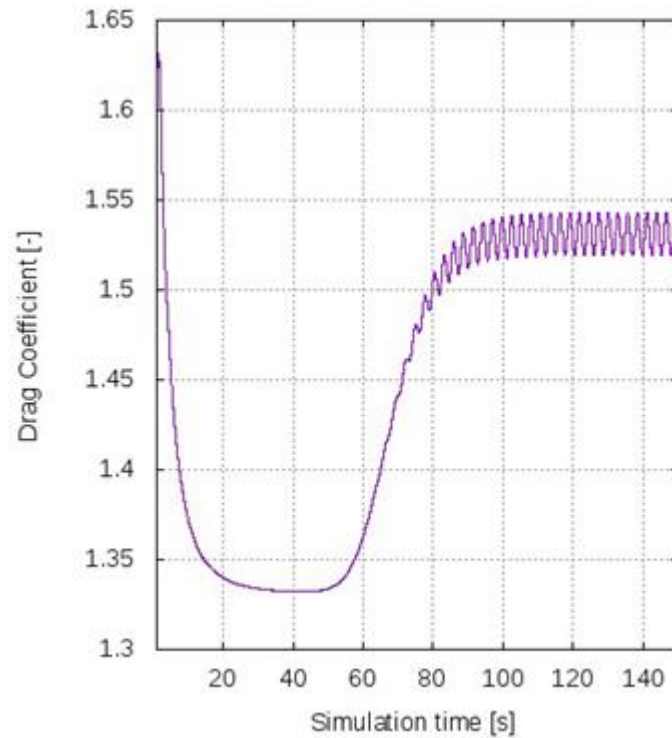


Figure 3.11: Drag coefficient of 2D cylinder at laminar flow of Re = 100

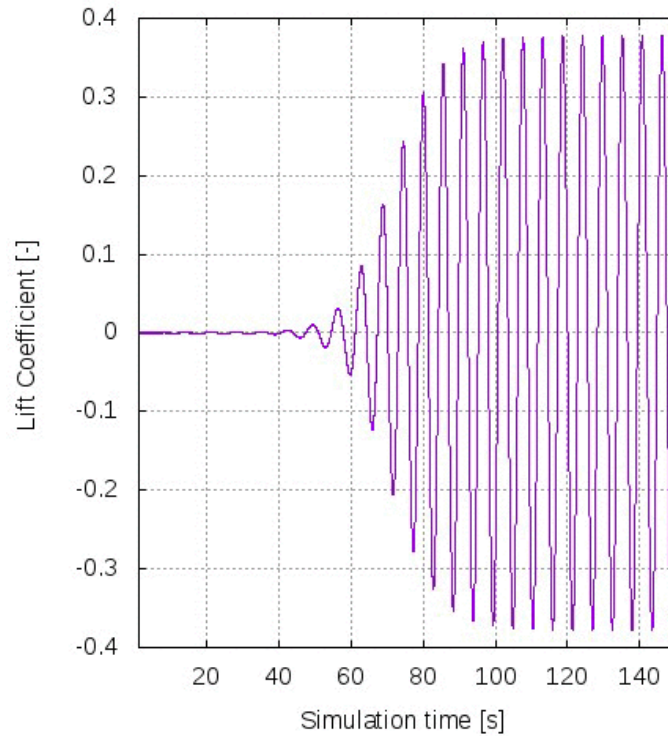


Figure 3.12: Lift coefficient of 2D cylinder at laminar flow of $Re = 100$

3.9.2 Vortex-Induced Vibration of the Circular Cylinder

The vortex-induced vibration simulation over the cylindrical structure cases will be discussed for the laminar flow with $Re = 100$. The system of spring-mass-damper will be presented to show the one- and two-degrees-of-freedom motion as illustrated in Figure 3.4. However, the structural damping is generated from the fluid viscosity since there is no cylinder motion is considered to produce the damping.

3.9.2.1 Mesh Generation

Since the vortex shedding allows the cylinder to move, the static and dynamic meshes are required. The dynamic mesh can move along with the cylinder's motion. Thus, there is a `dynamicMeshDict` file located under the `constant` directory to define the mesh motion as described in Figure 3.15.

As stated before, the dynamicMeshDict file is used to the dynamic mesh generation, and its content will be shown in Appendix 3.F, and its content explanation will be presented in the table on the same page. Whereas, the static mesh is generated by Gmsh software as mentioned previously in Section 3.9.1.

3.9.2.2 Boundary Conditions

Both the initial and the boundary conditions for the pressure, velocity, and the position of the cylinder should be defined. The pressure and velocity initial conditions were discussed previously in sub-section 3.9.1.3, while the cylinder's position initial condition will be defined in the pointDisplacement file (refer to Appendices 3.G, 3.H, and 3.I).

3.9.2.3 Time and Data Input/Output Controls

In order to get the accurate results, the time-steps should be adjusted by giving the Courant number value is below 0.2. In this vortex-induced vibration case, the time-step is set to be 0.001 for 100 s simulation time. This will lead to get low Courant number and eventually the results will be accurate (refer to the controlDict file Appendix 3.C).

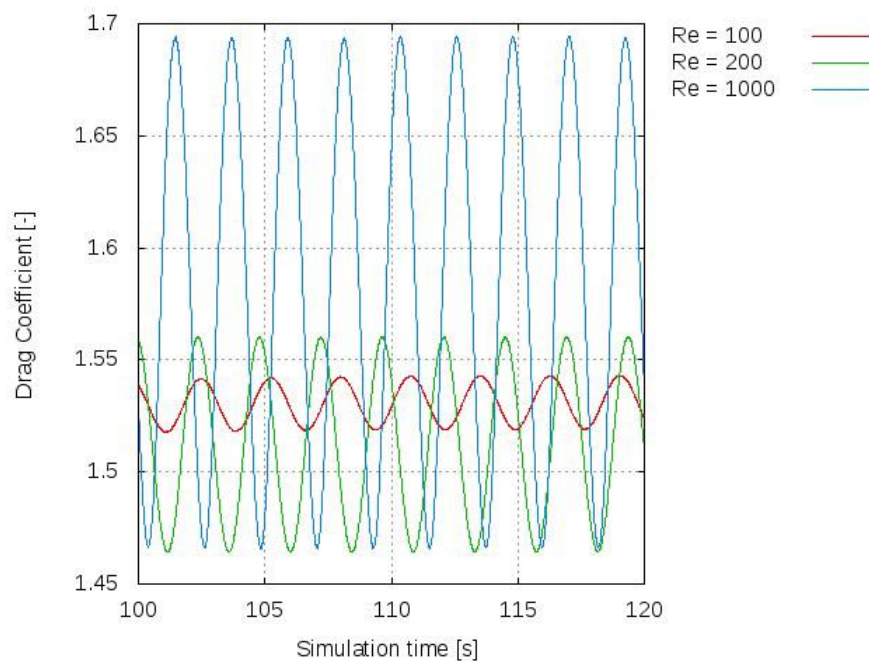


Figure 3.13: Drag coefficients of laminar flow at Re = 100, Re = 200, and Re = 1000

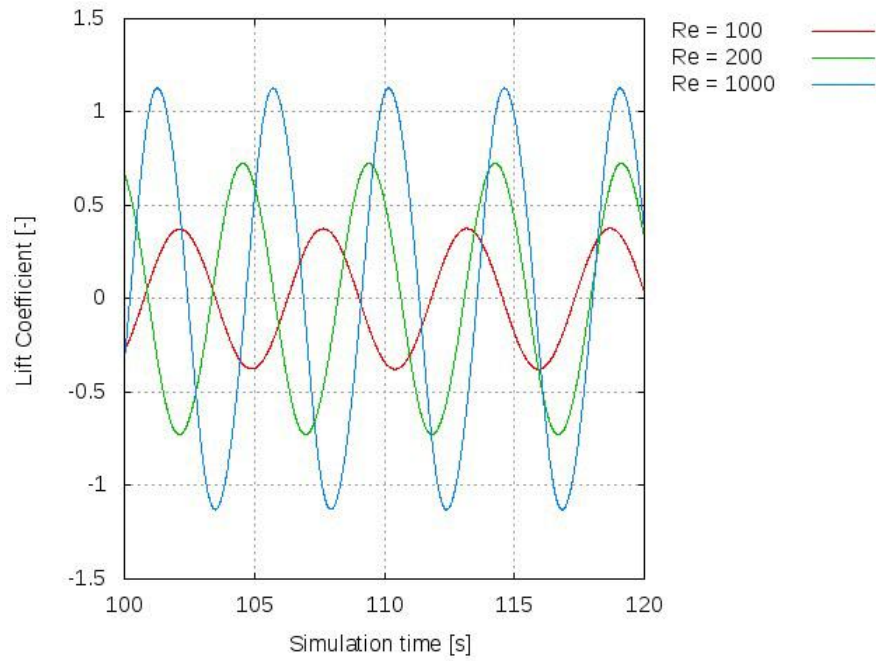


Figure 3.14: Lift coefficients of laminar flow at $Re = 100$, $Re = 200$, and $Re = 1000$

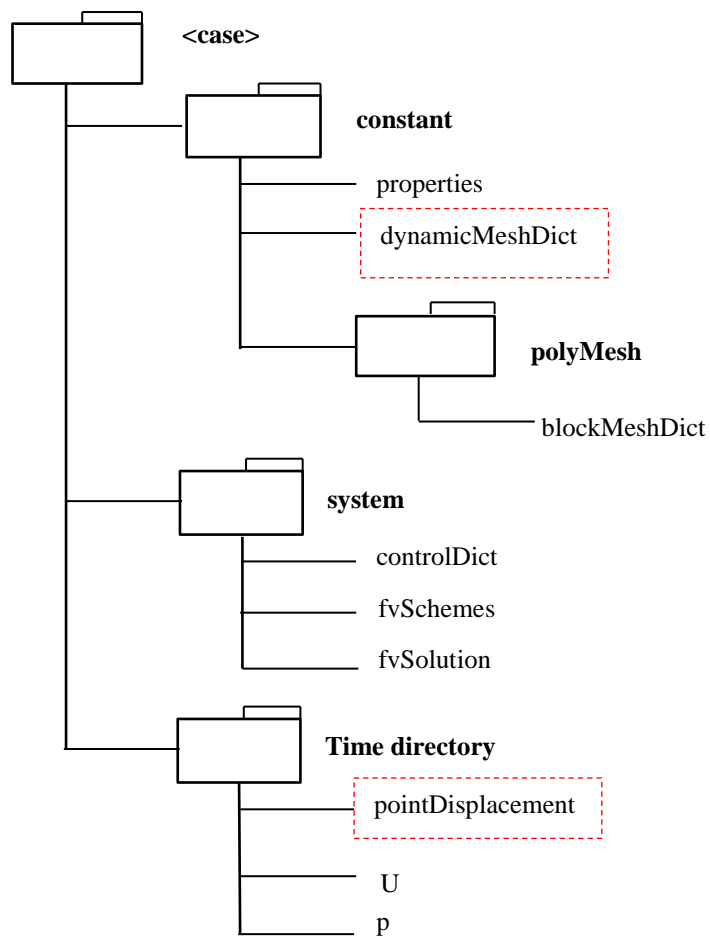


Figure 3.15: Directories and files for vortex-induced vibration case

3.9.2.4 Discretisation Scheme

As mentioned previously in Section 3.9.1, the `fvSchemes` file is used to describe the discretisation procedure of the case simulation and the sub dictionaries for the VIV simulation.

- `ddtSchemes` defines Euler time derivative scheme which is bounded and first order accuracy implicit.
- `divSchemes` uses the Gauss discretisation scheme, and the interpolation scheme for the velocity, flux and convective terms is set to be `linearUpWind`, `linear` and `limitedLinear`, respectively.
- `gradSchemes` estimates the Gauss discretisation method and the linear interpolation scheme are employed.
- `snGradSchemes` discretises the uncorrected scheme which leads not to apply any of the non-orthogonal corrections.
- `laplacianSchemes` uses to discretise the Laplacian terms, the Gauss discretisation method is applied for all variables, the linear interpolation scheme is also employed, and the surface normal gradient scheme is assumed as `limited` with a 0.5 corrected grade.
- `interpolationSchemes` uses to define the interpolation method. Here, the linear interpolation method for the velocity is applied which is unbounded and has the second order accuracy.
- `fluxRequired` defines the flux fields that are resulted from the pressure. Thus, the fluxes are calculated from the pressure equation [164].

3.9.2.5 Solution and Algorithm Control

The GaussSeidel algorithm is employed in the smoothing scheme solver. The PBiCG is used for the velocity and the GAMG for both the point displacement and the pressure.

Additionally, the PIMPLE algorithm for the pressure-velocity coupling is used. By referring to Appendix 3.E, the fvSolution directory for the simulation of the vortex induced vibration is presented.

3.9.2.6 VIV Solver

The pimpleDyMFoam solver is implemented for the VIV simulations. This solver is using the PIMPLE algorithm as the pimpleFoam solver. However, the differences between the pimpleDyMFoam solver and the pimpleFoam solver are that the earlier uses the dynamically moving mesh. In addition, they both have the same function and are using the transient incompressible flow solver.

In the following section, the non-resonance VIV case model will be discussed.

3.9.3 Non-Resonance VIV Schematic Description

To implement the non-resonance VIV case, Figure 3.16 illustrates the basic idea of how the circular cylinder will be fixed in such particular locations and allow to float around x- and y-axes while applying some constraints. The case's description is given as follows:

- There are four springs (S1, S2, S3, and S4) will be applied, and each of them has its own damping and stiffness coefficients and initially, for simplicity, can give all springs the same stiffness and the same damping values.
- All four springs are sharing the cylinder centre (O) and each of them has another defined points at A1, A2, A3, and A4.
- For each spring, the rest length will be applied when there is no fluid flow and gravity. Thus, the rest length will be the distance from that point to the cylinder centre, namely A1-O, A2-O, A3-O, and A4-O.
- Therefore, there will be two forces will be acting over each axis; namely two forces in the x-axis and another two forces in the y-axis. This will lead to such problems because the forces vectors will be the same all over the time due to the forces that are acting from the fluid flow around the cylinder.

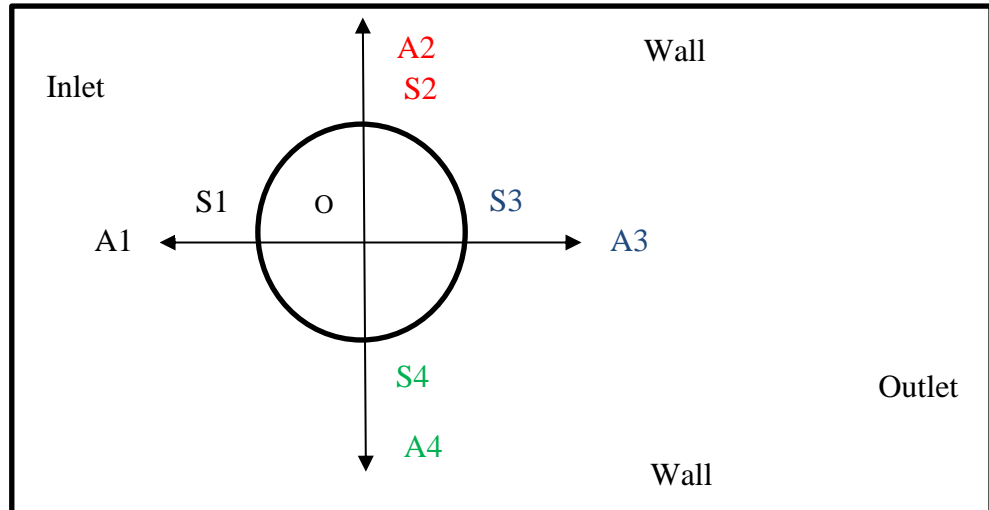


Figure 3.16: Schematic for the VIV non-resonance case

3.9.3.1 Computational Domains Schematics

According to the previous schematic that described the basic concepts of cylinder motion for the free oscillations cylinder case, there are two different computational domains will be discussed for one- and two-degrees-of-freedom as given in the next sub-section.

3.9.3.2 Scenarios of the VIV with Four-Springs Case, and Two-Springs and Two-Dampers Case

In the first schematic, both of 1DOF and 2DOF will be applied to study the vortex-induced vibration. In the 1DOF, the cylinder is allowed to move in the transverse direction (y-direction) only. However, for the second case, the cylinder is considered as free to oscillate in both x- and y-axes (inline and transverse directions).

In this schematic case, there are three different scenarios will be discussed

1. Scenario 1: has applied for four springs (S1, S2, S3, and S4) (refer to Figure 3.16) with damping effects. Thus, this means that there will be four forces are applied in all springs which are produced by the spring effects and the damper will slow down on all springs. Moreover, it is essential to note that all springs have the same size

and there is no rotation for the cylinder to rotate on its axis. The fluid flow is going over the cylinder with nearly symmetrical forces on the top and bottom sides. Technically this scenario is for the 2DOF system and will be presented in the pointDisplacement file in Appendix 3.G.

- Scenario 2: will be applied more constraints to the previous scenario, the result is that the cylinder is forced to move only along the y-direction. This case is 1DOF as will be presented in Appendix 3.H.

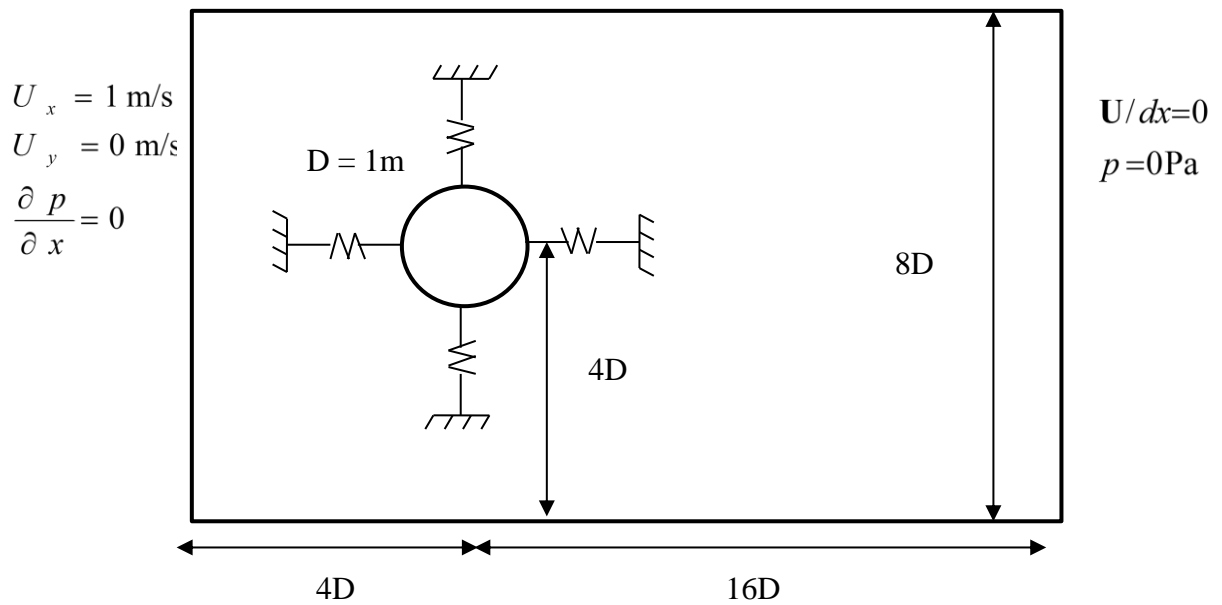


Figure 3.17: Geometry and flow conditions of case with 4-springs

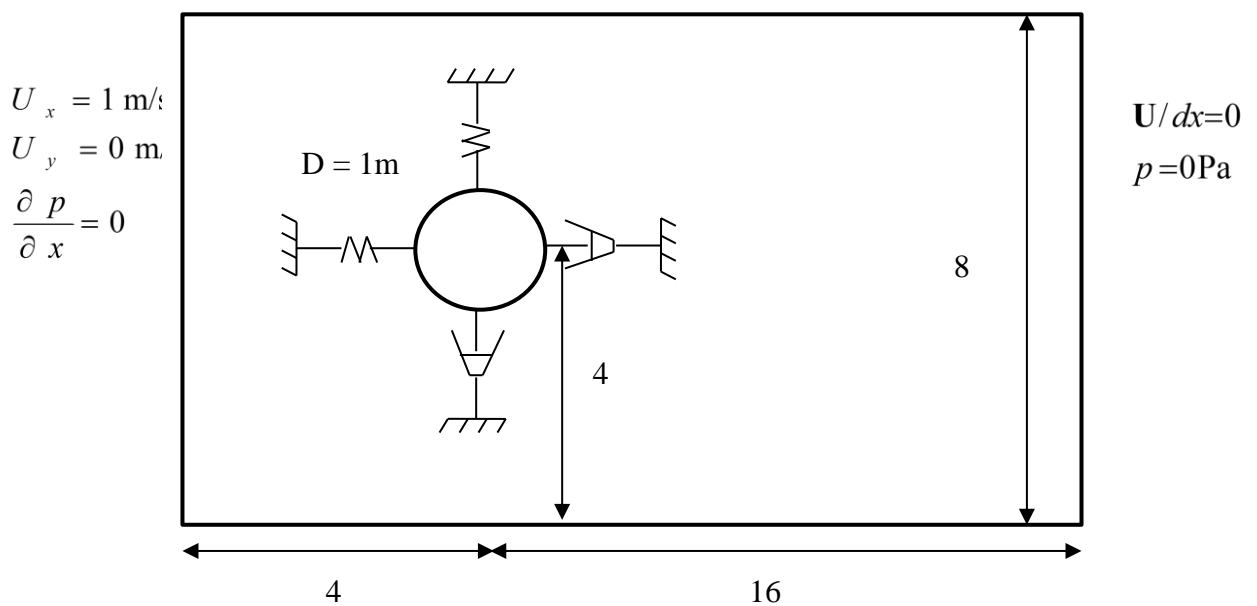


Figure 3.18: Schematic of 2-springs and 2-dampers

3. Scenario 3: has applied for two springs and two dampers as shown in Figure 3.18. This is also for the 2DOF system since the cylinder move in both in-line and transverse directions as described in Appendix 3.I.

For all three scenarios, the cylinder mass value is assumed to be 9.97 kg, damping and stiffness are 2 N.s/m and 4 N/m, respectively.

3.9.3.3 Results and Discussion of the VIV Simulation of 1DOF and 2DOF

The flow regime results around the free oscillating circular cylinder for the previously discussed scenarios will be presented as follows:

- **Drag and lift coefficients for 1DOF and 2DOF systems**

From Figure 3.19, it can be observed that drag coefficient behaviour for the one-degree-of-freedom system is very close to the static cylinder, while for the two-degrees-of-freedom it is slightly different particularly in the simulation time between 10s and 20s. Additionally, the drag coefficients for 1DOF is higher than the one for 2DOF, while the lift coefficients for both 1DOF and 2DOF are nearly the same.

However, there is a slight delay in the response time, where the 2DOF system takes about 0.2 seconds longer to reach the similar amplitude peaks, for both the drag and lift forces. This is related to how the 1DOF case can only move along the vertical direction, which consequently forces it to react directly to any vortices that are created on the wake side. On the other hand, on the 2DOF case, the cylinder can be dragged a bit longer towards the right (in the direction of the wake), which results in the delay in responding to the forces exerted by the vortices. In other words, the delay is directly associated with the time it takes for the springs to restrain the cylinder on the rightmost location where it starts to oscillate periodically.

Regarding the amplitude of the drag coefficient, when compared to the 1DOF, the 2DOF case has a smaller amplitude which should be associated with the additional degrees of freedom.

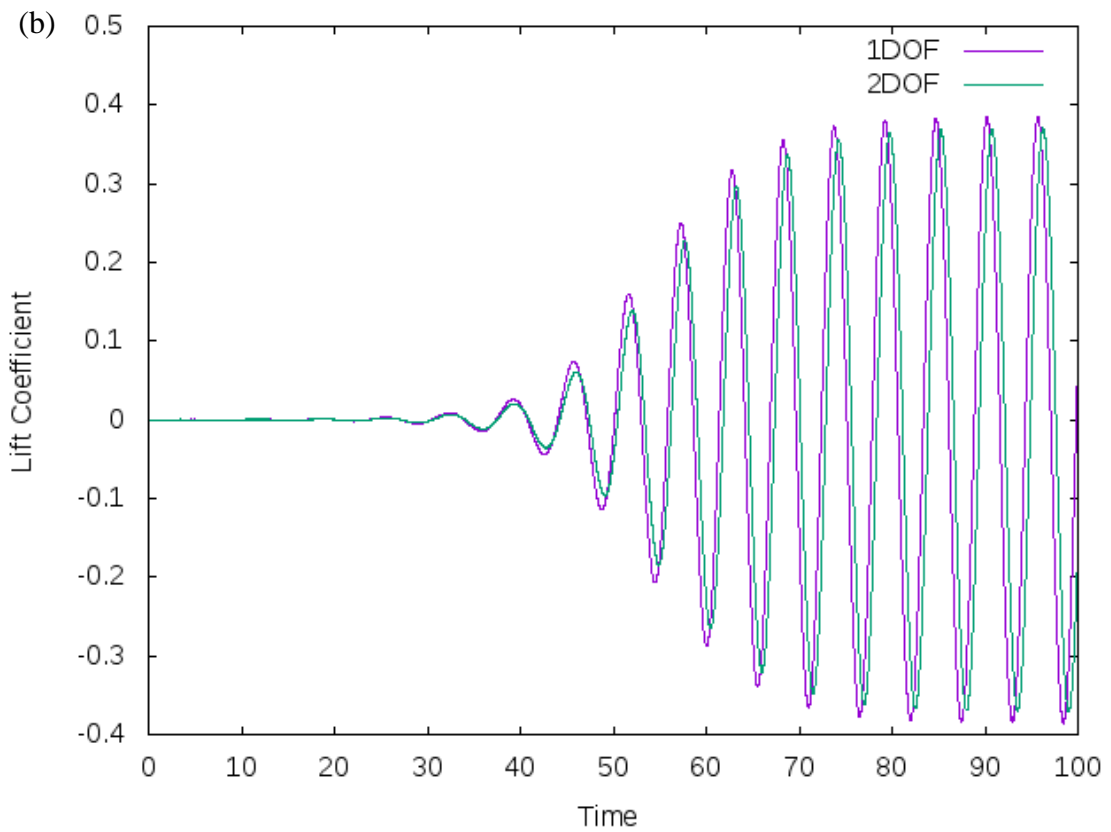
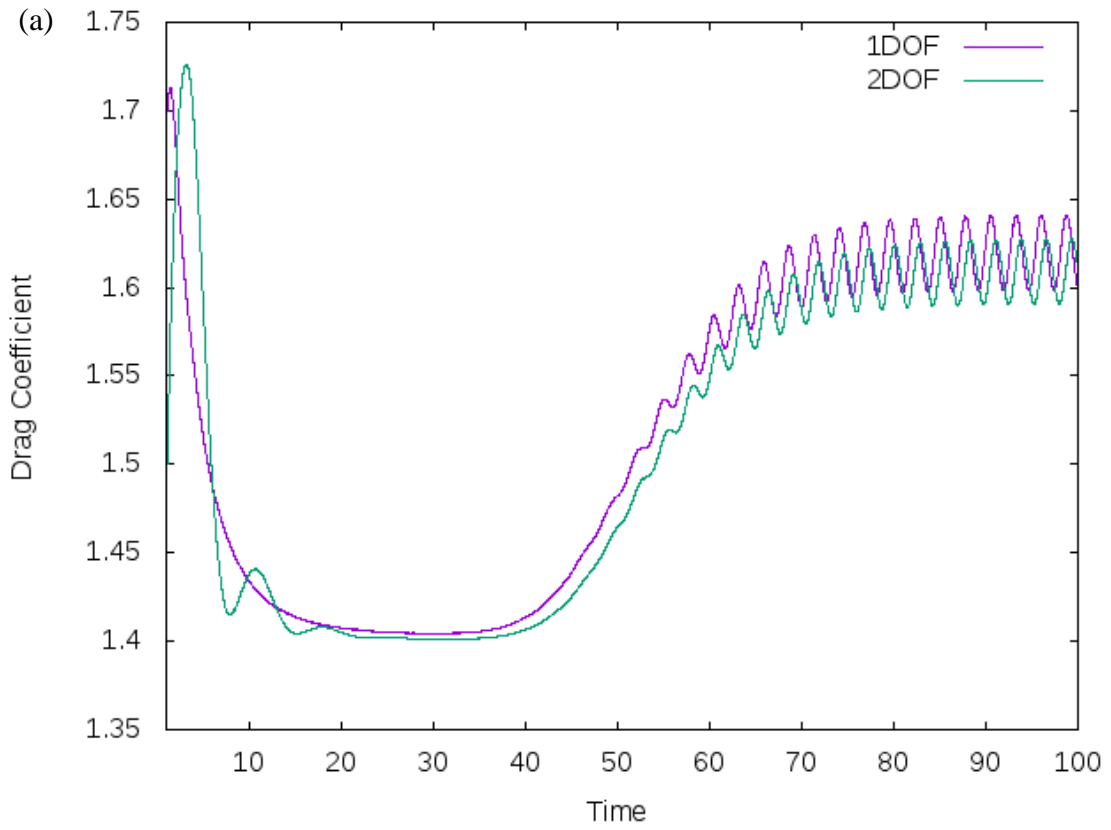


Figure 3.19: (a) Drag and (b) lift coefficients for 1DOF and 2DOF

Similarly, the lift coefficient has a slightly smaller amplitude on the 2DOF case, because of how the cylinder is being dragged further along with the flow and is moving closer to the paths of the vortices.

- **Drag and lift coefficients results for 1DOF system**

Figure 3.20 shows the drag and lift comparing results for the 1DOF system (scenario 2) and its results when reduced damping and stiffness coefficients to 1. The results show that the drag and lift amplitudes for the original case are higher than the ones when reduced damping or spring stiffness.

In order to understand what is happening in the two variants of this case, it is necessary first to revise how the spring and the damper work:

- The spring acts in direct proportion to the applied forces: $\text{Force} = \text{Stiffness} \times \text{Displacement}$. This means that the force induced by the spring is linearly proportional to the stiffness value, in function of the displacement of the spring.
- The damper acts in direct proportion to the velocity of the displacement: $\text{Force} = \text{Damping} \times \text{Velocity}$. Which means that the faster the displacement occurs, the more force the damper exerts.

Since these forces refer to the ability of the springs to react to the flow around them, these relate directly to the drag forces. In other words, the forces exerted by springs will affect the movement of the cylinder, as a direct response to the forces imposed by the flow; given that the drag coefficient is calculated based on the forces applied to the cylinder, therefore, this corresponds to the forces sustained by the springs

With these definitions in mind, it is possible to get a clearer understanding of what is happening when the stiffness and damping are reduced:

1. All of these three cases are under-damped, namely, the damper is not strong enough to stop the oscillation, which is why all three tend to have a uniform oscillation pattern with a maximum amplitude.

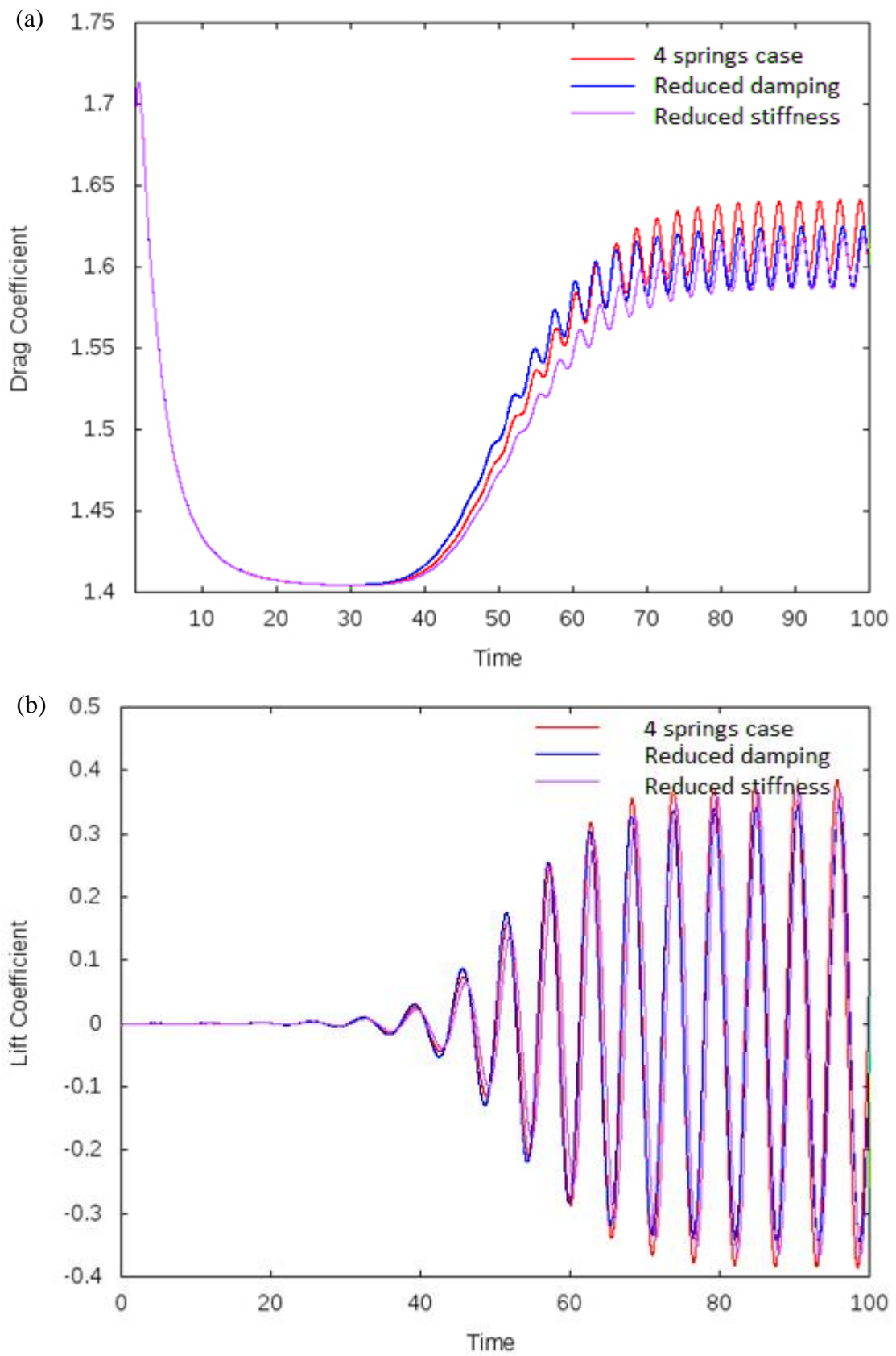


Figure 3.20: (a) Drag and (b) lift coefficients for 1DOF system

2. All of these three cases have nearly the same drag coefficient for the initial 30s, given that the motion trajectories are nearly the same for this initial period. The reduction in the drag coefficient should be directly related to how the first vortex is being generated and then released, with another being generated as the second vortex is being generated.
3. The case with reduced damping has a reduced ability to respond with force in function of velocity, resulting in:
 - The higher drag coefficient from the 30s to around 63s, is due to the cylinder being able to move faster for nearly the same forces exerted on the cylinder, which results in larger vortices and displaced fluid as the cylinder moves; as an analogy, it is as if the cylinder can more easily get in the way of the fluid going around it.
 - The lower drag coefficient after ~65s is related to the faster ability for the cylinder to bob and weave as the vortices are generated in its wake.
4. The case with reduced stiffness consistently has a smaller drag coefficient after the initial 30s period, given that it can be moved farther away for nearly the same forces imposed by the flow around the cylinder. Given that it can have a longer displacement on all springs (due to the smaller stiffness coefficient), it is then able to accommodate better the periodic generation and release of the vortices.
5. As for the lift, the results are consistent with the ones observed when comparing the 1DOF case with the 2DOF case with 4 springs, given that the added ability for the cylinder to move, will lead to a smaller amplitude in the lift, given that the vortices can be generated and released with smaller dragging forces.
6. As a final comparison, it can be noted that the reduction in stiffness resulted in the smaller amplitudes for drag and lift after the initial 30s period.

- **Drag and lift coefficients results for 2DOF system**

In comparison to the results given for 1DOF, the drag coefficients for the 2DOF system (scenario 1) show different behaviour for the initial 30s period. As already observed with

the changes in damping and stiffness with the 1DOF case, during this initial period it can be observed that:

1. The reduced damping relates to a faster movement (nearly the same force, smaller damping coefficient, results in a higher velocity: $\text{Force} = \text{damping} \times \text{velocity}$), which consequently results in a faster response time for the cylinder to move with the vortices at a higher rate than the other two variants, hence the peak amplitudes for the first 30s being higher/lower than the reference 2DOF case.
2. The reduced stiffness results in the ability for the cylinder to move farther and consequently generate smaller (less intense) vortices than the two other variants, since it is able to be displaced by the fluid flow around the cylinder.

After this initial 30s period, the results are analogous to what was observed in the same variants of the 1DOF case:

1. The reduced damping allows for a smaller drag coefficient than the reference case;
2. The reduced stiffness allows for the smallest drag coefficient;
3. Both variants have nearly the same frequency as the reference 2DOF case.

As for the lift, results are approximately the same, especially for reduced stiffness and damping cases. Generally, the same as for 1DOF system the 2DOF shows higher amplitudes for the original case than that for reduced spring stiffness or damping because of the same reality reason which describes the forces effect leads to disappear the oscillation after a while.

- **Drag and lift coefficients results reduced damping for 1DOF and 2DOF systems**

The drag coefficient in the reduced damping variants shows a significant difference between 1DOF and 2DOF systems in the early simulation time. The drag coefficient behaviour for the 1DOF is nearly identical to the reference 1DOF case, whereas the 2DOF variant shows a more oscillatory pattern until about 35s of the simulation time. This is due to the reduced damping, which allows for a faster reaction speed of the springs, as well as letting the majority of the oscillatory effect be brought in by the stiffness of the

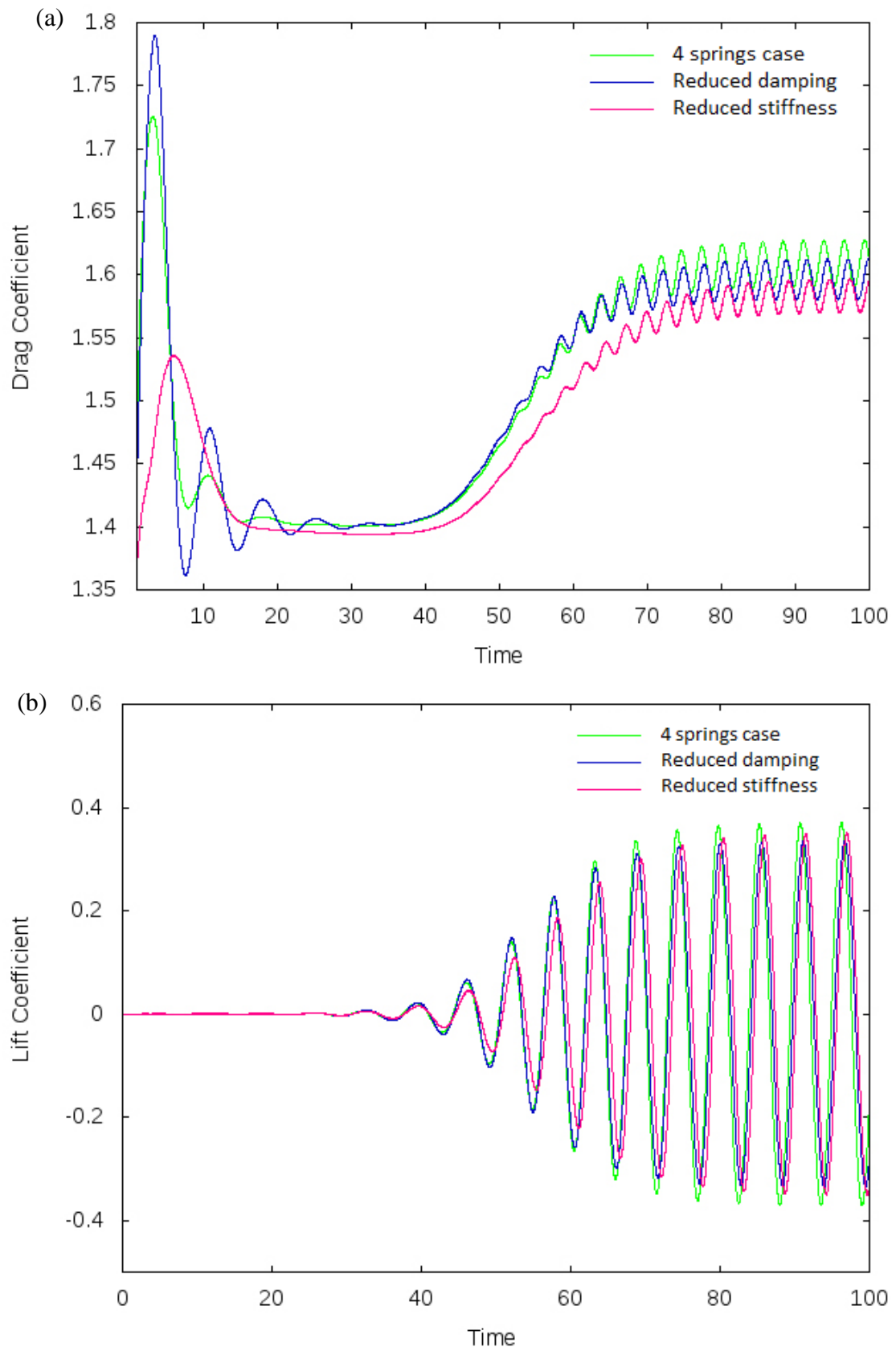


Figure 3.21: (a) Drag and (b) lift coefficients for 2DOF system

springs.

- When the oscillation becomes uniformly periodic for both cases, the result of 1DOF shows a higher drag coefficient, comparable to the respective reference cases.
- As for the lift, the results are approximately the same, as already expected from the results observed.

- **Drag and lift coefficients when reduced spring stiffness for 1DOF and 2DOF systems**

1DOF drag results show the highest values for all simulation time, and the lift results show the slightly higher results for 1DOF even it is not that much big difference.

For the most part, there is not much more to comment on these results that have not yet been addressed in the previous sections, except for with one particular detail: the 2DOF variant with reduced stiffness has the lowest drag coefficient in comparison to all other cases and variants, throughout the whole profile over time, as well as the smallest amplitude range during the uniform periodic time region. From a perspective of low energy and interference on the fluid flow, this would be the best configuration to be used, if further optimisation could not be done. Especially, this is because the lift performance is similar to all other cases and variants.

- **Drag and lift coefficients for 2DOF system in schematics of 4 springs, and 2 springs and 2 dampers**

Figure 3.24 illustrates the comparing results of 2DOF systems (scenario1 that shows the case of four springs attached to the cylinder and scenario3 which presents the cylinder is hanging with 2 springs and 2 dampers).

So before observing the results, keep in mind that the scenario3 effectively represents the 2 dampers are not always aligned with the 2 springs, which will result in not having a fully balanced system as the scenario1.

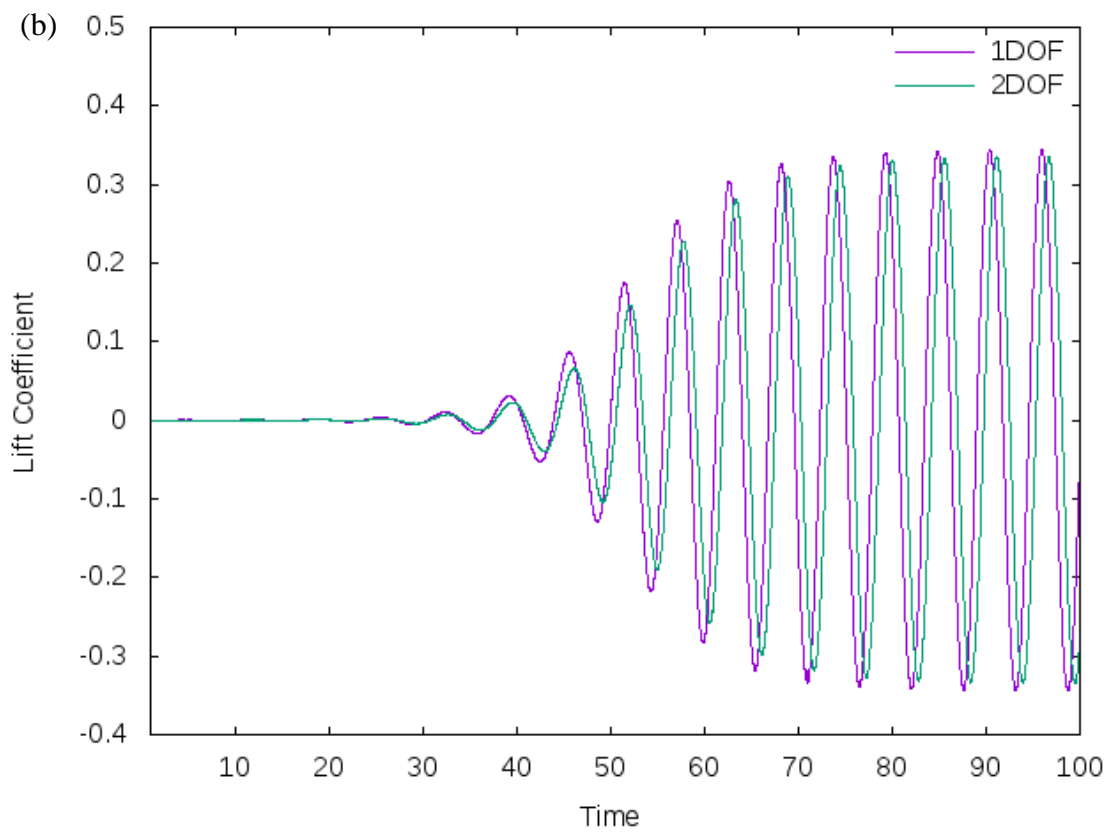
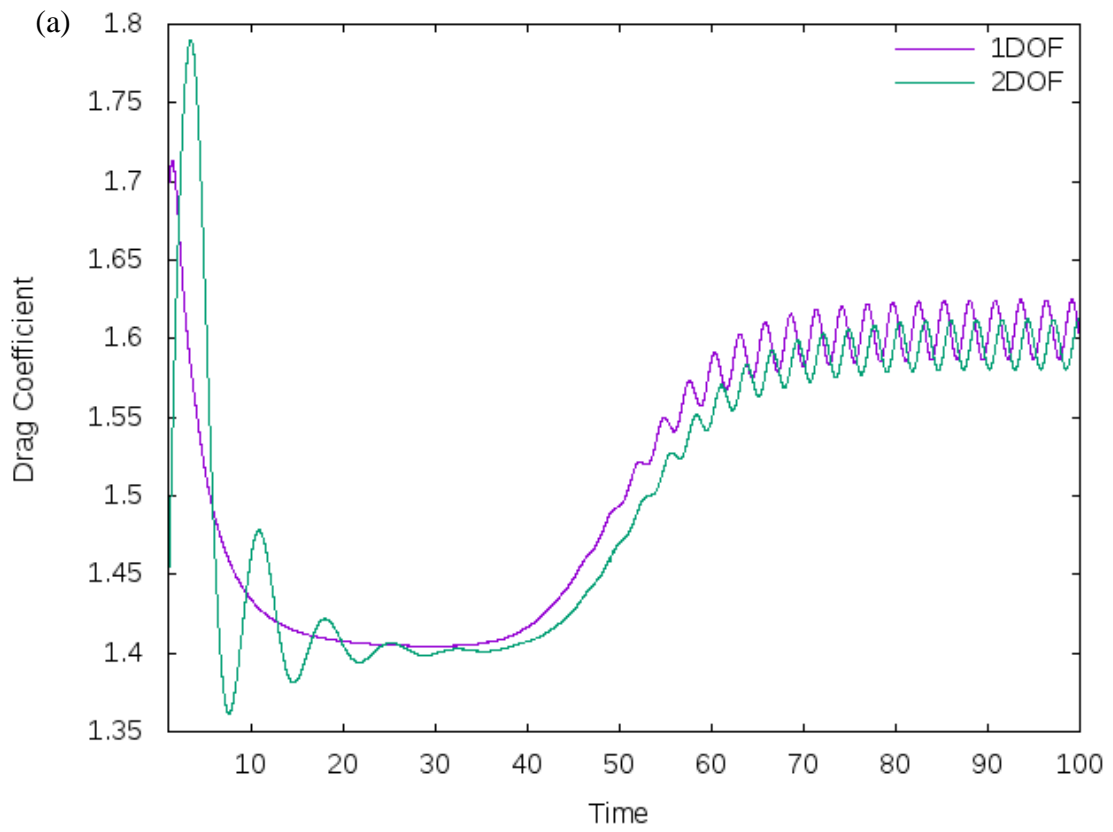


Figure 3.22: (a) Drag and (b) lift coefficients for reduced damping case for 1DOF and 2DOF

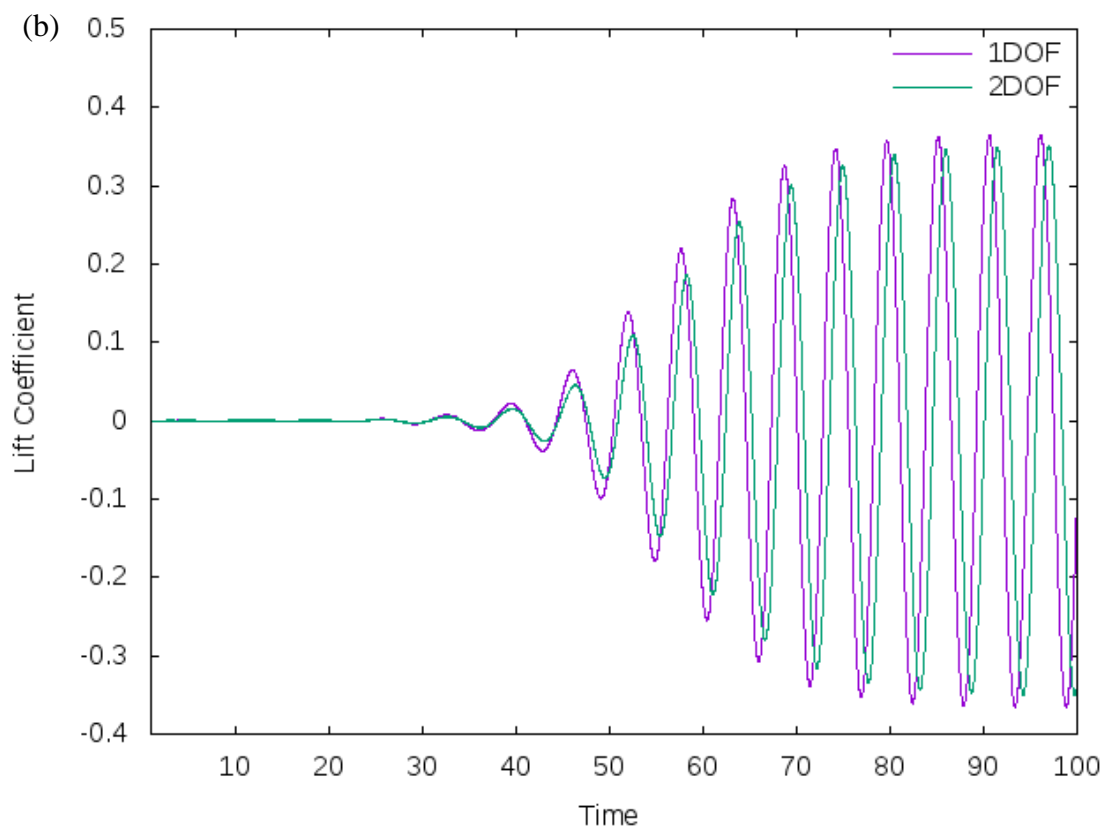
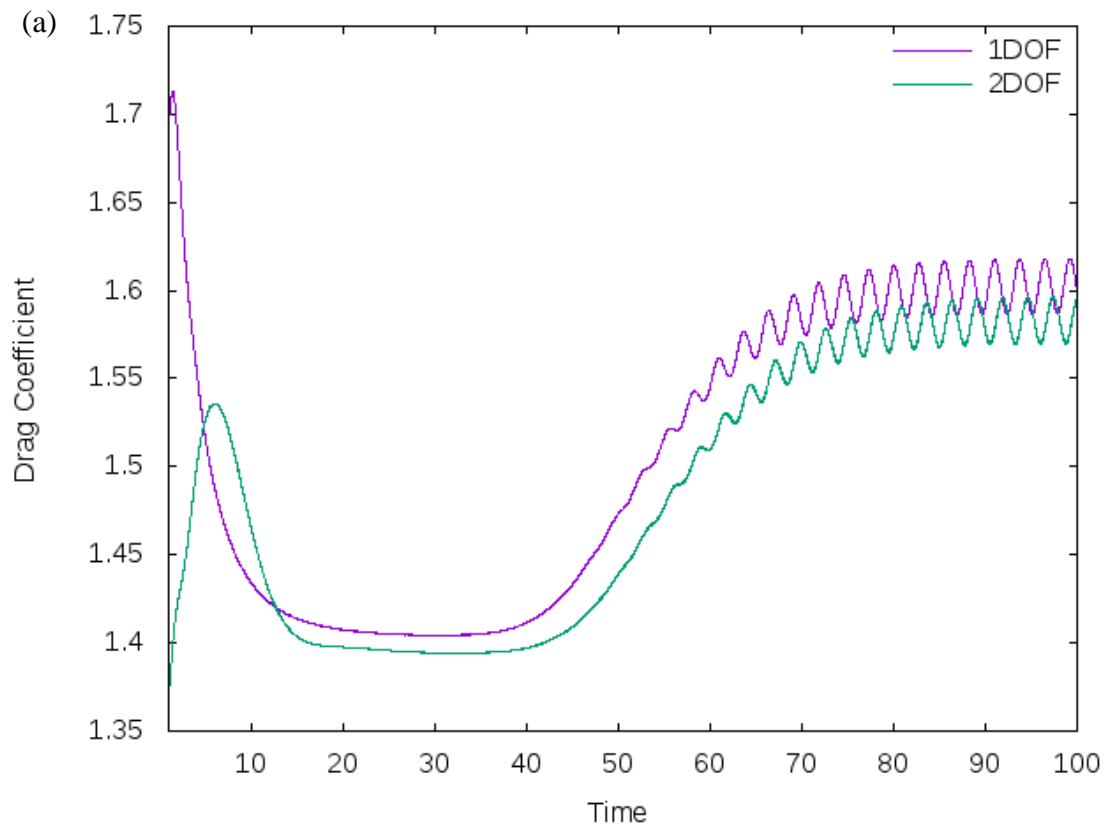


Figure 3.23: (a) Drag and (b) lift coefficients for reduced spring stiffness case for 1DOF and 2DOF

The drag results show a significantly different behaviour between both cases. For the case with four springs, the drag coefficient decreases dramatically from about 1.73 at ~3s to 1.4 at around the 40s of the simulation time and then increase gradually till about 70s of the simulation time to start the same fluctuating behaviour. Whereas, the 2DOF case with two springs and two dampers takes the same fluctuating behaviour from the 30s of the simulation time, instead of 70s.

On the other hand, the lift coefficient for the scenario1 has 0 lift till about 40s, while for the other case model the lift results show fluctuation from nearly the start till the end of the simulation time this is because of the dampers. Moreover, by the end of the simulation time, both cases lift coefficients are nearly the same.

When revising the drag curves for both scenarios, the following can be assessed:

1. The asymmetry in damper/spring configuration is visible in the respective plot, starting from the 25-30s mark; this is precisely because the dampers are not always aligned with the springs, resulting in periods where the cylinder can move faster or longer, depending on the location of the cylinder.
2. The maximum amplitude of the drag coefficient is smaller in the scenario3, as expected when comparing to the results for the other scenarios and variants, given that the stiffness for S3 and S4 has 0 value, and damping is 0 at S1 and S2.
3. The reduced damping can lead to a faster response time, which is effectively observed here, given that it takes roughly 40s less to reach the periodic working region.

3.9.4 Forced Oscillation Cylindrical Structure

In OpenFOAM, the file `pointDisplacement` is only needed in this case to set the ratio f_n/f_v and the oscillation amplitude (A/D). Additionally, this file is using the motion solver `displacementLaplacian` from the file `constant/dynamicMeshDict`. However, the file `pointMotionU` is used when the motion solver uses velocity based motion calculations of the point mesh. The idea is that the motion can either be done based on moving the cells

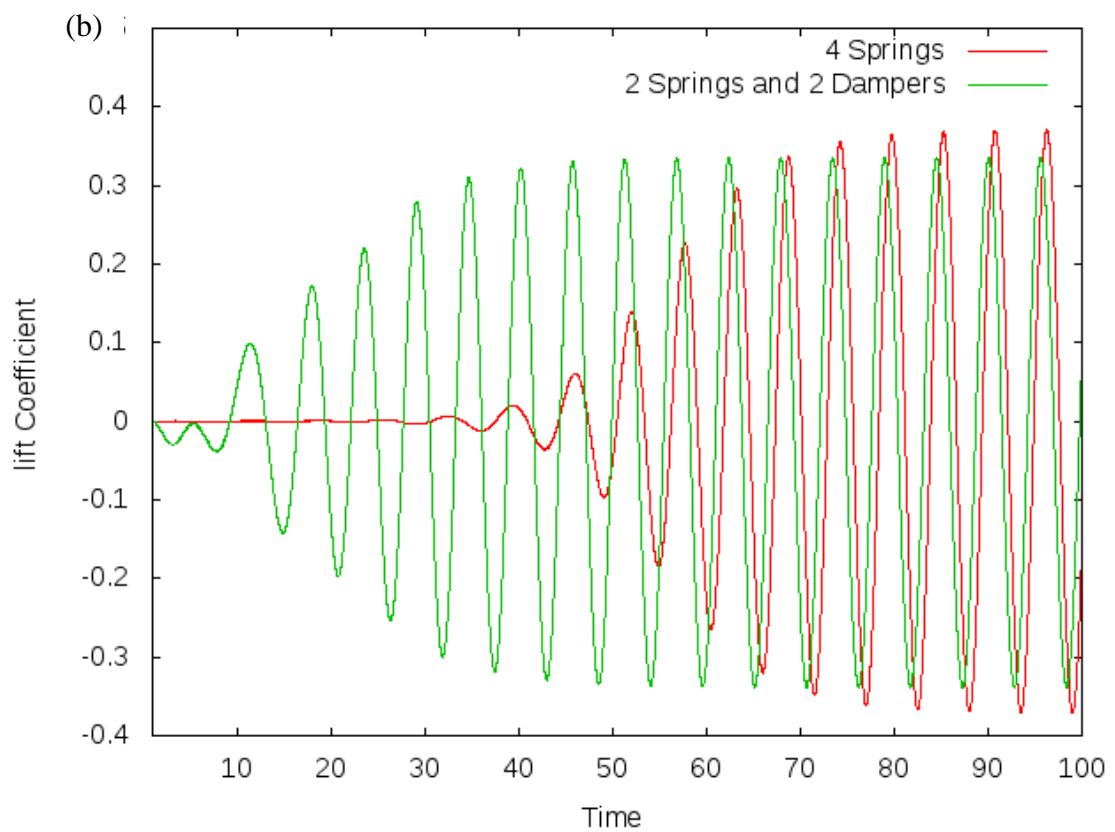
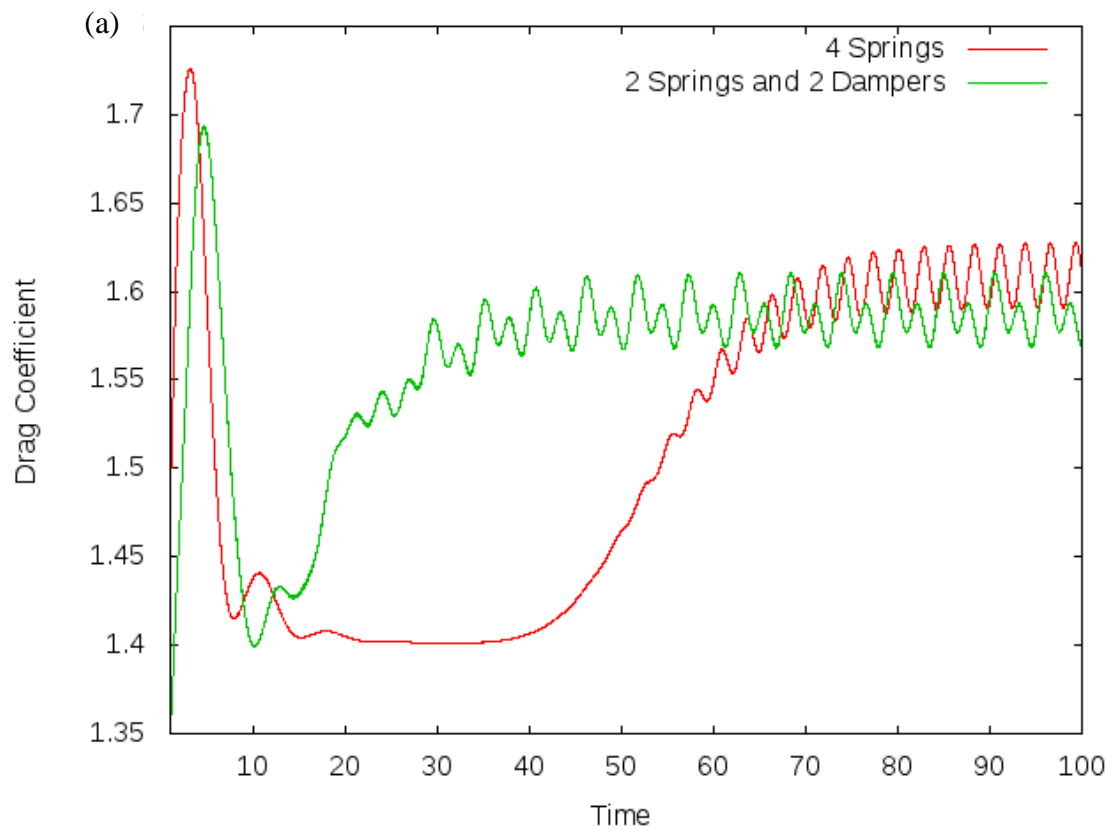


Figure 3.24: (a) Drag and (b) lift coefficients for 2DOF systems

themselves or move the points instead. Even though the mesh can move either based on cells or points, the user can also (sometimes) do the motion calculations either based on velocity “pointMotionU” or based on absolute (or relative) position “pointDisplacement” (refer to Appendix 3.J)

3.9.4.1 Results and Discussions

This case example was simulated for the same Reynolds number ($Re = 100$). The simulation results are presented in Figures 3.25 and 3.26 to present force coefficients. The natural frequency (f_n) result is 0.482 Hz. This case has a slightly high vertical displacement as presented in Figure 3.26 and due to the extremely displaced mesh, the simulation shows no movement at the beginning. This indicated that the cylinder is in the resonance mode. The frequency ratio ($F_r = f_n / f_v$) where f_n represents the natural frequency of the cylinder and f_v shows the vortex shedding frequency corresponding to the static cylinder). Strouhal number is equal to 0.167, which means that the vortex shedding frequency $f_v = 0.167$ Hz. The natural frequency, f_n , has the value 0.482 Hz. The cylinder amplitude reaches the maximum value when $f_n / f_v = 2.87$.

The time history of the laminar flow regime is also presented in Table 3.3. The reduced velocity can be calculated from Equation (3.15) and it is 5.988 in this case.

From the two graphs, it is possible to infer how the vortices are being generated and released as the cylinder moves up and down. To make this inference, the following can be observed:

1. From Figure 3.25 it can be seen that the maximum drag coefficient peaks match the minimum and maximum lift peaks, as well as the minimum drag coefficient peaks match the points at which the lift is zero.
2. Related to this, it can be observed in Figure 3.26 that the peaks in lift match the peaks in the displacement of the cylinder. This means that the peaks in displacement match the peaks in drag coefficient.
3. The zero-lift positions related to when the cylinder is passing the middle axis of forced oscillation.

Therefore, from these plotted relations, it can be inferred:

1. When it is zero-lift, is when the vortex generating forces balance each other out, as it can be visible in Table 3.3, at around 10s, which is when the wake near the cylinder is almost aligned with the fluid flow which is going from the left to the right.
2. When the drag and lift coefficients are maximum, it is when:
 - i. a region of high-velocity stream is going over the top of the cylinder when this is at the top;
 - ii. or when the stream of high velocity is going to the cylinder, when this is at the bottom, which can be observed at around 30s in Table 3.3.
3. These peaks of drag and lift coefficients are consistent with the behaviour of wings and airfoils in similar working conditions, namely when comparing the maximum value for those wings/airfoils [178].

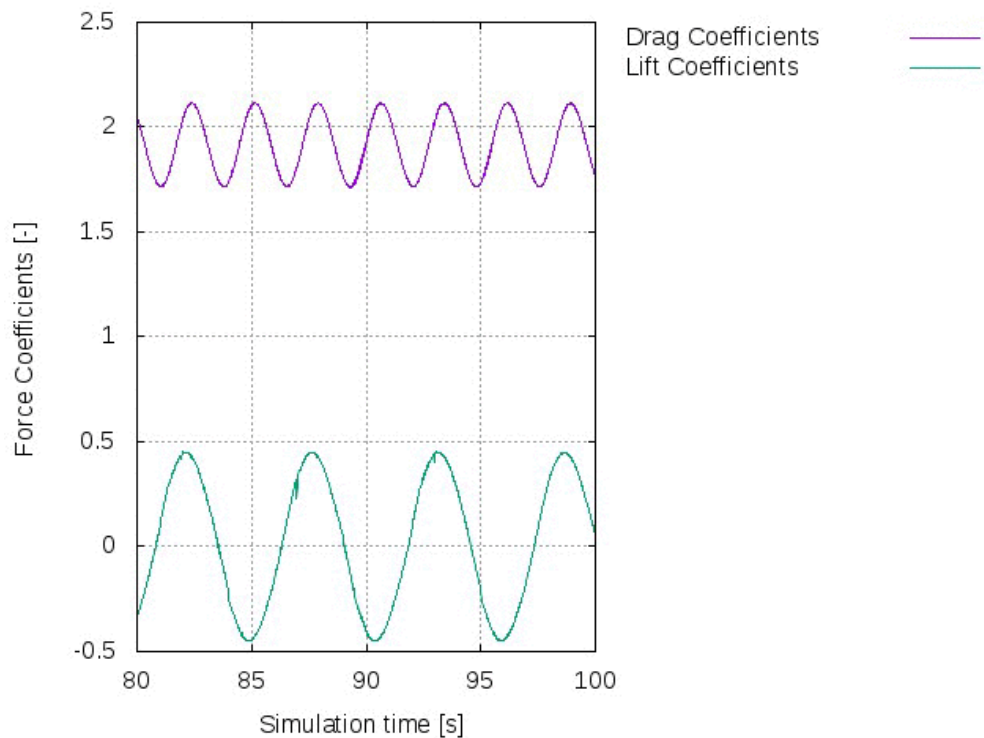


Figure 3.25: Force coefficient of forced oscillation cylinder case at laminar flow with $Re = 100$

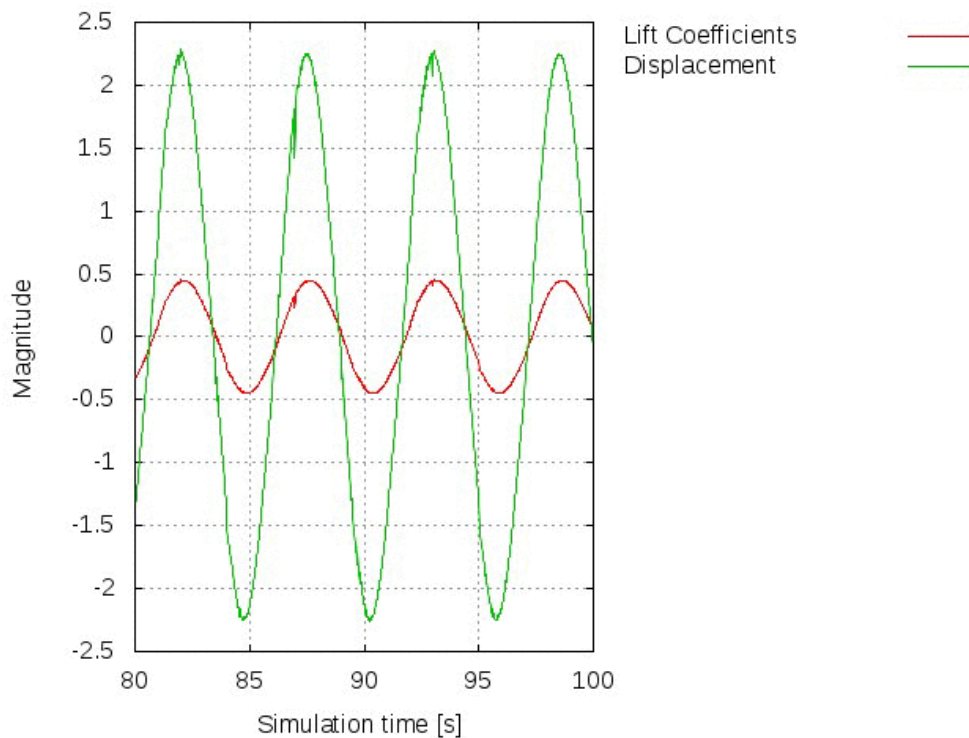


Figure 3.26: Vertical displacement and lift coefficient time histories of the forced oscillation case

3.10 Chapter Summary

Understanding the vortex-induced vibration phenomenon is important as it plays a critical role in designing the fluid-structure interaction models. The behaviours of both static cylinder and motion cylinder have been examined and discussed its inline and transverse forces.

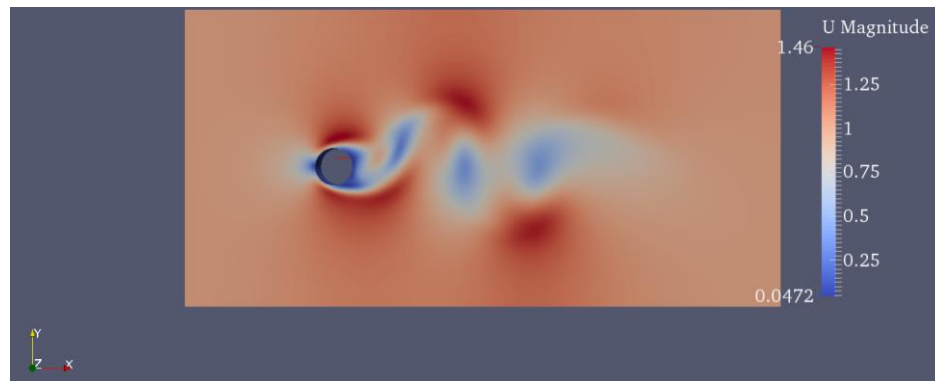
For the stationary circular cylinder case, both drag and lift coefficients represent a good agreement compared to other studies at Reynolds number equals 100 and 200. However, there is a significant difference at $Re = 1000$.

From the variously reported VIV simulations, it is possible to ascertain that:

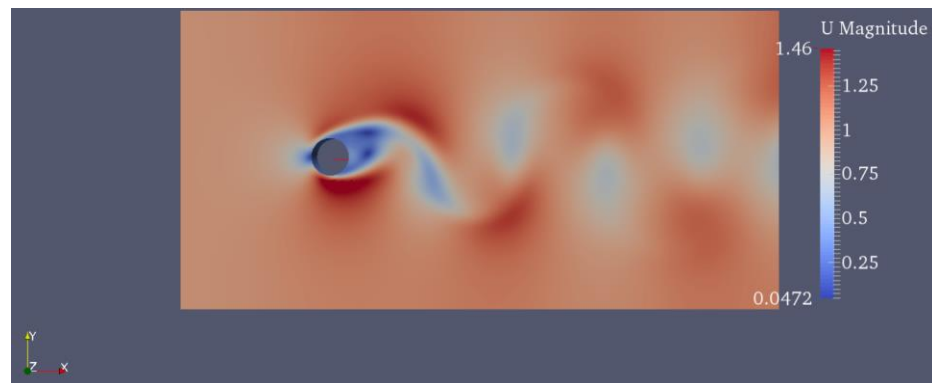
- Both damping and stiffness will affect how the cylinder will behave within the flow, where the main noticeable differences will be:

- response time, in regard to the initial speed with which the mechanism (cylinder + springs) reacts to the flow, as well as in regard to when it will stabilise at the uniform periodic oscillation.

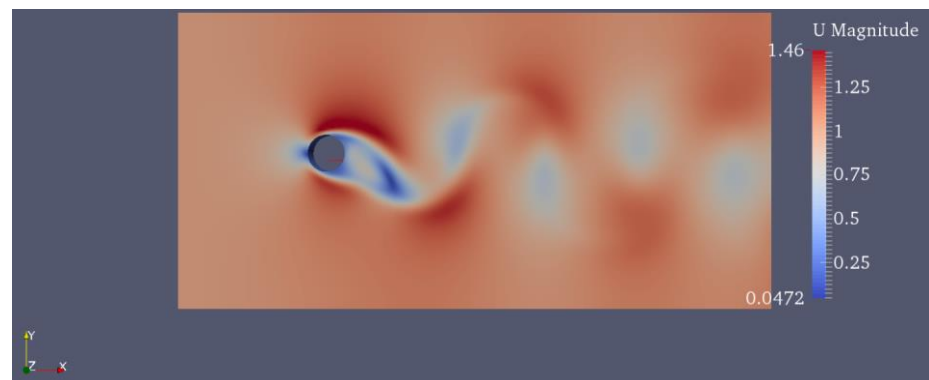
Table 3.3: Flow regimes of forced vibration cylinder case using ParaView



$t = 10\text{s}$



$t = 30\text{s}$



$t = 100\text{s}$

- the range of amplitudes of oscillation.
- maximum drag coefficient in the uniform periodic oscillation (after the system has stabilised)
- Both the reduction of damping and stiffness will reduce the maximum drag and lift coefficients after the flow has stabilised.
- However, for the 2DOF case:
 - reduction of damping can result in a much higher drag coefficient during the initial phase;
 - while the reduction of the stiffness will reduce the highest drag coefficient during the initial phase.
- The lift coefficient seems to be less affected by the changes in damping and stiffness when compared to the scale of the change in the drag coefficient results. Based on this assessment, it looks like the lift coefficient is mostly associated with the cylinder geometry and the fluid properties, then the influence of the springs and dampers, although it strongly depends on the cylinders ability to follow the vortices in its wake.
- Depending on the desired performance for this mechanism, the optimum design decisions could be as follows:
 - If the mechanism is meant to have as much drag as possible, then it would be best for the dampers to be minimum or be removed, while having fairly stiff springs. Although at this point, might as well fixate the cylinder in place, instead of using springs.
 - If the mechanism is meant to have minimum drag, then use moderate damping values and fairly relaxed springs. Although, keep in mind that a complete study wasn't conducted for this kind of scenario and there could be a fairly degraded performance in case the springs were replaced with just having 4 dampers.

Finally, to avoid resonance, it is important to keep the reduced velocity value is less than 4.

Chapter 4

Two- and Three-Dimensional Benchmarking Models

In the previous chapter, the simulation of the flow over the two-dimensional rigid cylinder and the vortex-induced vibration of the cylinder have been studied. However, the focus on the fluid and solid behaviours were simply presented in the hydrodynamic forces. In reality, the fluid-structure interaction problems display a strong coupling between the fluid and the structure due to the forces acting from fluid to solid and then causing the structural geometry deformation. This chapter discusses four strongly coupled fluid-structure interaction numerical examples; two models are in two-dimensional space and two are their extension in three-dimensional space.

4.1 2D Example of Flow-Induced Oscillations of a Flexible Tail behind a Block

This benchmark has been used for the fluid-structure interaction validation in many studies started by Wall [179] and later by many others such as [42-43,45,106,180-187] to examine the solution strategies for FSI problems.

4.1.1 Geometry and Boundary Conditions

As shown in Figure 4.1, this model problem consists of a flexible thin tail attached to a square block support in the middle of the downstream face. This model example is immersed in an incompressible fluid flow which is considered initially at the rest position. The vortices in the square support wake communicate with the attached tail, and large oscillations amplitude will excite. In other words, the vortices that separate from the square bluff body corner will create the lift forces which cause the flexible plate oscillation [106,188].

The fluid geometry is bounded by the inlet velocity, outlet pressure, and the walls. The top and bottom walls are applied to the slip boundary conditions and no-slip conditions are defined on the square block interface and the flexible tail. At the inlet, the uniform

flow is distributed with stream-wise mean velocity $u = u_\infty \text{ m/s}^2$; here $u_\infty = 0.513 \text{ m/s}^2$. Whereas at the outlet, the pressure is zero and the Neumann zero is considered for the velocity.

4.1.2 Problem Definition

The physical properties of flow, fluid and solid are represented in Table 4.1 The critical Reynolds number is less than the given Reynolds number value; i.e., ≈ 333 . Thus, as mentioned previously, the Von Karman vortex street will be produced by the flow separation from the square support corner. This vortex behaviour of the pressure and viscous stress near the block wake causes the vibrations of the attached tail. It is also important to note that the tail density is 84.75 times larger than fluid density and the tail length is 66.67 times more than its thickness.

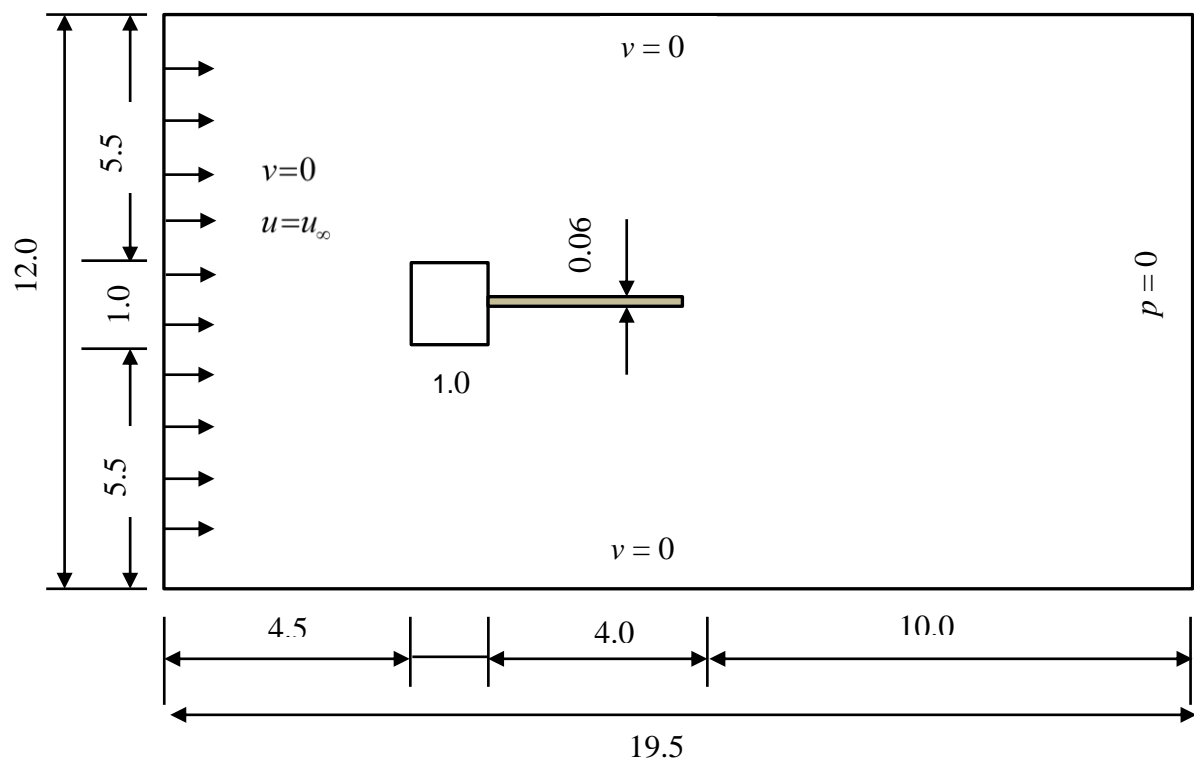


Figure 4.1: Computational domain and boundary conditions of tail behind block

4.1.3 Mesh Generation

Mesh for both fluid and solid domains are generated by using the blockMesh utility in OpenFOAM. The fluid domain includes 18,010 hexahedral elements and the structure domain contains 80 hexahedral cells. However, the fine mesh is created in order to compare results between both. After refinement, the fluid domain consists of 157,110 cells and the structure domain contains 240 hexahedral elements. The fluid domain initial fine mesh is shown in Figure 4.2a representing the mesh refined near the block wall and the fluid-solid interface, and Figure 4.2b illustrates the deformed mesh at $t = 6s$ where the good quality of the mesh is preserved. This mesh quality is shown by the skewness value given by checkMesh utility which is not exceeding 0.3, and this emphasizes a good mesh quality.

4.1.4 Spatial and Temporal Discretisation

A second order implicit scheme is used to perform the temporal discretisation that is unconditionally stable. The time-step is updated by defining the maximum value of

Table 4.1: Physical properties of the tail behind block model

Domain	Parameter	Value	Unit
Fluid	ρ_f	1.18	Kg m^{-3}
	ν_f	1.54×10^{-5}	$\text{m}^2 \text{s}^{-1}$
Solid	ρ_s	100	Kg m^{-3}
	ν_s	0.35	-
	E_s	2.5×10^9	Pa
Flow	u_∞	0.513	m s^{-1}
	Re	332.6	-

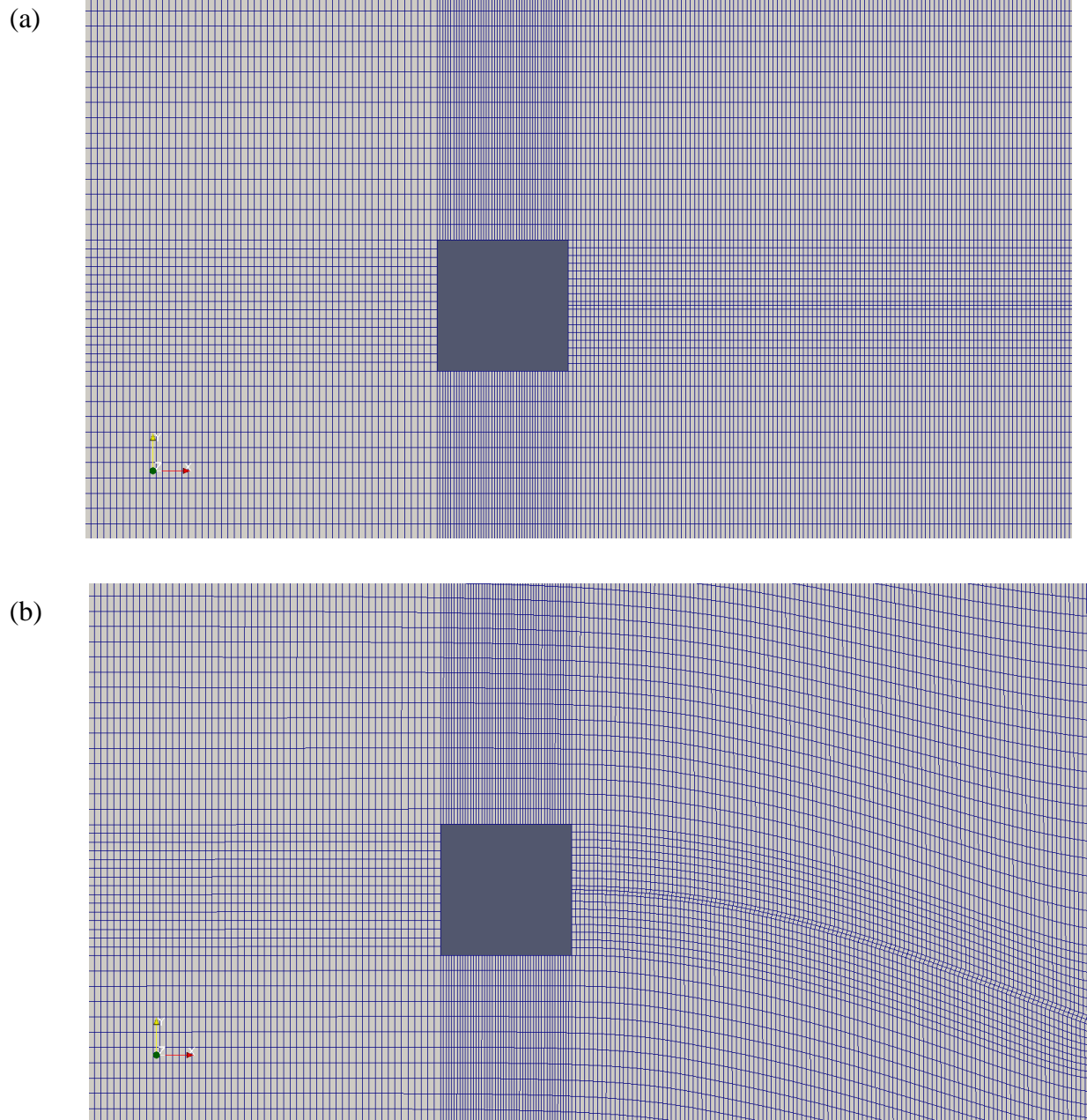


Figure 4.2: 2D tail attached to solid support computational domain for refined mesh at (a) $t = 0s$ and (b) $t = 6s$

Courant number $Co_{max} = 0.2$ (Equation 2.30). Therefore, by setting a fixed value of Co_{max} , then the time-step Δt can be considered for all fluid cells.

For the symmetric matrices, the GAMG iterative solver along with GaussSeidel smoother used in order to solve the pressure equation. The criterion of the convergence for the pressure fixed as 10^{-6} . For the asymmetric matrices, the PBiCG used for the

velocity-pressure coupling equation along with the DILU pre-conditioner and the criterion of convergence for the velocity is 10^{-6} .

The GAMG used for the mesh motion solver along with the GaussSeidel smoother and convergence criteria is also given as 10^{-6} . Further information on algorithms and numerical schemes can be found in Kassiotis [128], Bos [117] and user guide of OpenFOAM [125].

4.1.5 Post-Processing, Results and Discussions

Figure 4.3a shows the conventional flow pattern of the tail oscillation once the vibration is fully formed. The velocity that is represented in the figure shows the unsteady flow behaviour which leads to unsteady vibrations of the elastic tail. Consequently, these vibrations cause in the flow velocity gradient, toward the tail tip, effect on the new vortices generation. It could be examined that the Von Karman vortex street that is generated by the solid block is distributed near the lower and upper regions because of the elastic tail flapping. Figure 4.3b shows the pressure where the vortex shedding can be presented downstream from the square block. The Von Karman vortex street is generated by the bluff body which is distributed near the lower and upper walls due to elastic tail flapping. Furthermore, the velocity gradient is generated by the tail oscillation at its tip.

When comparing this case with the cylinder simulations reported in the previous chapter, this case is as if there were two cylinders: one stationary cylinder, but with a square section, along with a second cylinder which has been flattened and is represented on the second half of tail on the right. With this comparison in mind, it is possible to observe that:

1. The stationary square cylinder is seeding the usual Von Karman street that is common to this kind of flows.
2. When comparing to the springs on the oscillating cylinder cases, the non-fixed part of the tail is oscillating with the generation, and release of the vortices with a solid stiffness.

3. In addition, given the very small thickness of the tail, results in the tip forming a pattern similar to the tip of a pen through sand.
4. As vortices are generated with greater intensity on one side of the elastic tail, the tail will bend towards that direction, until it reaches the maximum deflection achievable for the vortices which have generated and released on that side, similarly to when a spring reaches the maximum compression/extension for the force applied to it.
5. The flexible tail moves in the opposite direction when the forces on that side become large enough and have resulted in large enough vortices to pull the tail back towards where it was originally.

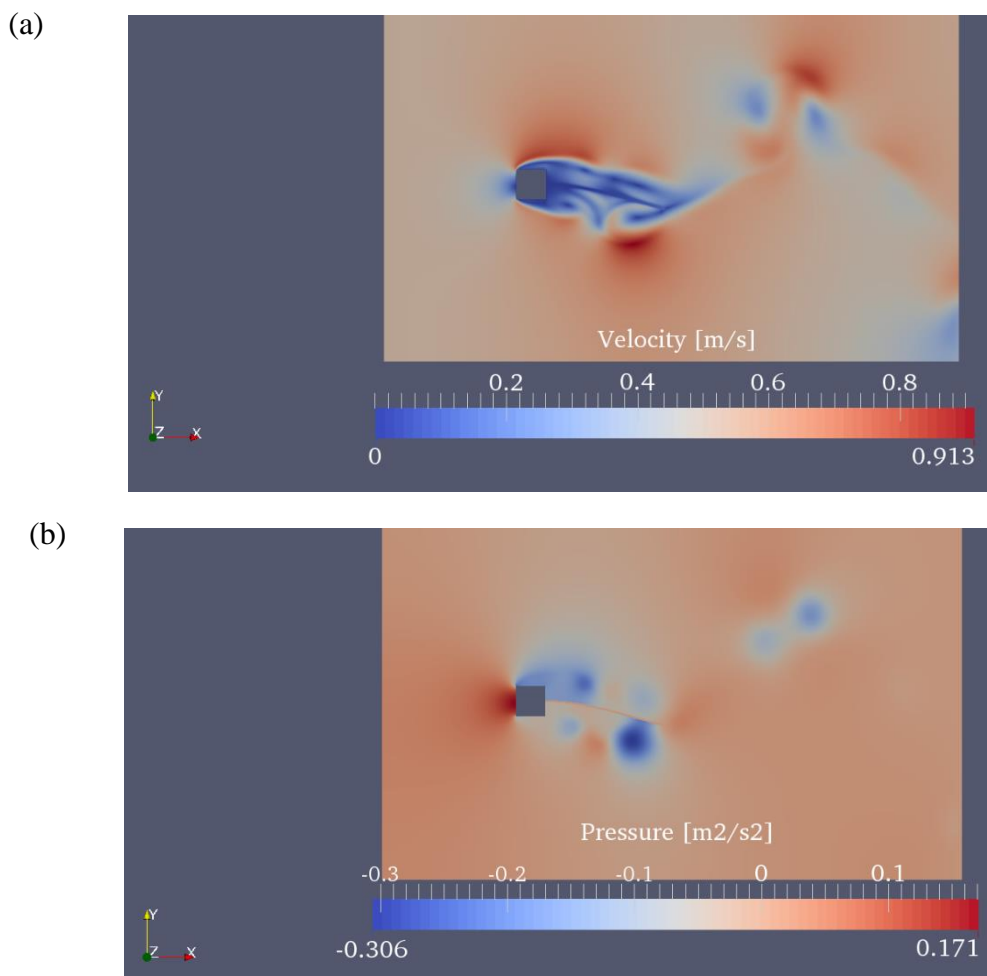


Figure 4.3: Oscillation of a tail attached to a solid block at $t = 6s$: (a) velocity and (b) pressure/density

6. Effectively, the tail will bend one way or another, depending on the pressure balance on each side of the elastic tail, for example, if the total pressure is smaller on the top side than on the bottom, then the end of the tail will move upwards. Although keep in mind that it will favour the pressure/forces exerted near the tip, since it is where the highest moment-force can be imposed with the maximum displacement.

The maximum displacement and the vibration frequency are presented in Table 4.2. In the present study, the maximum tip displacement is 0.8552 cm for the coarse mesh and 1.132 cm for the fine mesh. The tail oscillation frequency value is 4 s^{-1} and 3.5 s^{-1} for the coarse and fine mesh, respectively. In comparison to other studies presented in Table 4.2, the maximum tip tail motion values are within the range [0.95cm, 1.25cm], so more refinement mesh is expected to be necessary to achieve the same range.

The simulation has been done for both coarse and fine meshes at $\Delta t = 0.001 \text{ s}$. Figure 4.4 shows the comparison of displacement results of the flexible tail attached to the square bluff body.

The reason for such a strict dependency on mesh resolution has to do with the numerical modelling which was used for these simulations. Given that the Reynolds number indicates that the flow regime is laminar, it ended up revealing that there were considerably sized vortices being generated, which consequently requires either:

- a properly refined mesh (in function of the level of precision that is aimed to be reached);
- or requires a different numerical model, specifically a turbulence model which could estimate the intensity of the vortices being generated, instead of having to depend on the sizes of the cells.

Therefore, since laminar modelling was used for these simulations, the consequence was that only after conducting an exhaustive battery of simulations for testing how much the results changed in function of mesh resolution, could an accurate result be reached. However, this was not conducted due to the computational requirements which would be necessary, as well as the extensive simulation time it would take.

Table 4.2: Comparison present work results with other literature for the elastic tail attached to the solid block

Literature	Fluid flow	Structural deformation	Coupling approach	Frequency (s ⁻¹)	Displacement (cm)
Dettmer & Peric [43]	Stabilised FEM	FEM	Partitioned NR ¹	3.03	1.25
Habchi <i>et al.</i> [45]	FVM	FVM	Partitioned BGS ³	3.25	1.02
Kassiotis <i>et al.</i> [106]	FVM	FEM	Partitioned BGS	2.98	1.05
Matthies & Steindorf [182]	FVM	FEM	Partitioned BN ²	3.13	1.18
Oliver <i>et al.</i> [183]	FVM	FVM	Partitioned BGS	3.17	0.95
Walhorn <i>et al.</i> [189]	Stabilised FEM	FEM	Partitioned BGS	3.14	1.02
Wall [179]	Stabilised FEM	FEM	Partitioned BGS	2.99	1.22
Wood <i>et al.</i> [186]	FVM	FEM	Partitioned BGS	2.94	1.15
Yvin [188]	FVM	FVM	Partitioned BGS	3.16	1.20
Present work (coarse mesh)	FVM	FVM	Partitioned BGS	4.0	0.8552
Present work (fine mesh)	FVM	FVM	Partitioned BGS	3.5	1.132

¹ Newton-Raphson

² Block-Newton

³ Block Gauss-Seidal

4.2 3D Elastic Cantilever Plate Attached to a Solid Block

This three-dimensional numerical approach follows the two-dimensional example model presented in the previous section. This test model is based on the studies done by Kassiotis *et al.* [106,190] and von Scheven and Ramm [44]. There is an unstable manner could be expected by this system like the sharp angles of the solid block which lead to high vorticity in the fluid flow through the slender plate.

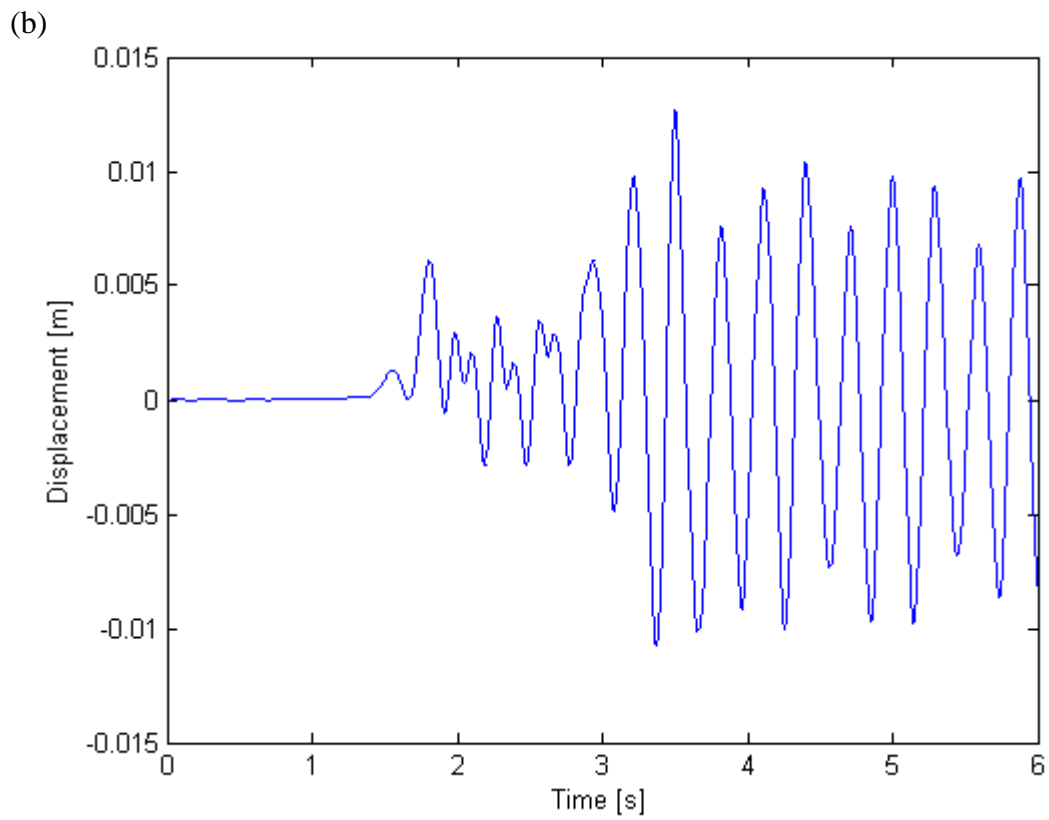
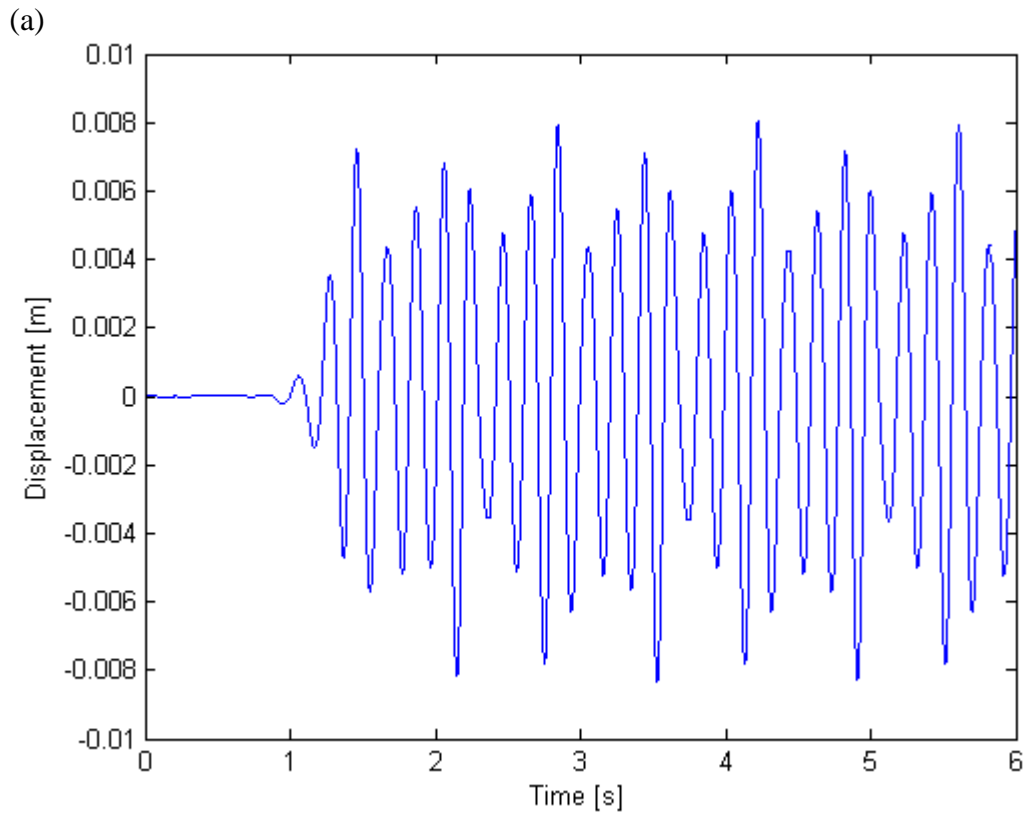


Figure 4.4: Comparison results on the tip displacement, (a) coarse mesh (b) fine mesh

4.2.1 Model Geometry and Boundary Conditions

The geometrical properties and the boundary conditions of the case are shown in Figure 4.5. This model problem consists of a light-weight and stiff elastic cantilever structure fixed to the solid support and fully immersed in a fluid. The domain of the fluid is 20cm × 11cm × 11cm. The structure's domain includes the slender plate with dimensions of 4cm × 3cm and its thickness is 0.06cm. This plate is attached to one end of the rigid block with dimensions 3cm × 3cm and 1cm thickness.

In this model example, the light wind should be applied to simulate the flapping of the flexible plate and the displacement of the flexible body results in the uniform velocity u_∞

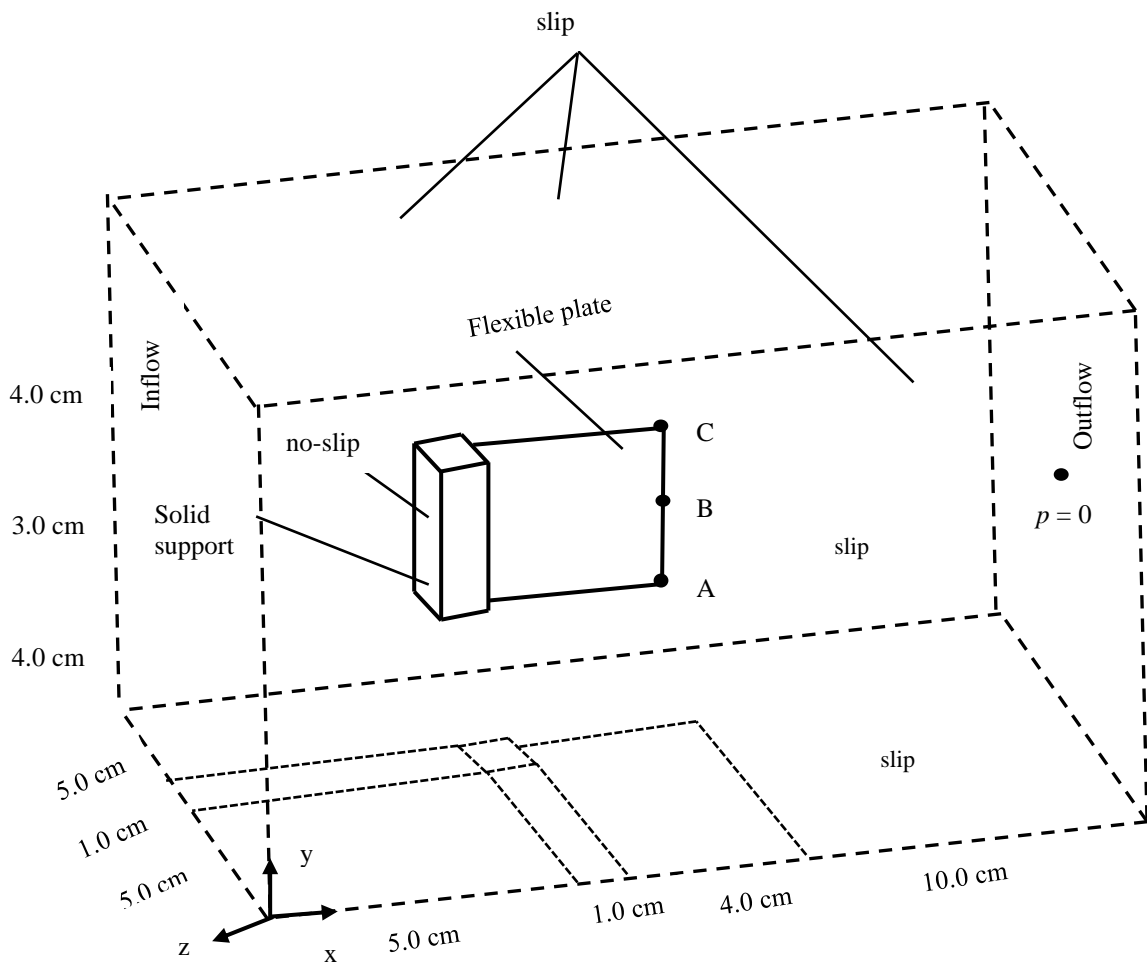


Figure 4.5: 3D flexible plate model: geometry and boundary conditions

that gradually applied over time. To solve this fluid-structure interaction case numerically, the plate weight should be neglected, and a moderate wind speed should be only considered as 100 cm/s.

The material properties of the fluid and solid are presented in Table 4.3. The fluid properties are corresponding to the air properties at 20° C.

4.2.2 Mesh Generation

Both fluid and solid domain meshes were generated by using blockMesh utility. It is quite complicated to write this type of meshes in the blockMeshDict dictionary but divided the domains into levels will help to make it much easier as illustrated in Figure 4.6 and the way of writing points in the blockMeshDict file is shown in Appendix 4.A.

Then, the blockMesh results shown that the mesh in total contains 817,168 nodes and 785,700 hexahedral cells for the fluid domain, and 5,612 points and 2,700 hexahedral

Table 4.3: Material properties of the 3D flexible plate attached to solid block

Domain	Parameter	Value	Unit
Fluid	ρ_f	1.18	kg m ⁻³
	ν_f	1.54 x 10 ⁻⁵	m ² s ⁻¹
Solid	ρ_s	2000	kg m ⁻³
	ν_s	0.35	-
	E_s	2.0 x 10 ¹⁰	Pa
Flow	u_∞	1.0	m s ⁻¹
	Re	649.35	-

elements for the solid domain as presented in Figure 4.8. For both fluid and solid domains, the boundary conditions are the same, i.e. the inflow is being fixed, the pressure in the outflow is also fixed at one point, and all other defined boundaries are slip (Figure 4.5). Moreover, the solid block in the fluid domain is fixed with no-slip boundary conditions and the elastic plate is chosen to be included as a part of the solid in the fluid-structure interface.

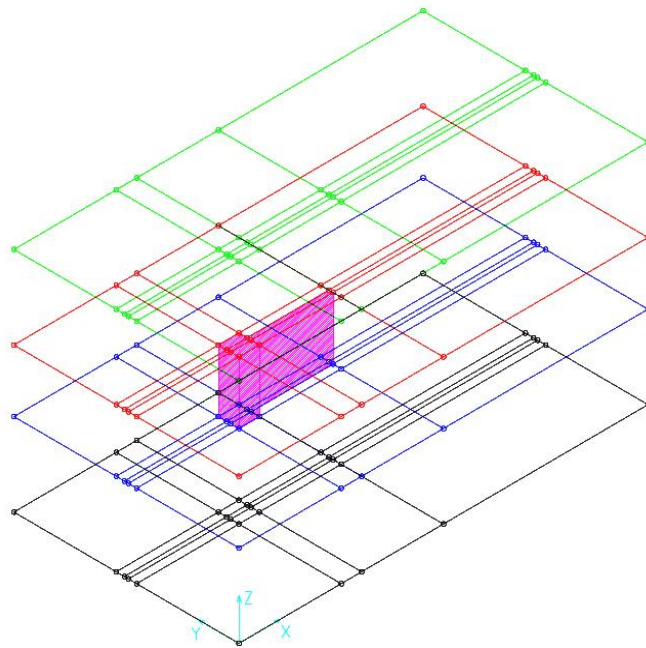


Figure 4.6: 3D elastic plate case block levels

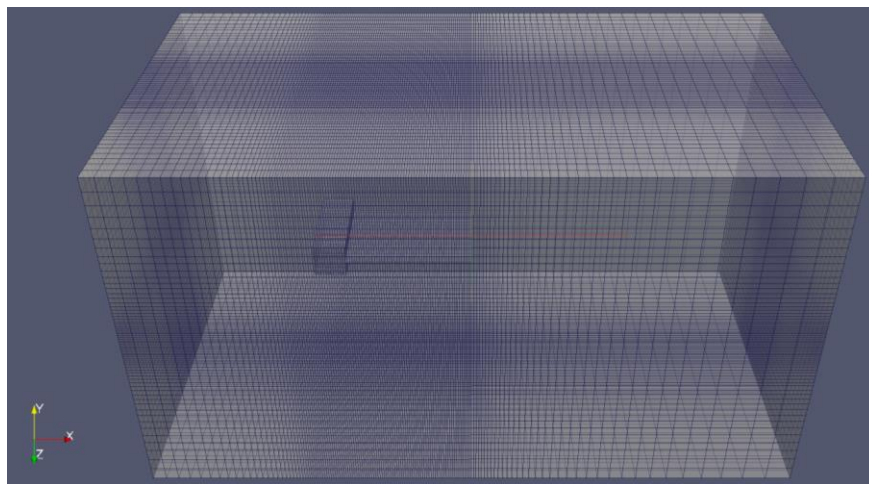


Figure 4.7: 3D tail attached to solid block mesh in ParaView

The mesh domain of fluid and solid is represented by ParaView and shown in Figure 4.7 and Figure 4.8 illustrates the mesh of the plate and the solid support inside the fluid region.

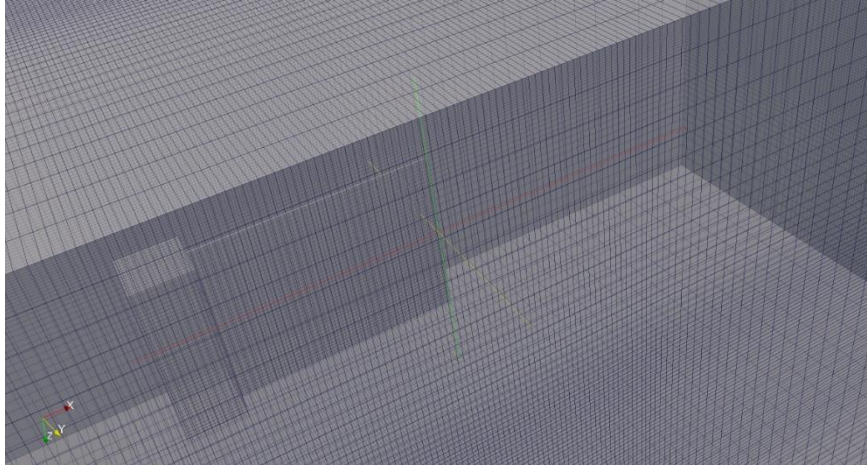


Figure 4.8: Meshing representation of the plate and the solid block in ParaView

In comparison to other 3D plate attached to support block models, Tables 4.4 and 4.5 show the differences between this work case and other studies.

Table 4.4: Comparing of fluid mesh properties

Mesh	Incompressible fluid elements	D.O.F	Solution strategy	Δt
Kassiotis <i>et. al</i> [106,190]	290×10^3	1159×10^3	Finite volume	0.001
Von Scheven [191]	200×10^3	794×10^3 (total)	Finite elements	0.01
Taylor [192] Mpap ¹	166×10^3	751×10^3 (total)	Finite elements	0.005
Present work (blockMesh)	785,700 hexahedral	n/a	Finite volume	0.001

¹Multiphysics Analysis Program used to denote solvers in Taylor's work [192]

4.2.3 Case Implementation

The 3D plate attached to the support solid block model was implemented to simulate using `fsiFoam` solver in `FluidSolidInteraction` library in `foam-extend 4.0` (the latest version for fluid-structure interaction library). The implementation structures are shown in Appendix 4.B. Additionally, different files related to the case implementation are shown in Appendices 4.C - 4.H.

In the fluid domain where x is greater than 6.0 cm, the ALE formulation is employed for the spatial discretisation and the temporal discretisation, 1,200 time-steps with $\Delta t = 0.001$ s are used.

Table 4.5: Comparing of structure mesh properties

Mesh	Solid elements	D.O.F	Solution strategy	Coupling strategy
Kassiotis <i>et. al</i> [106,190]	300 27-Noded Quadratic	7425	St. Venant Kirchhoff	DMFT-BGS with Aitken relaxation
Von Scheven [191]	432 7 Parameter Shell	n/a	St. Venant Kirchhoff	BGS with Aitken relaxation
Taylor [192] Mpap ¹	300 20-Noded Quadratic	n/a	Neo-Hooke Elasticity	MN ¹ / GS ² / WC ³
Present work (blockMesh)	2,700 hexahedral	n/a	St. Venant Kirchhoff	BGS with IQN-ILS

¹Monolithic Newton

²Gauss-Seidel

³Weakly coupled

4.2.4 Simulation Problems

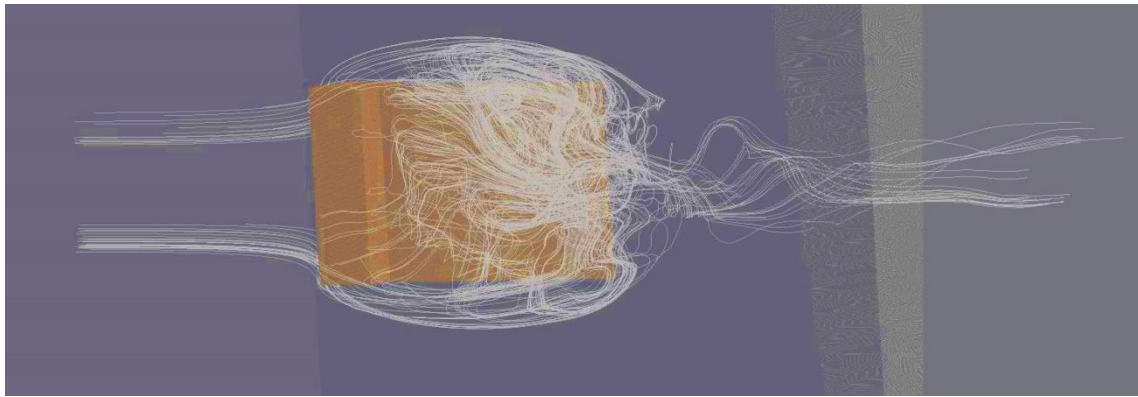
Since the foam-extend 4.0 version is the latest version for FSI solvers, there are some problems were faced during run that case as the following

- Still more than 1082h wall-clock time on only 1 core of CVC2EC remote computer is required for the 1,200 time-steps.
- For foam-extend 4.0, running in parallel does not show any considerable performance increase (does not seem to scale with 2-4 cores, for a mesh with more than 300,000 cells). As regards speed up, the parallel efficiency of all OpenFOAM and foam-extend solvers depends on the number of cells per processors; in this case, it would Gauss that the solid has much fewer cells than the fluid.
- The speed-up is examined in some different cases where for example that solid has a similar number of cells to the fluid.
- Increasing the number of cells in the solid region will unlikely to improve the time needed to simulate in parallel.
- It is believed that the restart option was not of particular interest when developing the FSI solver so it was never checked, though it should be possible to solve in the code.

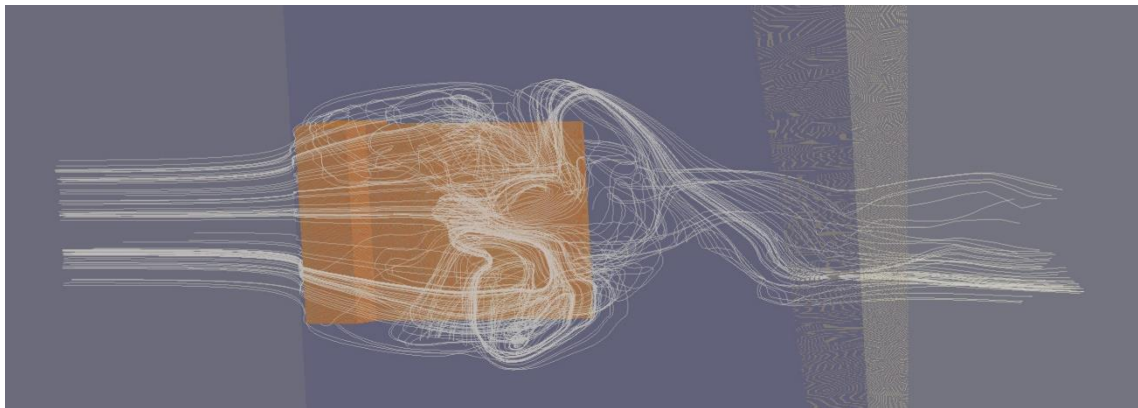
4.2.5 Results and Discussions

The simulation is carried out with 0.001s coupling time-step. The coupling scheme that used is IQN-ILS which is presented in the fluidSolidInteraction library in foam-extend 4.0. The value of the initial relaxation is 0.50.

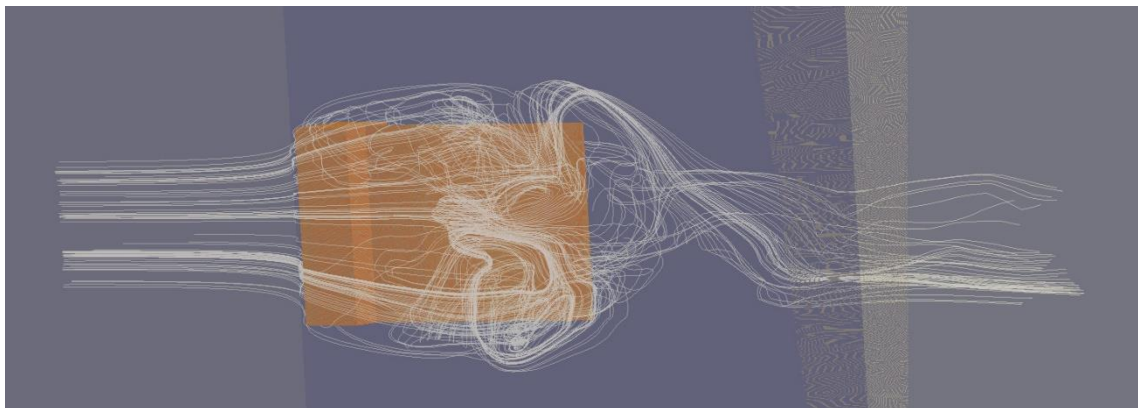
In the 3D flow for FSI problems, it is not easy to select the relevant results. Therefore, Figure 4.16 illustrates the stream-tubes over three points (0.10, 0.055, 0.07), (0.10, 0.055, 0.055) and (0.10, 0.055, 0.04) along with the deformation of the plate shape to get an approximate illustration of the results.



$t = 5s$



$t = 10$



$t = 12s$

Figure 4.9: 3D plate in wind: structural motion and stream-tube snapshots for some simulation time

Figure 4.9 displays the perpendicular displacement results obtained from the simulation. After the initial deformation, the displacement of the elastic plate damps under the zero value. The deformation of points A, B and C located at the free-end of the plate as represented in Figure 4.5. These show the flexible plate motion corresponding to the

twisted form. It is important to note that it is not easy to find the exact solution for the 3D flow with high Reynolds number. The results in [191] indicated that there are twisted modes appear in the plate after some time and that is not validated in this work even with the same geometrical and physical properties being used. In this work, the amplitude of the motion is about 20 times more than the results obtained in [191].

The 3D plate model results are summarised in Table 4.6 along with other results obtained in [44,106,190-192]. It could be noted that the results do not show full agreement between the case model presentations. It is also important to note that the behaviour of the flexible plate in this work and in [106,190] was tested with transitional displacement and no rotation in the solid. However, the results obtained in [44,192] present the transitional and torsional rotation.

The disagreement magnitude differences between the 3D plate model presentations do not completely reflect the accuracy in different solution strategies. Furthermore, the time intervals that are presented by Von Scheven [191] and Kassiotis *et al.* [106,190] could be insufficient to define the final system response. There are two reasons for this deduction:

1. All 3 Taylors [192] simulations use extended simulation times, ranging from 20s to 75s, while Von Scheven [191] and Kassiotis *et al.* [106,190] used only 6s and 12s respectively.
2. In order for torsion to occur, it is necessary that at some points in time the fluid flow will create cross-over flow, in order for vortices to stop to be generated in a synchronous way along the plate surface. For example, given that the surface area near points A and C have a smaller contact resistance to the flow than point B, it is expected that the dimensions of the vortices generated near A and C should eventually have different shapes from those near point B.

As for the overall disagreement in the results between this work and the other studies, this may very likely be related to what was diagnosed in the 2D plate simulations. Specifically, it was assessed how closely tied is the mesh refinement/resolution to the response time of the plate and that even though the fluid flow is laminar (Reynolds numbers well below 1000), the presence of vortices indicates that a turbulence model should still have been used for properly modelling their presence without the need to

increase drastically the mesh refinement/resolution. Therefore, in order to get better results for this 3D plate case, it would be necessary to either:

- use a turbulence model for the same mesh;
- or use a more refined mesh, in order to have enough mesh resolution for vortices to be revealed in the flow and to break the synchronous flow profile witnessed in this case.

Furthermore, when correlating the results presented in Figures 4.9 and 4.10, it is possible to see how much the fluid is deflecting the plate, as if it was a fully laminar flow, with no vortices. This reinforces the reasoning regarding the need for turbulence modelling or increased mesh resolution, given that the streamlines do not give any indication of vortices being generated. When compared to the VIV simulations, the ability to oscillate along with vortices will lead to the reduction of the amplitude of the displacement, which would have benefited in this case.

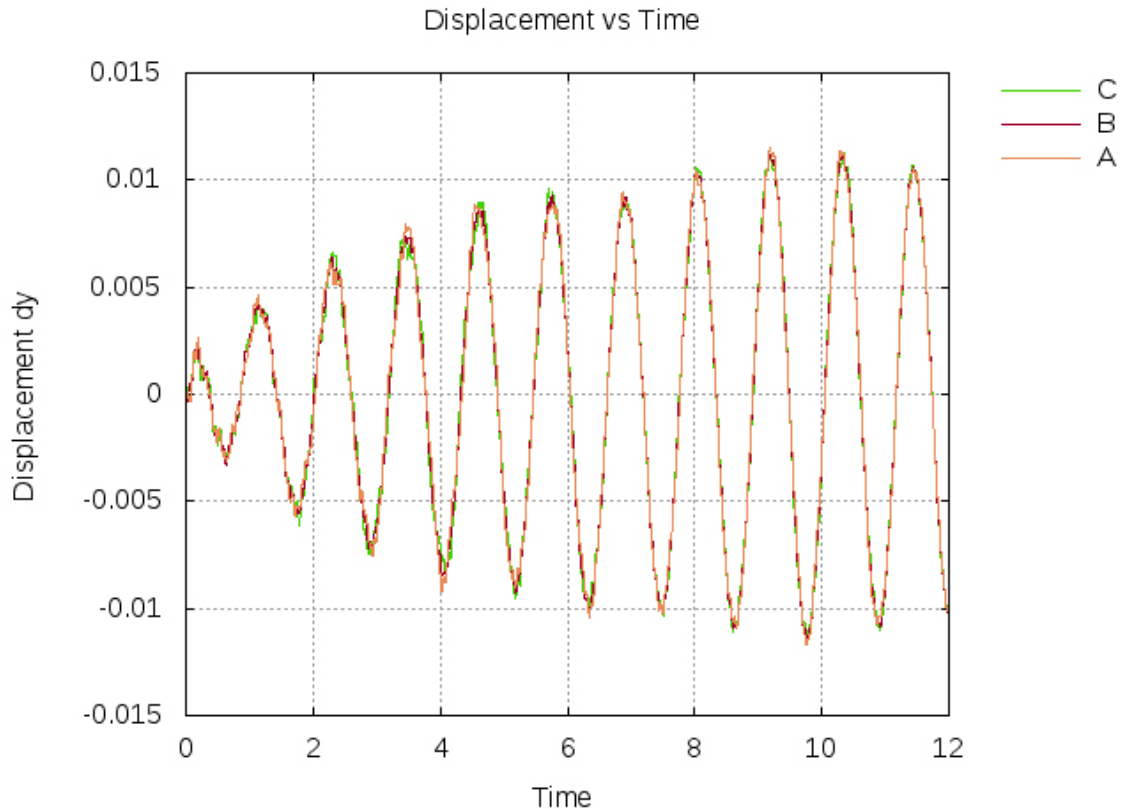


Figure 4.10: Perpendicular displacement in meter of flexible plate over simulation time in second for points A = (0.10, 0.055, 0.07), B = (0.10, 0.055, 0.055) and C = (0.10, 0.055, 0.04)

However, it was not possible to confirm if this reasoning, given how long it takes to run each 3D simulation. As a reference, looking at the 2D plate case, the mesh refinement was increased from the coarse mesh with 18,010 cells to 157,110 cells for the fine mesh, in order to get results somewhat closer to the ones in the reference articles [106,190-191]. This meant that the total wall clock time needed to simulate increased in the same proportion, nearly 9 times longer to simulate. Increasing the mesh resolution of this 3D case could lead to simulations taking 5 to 10 times longer than the 45 days (1082h) it took to conduct this 3D simulation. This due to the limitations of the software, as stated in Section 4.2.4.

On the other hand, introducing a turbulence model would have required redoing all of the work from the start, as well as fixing any additional issues that would likely occur, when switching to the new workflow.

Table 4.6: Results comparison for the 3D flexible plate attached to solid block model

System	Oscillation frequency (Hz)	Oscillation amplitude (cm)	Time interval (s)
Kassiotis <i>et al.</i> FSI [106,190]	4.0	0.5	6
Mpap 88k Fluid, 20N Solid [192]*	1.1	0.05 (damped to 0)	20
Mpap 166k Fluid, 8N Fbar Solid [192]*	2.1	0.07	75
Mpap 166k Fluid, 20N Solid [192]*	0.7	0.08	70
Von Scheven FSI [44,191]	0.9 or 10 (2 methods)	0.1	12
Present work FSI	0.87	0.55	12

*According to Taylor [192], three types of the solid mesh were studied He suggested that there is special care should be taken to choose the type of the structural elements were utilised to represent the plate's behaviour. A structural test of the un-damped free plate vibration is presented by different types of the solid elements. In both 166k mesh cases, the solid oscillation consists of a rotation and lateral displacement.

4.3 2D Model of Hanging Membrane Roof Subjected to the Wind

The structure is modeled as an elastic body. This model example is a model of an introductory study of a 3D membrane roof subjected to wind loading presented in Section 4.4. The main purposes are to study the coupling of fluid-structure interaction problems and demonstrate different applications of the solution techniques [44]. The membrane is modelled and solved by the FSI solver in foam-extend.

4.3.1 Geometry, Material Properties, and Boundary Conditions

The model geometry and its material properties are presented in Figure 4.11. According to Hübner [193], the exponential law applies to the velocity profile at the inflow edge reads

$$v_x(y) = 35 \text{ m/s} \cdot (y/350)^{0.22} . \quad (4.1)$$

Where the flow velocity in the roof height is given by the value $v_x(0.2) = 6.77 \text{ m/s}$ (see Figure 4.12). Thus, the Reynolds number for this model equals 270.8. The maximum fluid flow inlet velocity \hat{u}_{\max} is linearly increased to reach about 0.6 m/s in 2s of the simulation time as will be shown later in the simulation results in Figure 4.15 The simulation has been done for 20 time-steps with $\Delta t = 0.1 \text{ s}$.

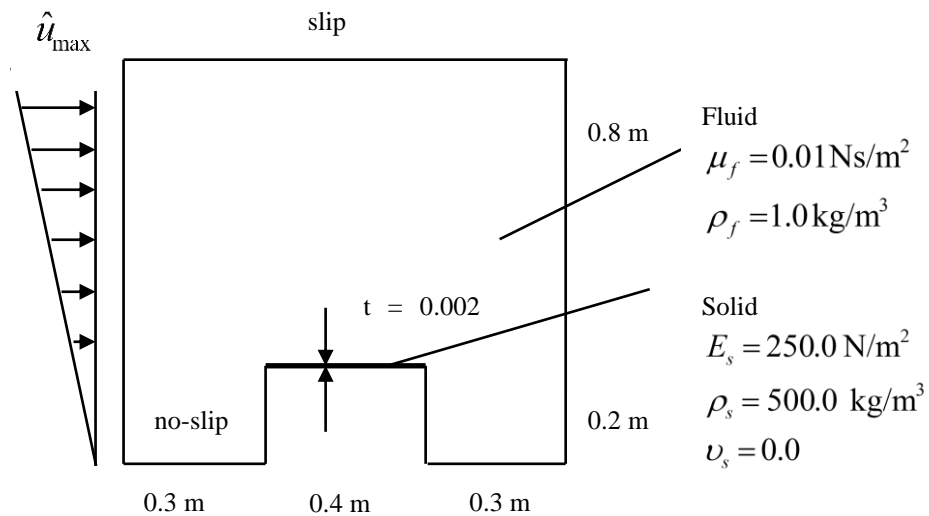


Figure 4.11: 2D membrane roof model: geometry, material properties, and boundary conditions

4.3.2 Mesh Generation

The mesh is generated by the blockMesh utility in foam-extend 3.1 version, and both coarse and fine meshes are performed to compare the deformation results. The finer mesh for the fluid region is generated by dividing the coarse mesh elements into four as the meshing results shown in Figures 4.13 (a) and (b) are also summarised in Table 4.7.

Table 4.7: 2D hanging roof meshing details

Mesh	No. of nodes	No. of elements
Coarse	24,522	12,000
Fine	97,242	48,000

4.3.3 Post-Processing, Results and Discussion

This case model is solved using block Gauss-Sidel iterations with Aitken relaxation using fsiFoam solver. The deformed shape of the model example after 2s of the simulation time is shown in Figure 4.14. At membrane roof corners for both grids, the displacement represents steep gradient and the deformation coincides with the decrease in pressure in the fluid flow area. Additionally, the two grids also provide the same behaviour of the deformation of the building. In other words, the coarse grid has a good converged indicator to the fine mesh solution, and for more clarification Figures 4.16 and 4.17 are respectively presented to show the deformation for both grids in x and y-directions.

The steep way how the membrane bended into the building region, seems to be related to how the case was defined in 2D. For example, in a 3D case, the fluid would have more directions to spread out and around the building; whereas in 2D, the fluid is forced to accelerate and increase pressure to account for the building (obstacle) which is reducing the total volume within the domain, which results in a high deformation of the roof membrane when said fluid could expand into the inside of the building. From Figure 4.16 it is possible to see more clearly how this 2D structure affects the passage of the fluid

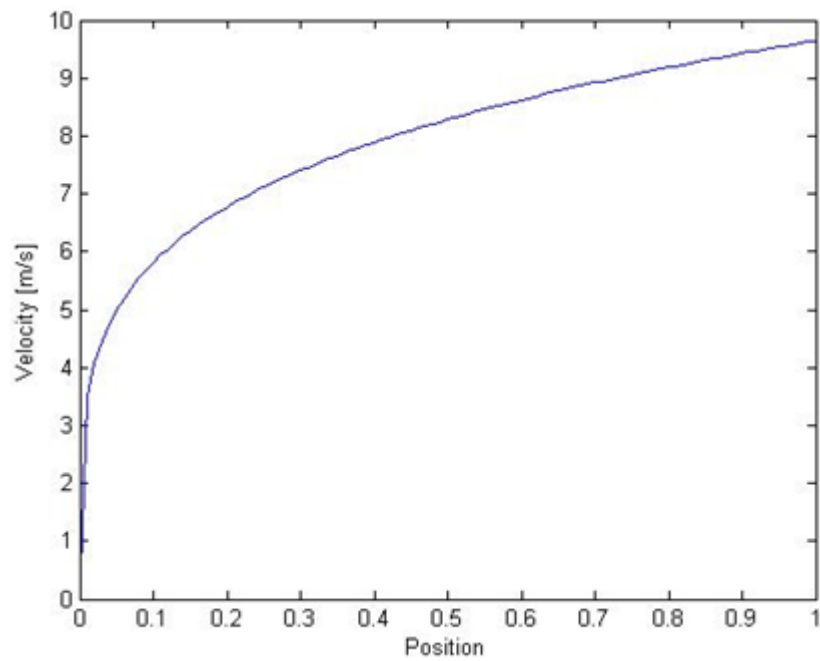


Figure 4.12: Flow velocity in the 2D membrane roof

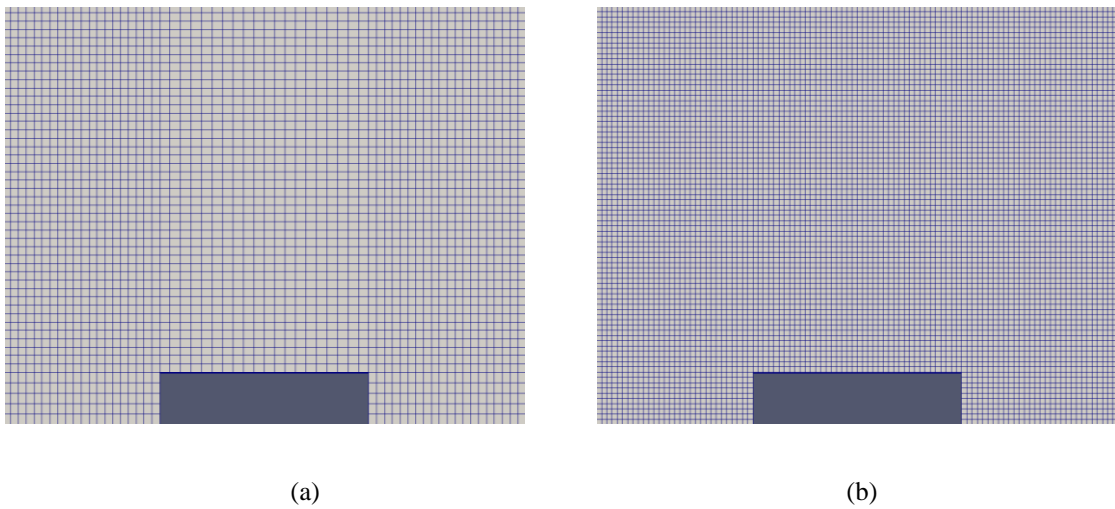


Figure 4.13: Meshing details: (a) coarse mesh (b) fine mesh

flow over the building, given that the centre of the membrane is actually moving forward due to the impact of the fluid when going over the building and down again into the membrane and inside the building.

Figure 4.15 shows the coupling interface deformation at the top edge of the membrane roof at the end of the simulation time. The calculation time for the one time-step is 1.17s for the coarse mesh and 3.94s for the fine mesh.

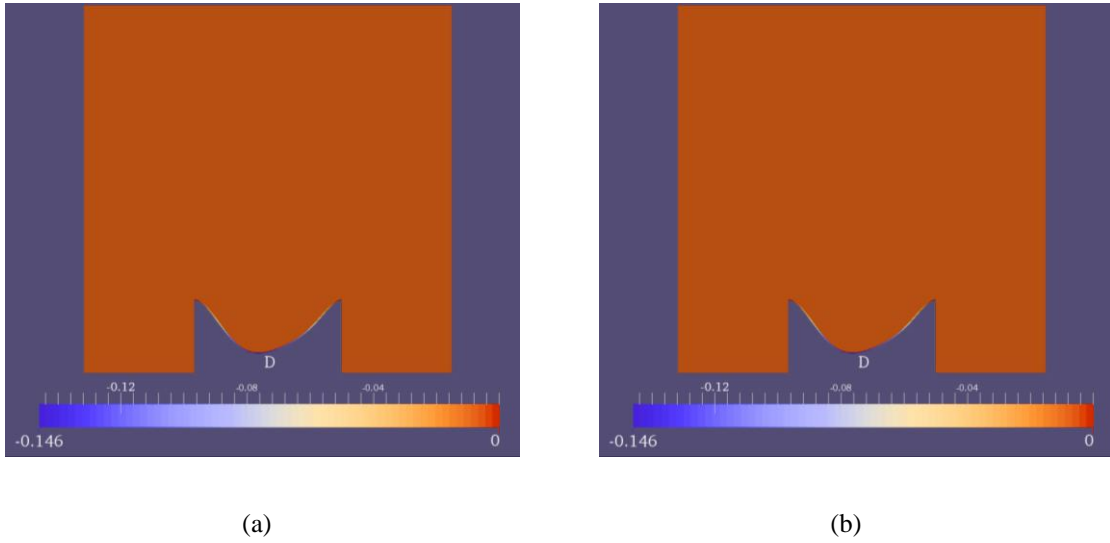


Figure 4.14: Solution for the displacement at the interface: (a) coarse (b) fine

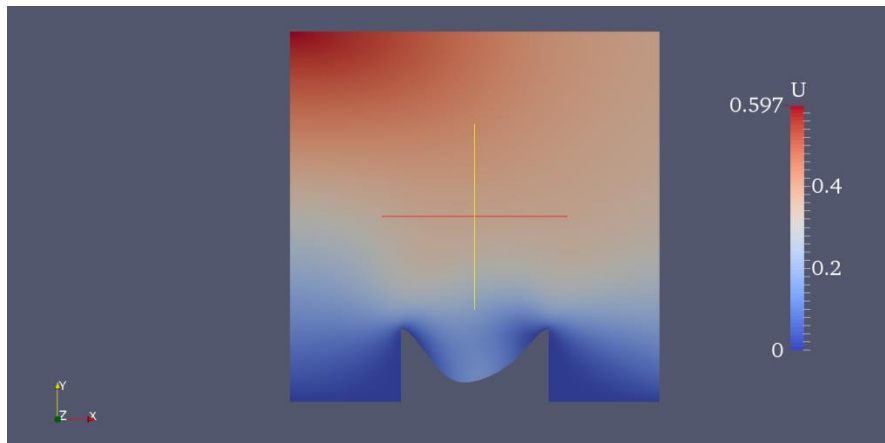


Figure 4.15: The maximum inflow speed for the 2D membrane roof model

4.4 3D Membrane Roof Benchmark

In the concluding section of this chapter, the three-dimensional membrane roof characteristics will be studied. The 3D flexible membrane roof example is an extension of the 2D model discussed in Section 4.2. The flexible structure is surrounded by the fluid and therefore a relatively large fluid area is required in the simulation in order to detect all the effects of the flow in the environment.

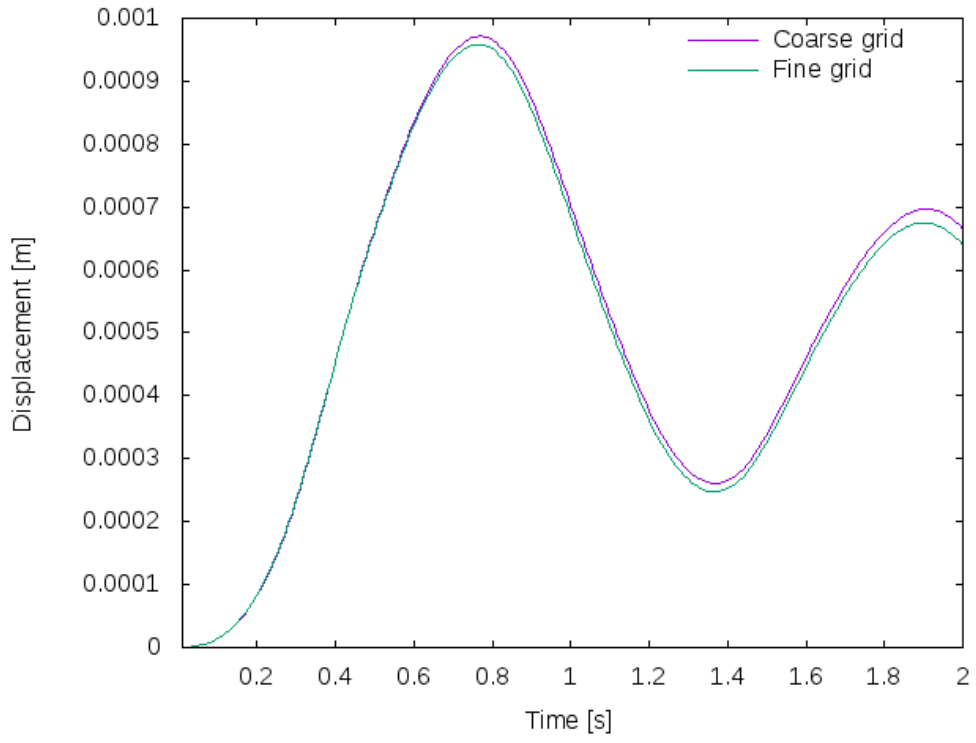


Figure 4.16: Membrane roof deformation in x-direction

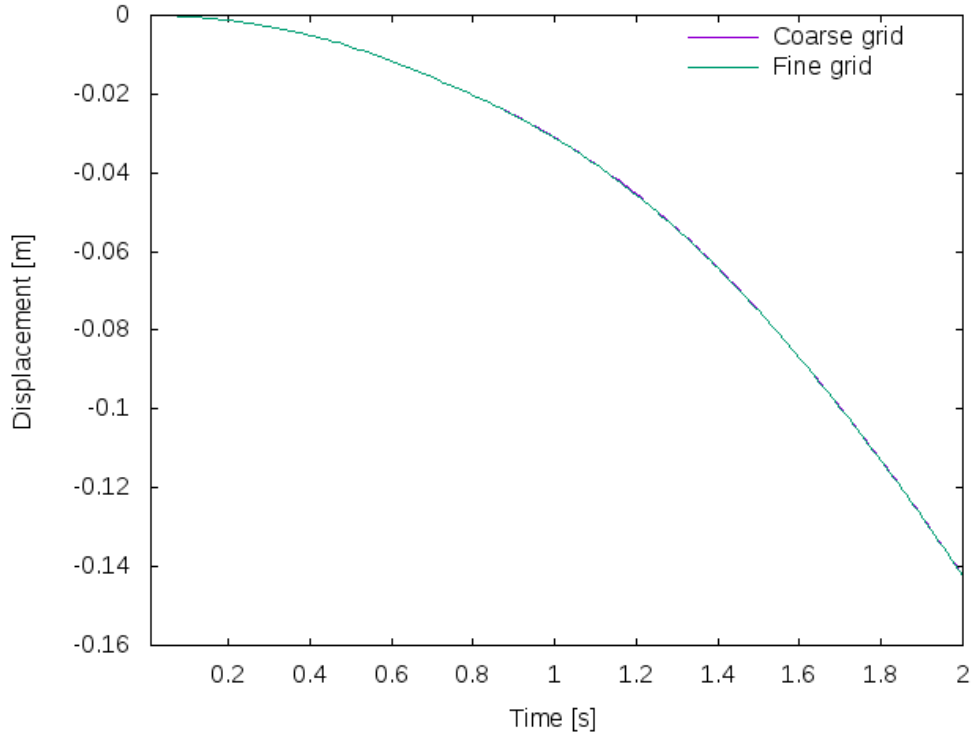


Figure 4.17: Membrane roof deformation in y-direction

4.4.1 Model Geometry

Figure 4.18 shows the geometrical properties of both the fluid and the structure. The fluid dimensional domain is $150\text{m} \times 100\text{m} \times 75\text{m}$ and is $10\text{m} \times 10\text{m} \times 5\text{m}$ for the rectangular membrane roof.

4.4.2 Material Properties

In this case, the Newtonian fluid flow and the St. Venant-Kirchhoff structure are modeled. The material parameters are represented in the table below. The Reynolds number given in this case is larger than the critical Reynolds number for the laminar flow.

4.4.3 Mesh Generation

Following the same strategy of generating a mesh for 3D flexible plate attached to a solid support in Section 4.3 by dividing the domain into levels, the 3D flexible membrane roof mesh is generated and finally, the meshing result is illustrated in Figure 4.19.

Table 4.8: Material properties of the 3D membrane roof model

Domain	Parameter	Value	Unit
Fluid	ρ_f	1.25	kg m^{-3}
	ν_f	0.08	$\text{m}^2 \text{s}^{-1}$
Solid	ρ_s	1000	kg m^{-3}
	ν_s	0.0	
	E_s	1.0×10^9	Pa
Flow	Re	8,900	-

4.4.4 Discretisation, Boundary Conditions, and Simulation Results

For the spatial discretisation, the meshing results show that the fluid domain consists of 36,858 nodes and 33,450 hexahedral elements and the solid domain contains 6,448 nodes and 5,400 cells. The ALE formulation is applied to the area above the flexible membrane boundary.

The membrane roof is extremely slender due to its thickness is 0.01 m which gives the ratio of length/thickness equals 1000. Therefore, for temporal discretisation, $\Delta t = 0.01s$ is selected for 500 time-steps.

The boundary conditions that used in this case model and shown in Figure 4.18 above and taken from Hübner's study [193]. The inflow velocity in x-direction is defined on the boundary where $x = 0$ as

$$\hat{u}_x(z,t) = 100.0 \text{ m/s} \cdot \hat{u}_t(t) \cdot \hat{u}_z(z). \quad (4.2)$$

Thus, the inflow velocity in z-direction is changing exponentially and sinusoidally until $t = 5s$ as shown in the two equations below and their plots are presented in Figure 4.21a and b

$$\hat{u}_z(z) = \left(\frac{z}{350} \right)^{0.22} \quad (4.3)$$

$$\hat{u}_t(t) = \left(\sin \left(\pi \left(\frac{t}{5.0} - 0.5 \right) \right) + 1 \right) \cdot (0.5) \quad (4.4)$$

Therefore, at $z = 75$ m and after $t = 5s$, the maximum inflow velocity is 71.2 m/s as presented in Figure 4.20 and this corresponds to $Re = 8900$ as mentioned in Table 4.8.

For the coupled problem, there are ten iterations in each time-step, and the Aitken scheme is applied for convergence. About 20h wall-clock times for one core are required for the 500 time-steps.

The coupled FSI simulation represents very small vortices shedding at upper membrane roof boundary. These vortices will move downstream through the membrane and will cause the pressure vibration on the membrane. In addition, this leads to the 3D membrane

roof oscillation. However, the large vortices shedding occurs in the membrane roof wake area.

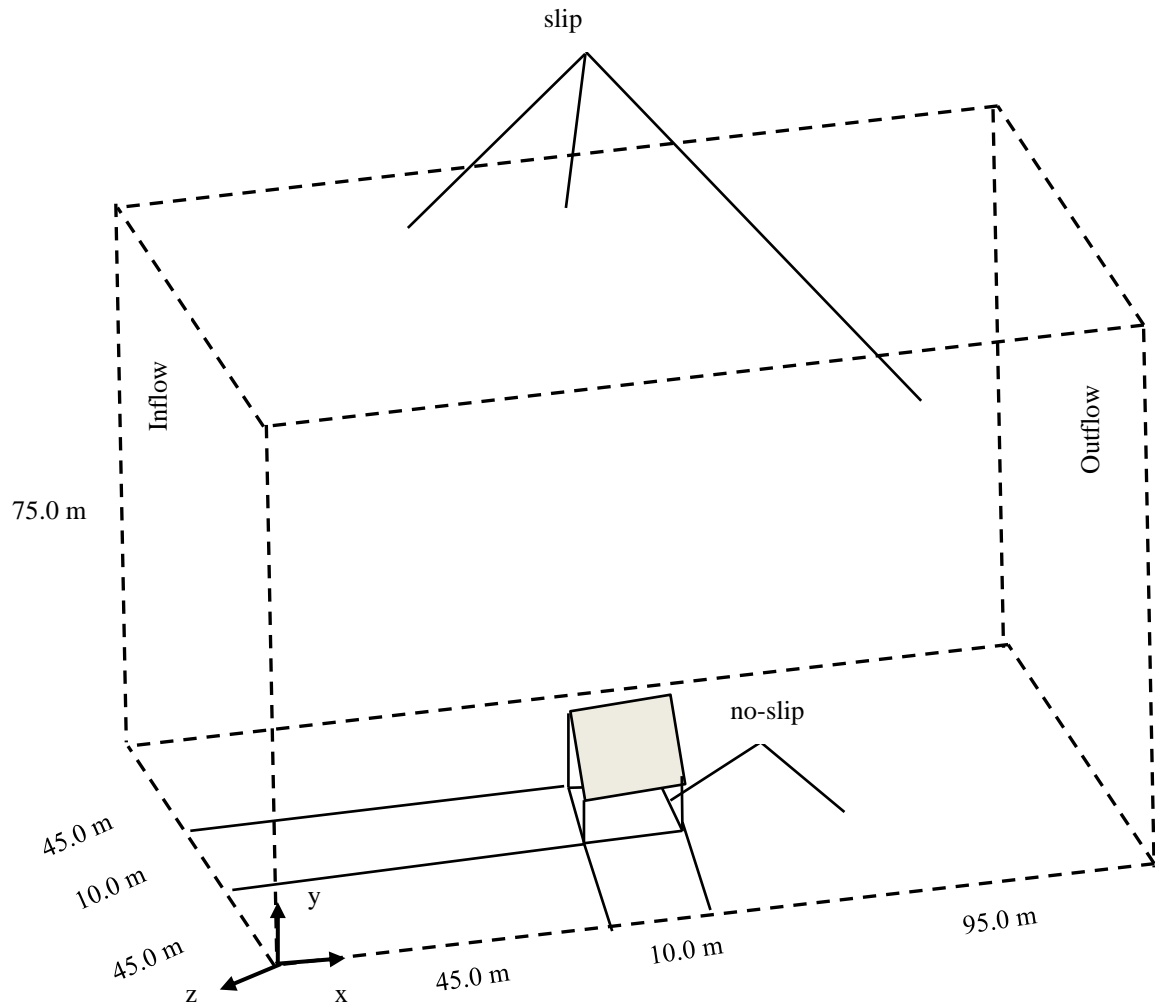


Figure 4.18: Geometrical properties and boundary conditions of 3D membrane roof

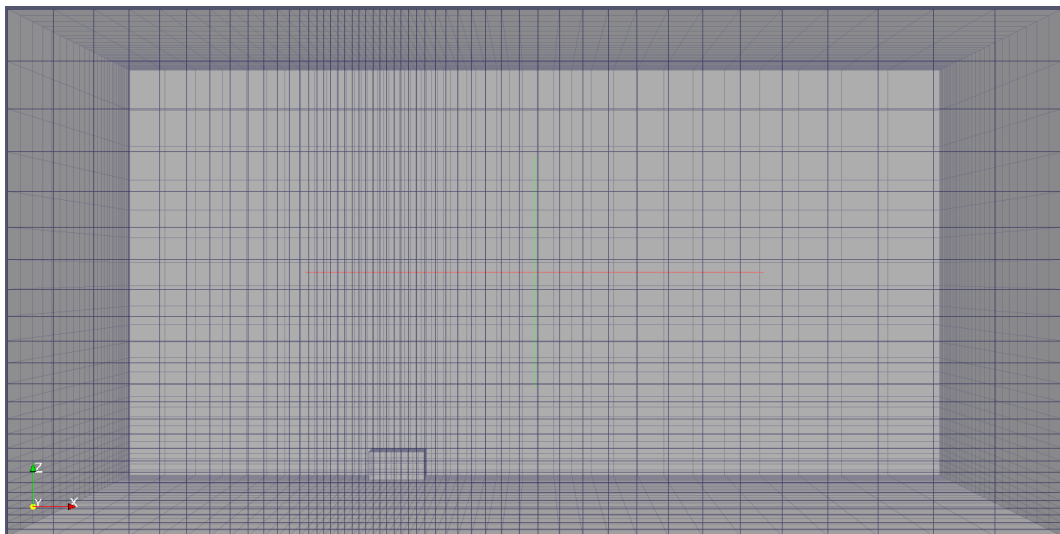


Figure 4.19: Meshing results for 3D membrane roof model

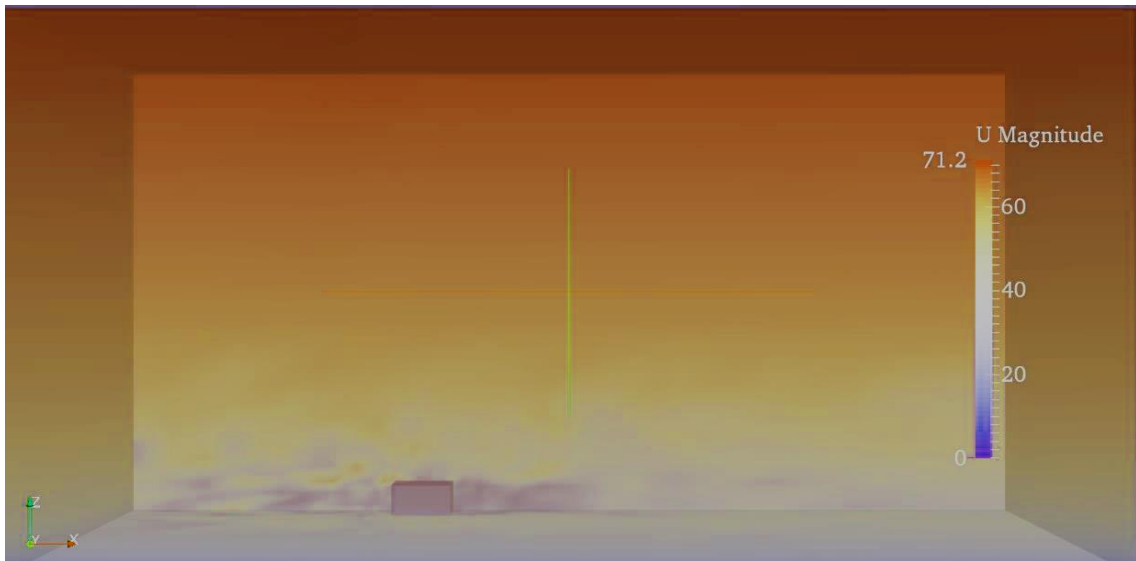


Figure 4.20: Maximum flow velocity for 3D membrane roof benchmark

Figure 4.22 represents the irregularity of the 3D oscillation of the flexible membrane roof in the z-direction. This oscillation is very different from the related 2D simulation. It was noted in the results section for that simulation that the absence of space around the building was the critical reason as to why the membrane was deflecting so much in the 2D case. This 3D case and the results from Figure 4.22 are a clear indicator of how much the deflection profile is different, namely the maximum displacement was of around 0.15m on the 2D case, and it is smaller than 0.09m in the 3D case, with the uniform periodic region after 3s to be below 0.05m, making it roughly 3 times smaller than the 2D case.

Associated with the smaller displacement, oscillations can more easily occur in this 3D case, given that the fluid can more easily go over and around the building, along with generating and releasing vortices, which did not occur in the 2D case.

The snapshots shown in Figure 4.23 should be repeated through the membrane after some period. Von Scheven and Ramm [44] showed that the snaps repetition occurs after one period of simulation time.

However, this 3D case was simulated by taking advantage of the 3D symmetry of the domain and geometry, which allowed to reduce the total wall clock time to half the time

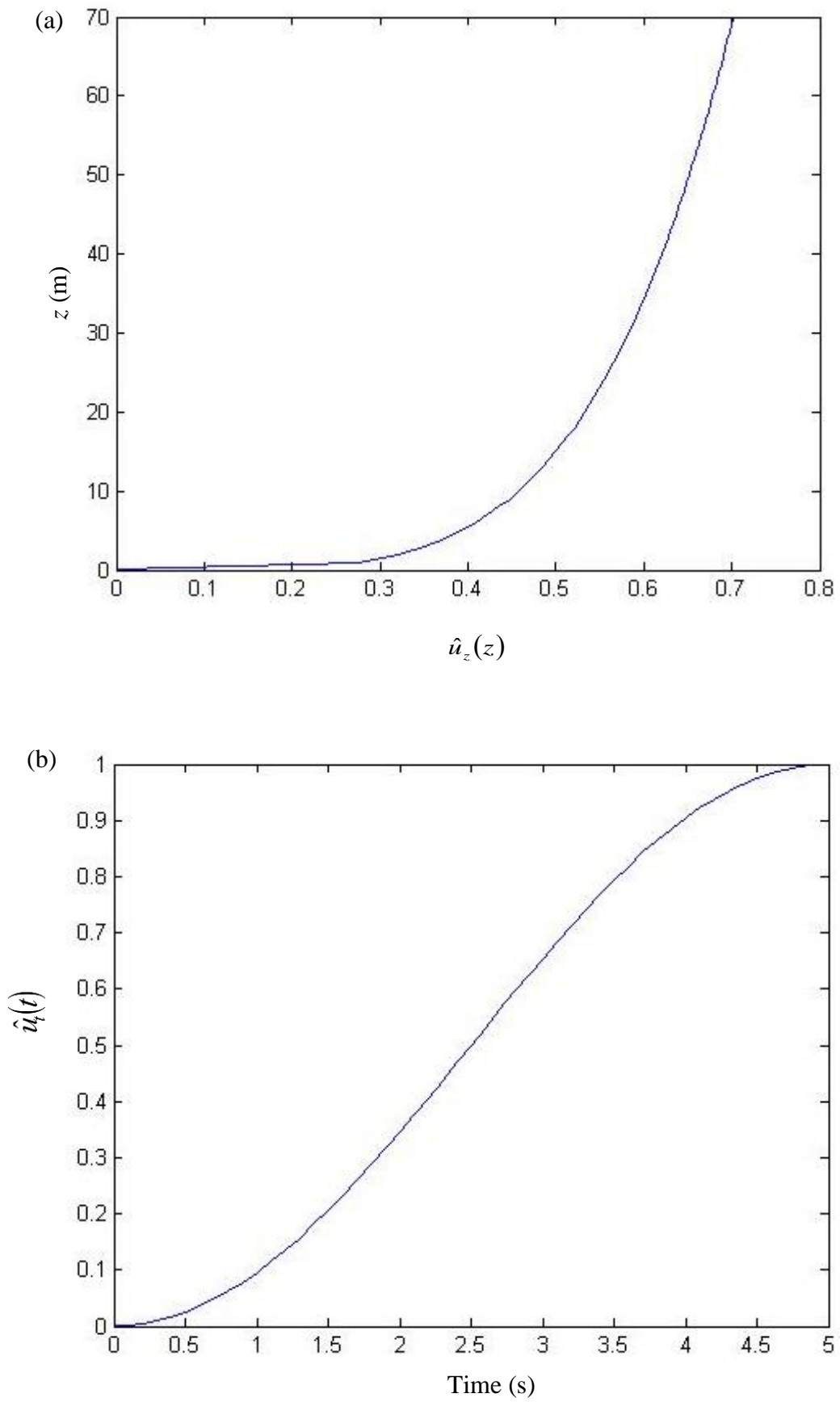


Figure 4.21: 3D membrane roof: (a) spatial (b) temporal inflow boundary conditions variations

it would be necessary otherwise. But there is some concern that this 3D symmetry modelling strategy should perhaps not have been used, given that the results for the 3D plate in the reference articles [44,191] gave an indication that there should be some torsion applied on the plate and that symmetry would have been broken. In other words, the concern here is if fluid flow speed versus the building and membrane sizes, could have led to a significant non-symmetrical pattern to the vortex generation and shedding, and consequently changed the displacement pattern on the membrane considerably.

But then again, associated to this, the same modelling limitations which were diagnosed for the previous cases would likely also need to be accounted in this case as well, although this was not noticed in this case as significantly.

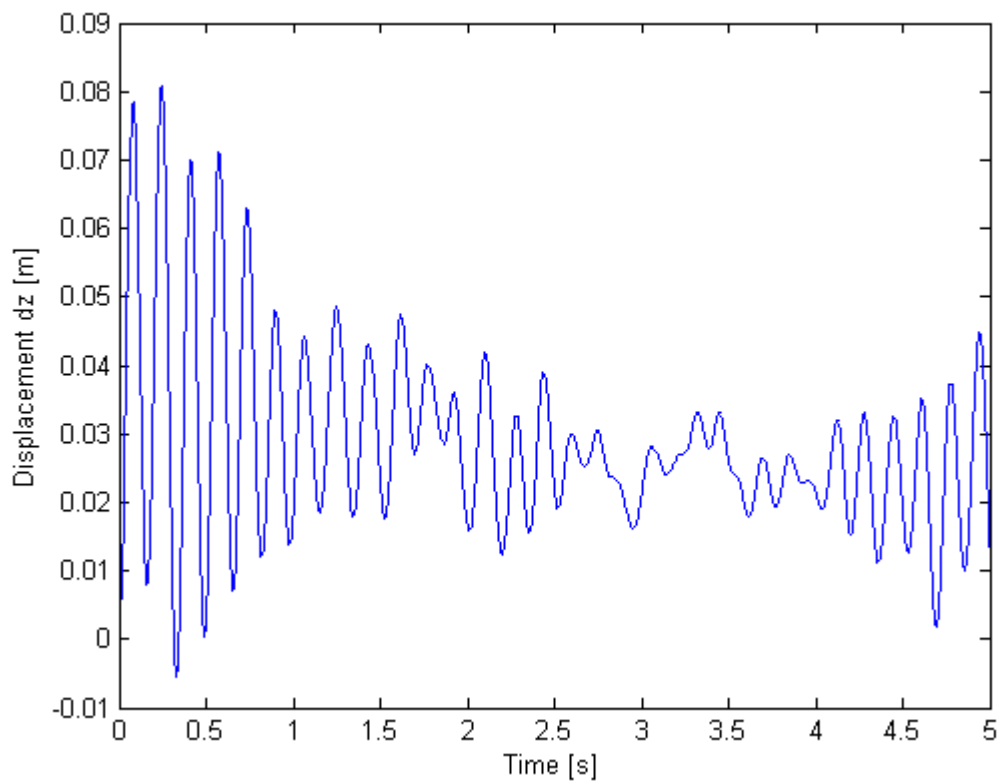


Figure 4.22: 3D flexible membrane roof deformation over z-direction

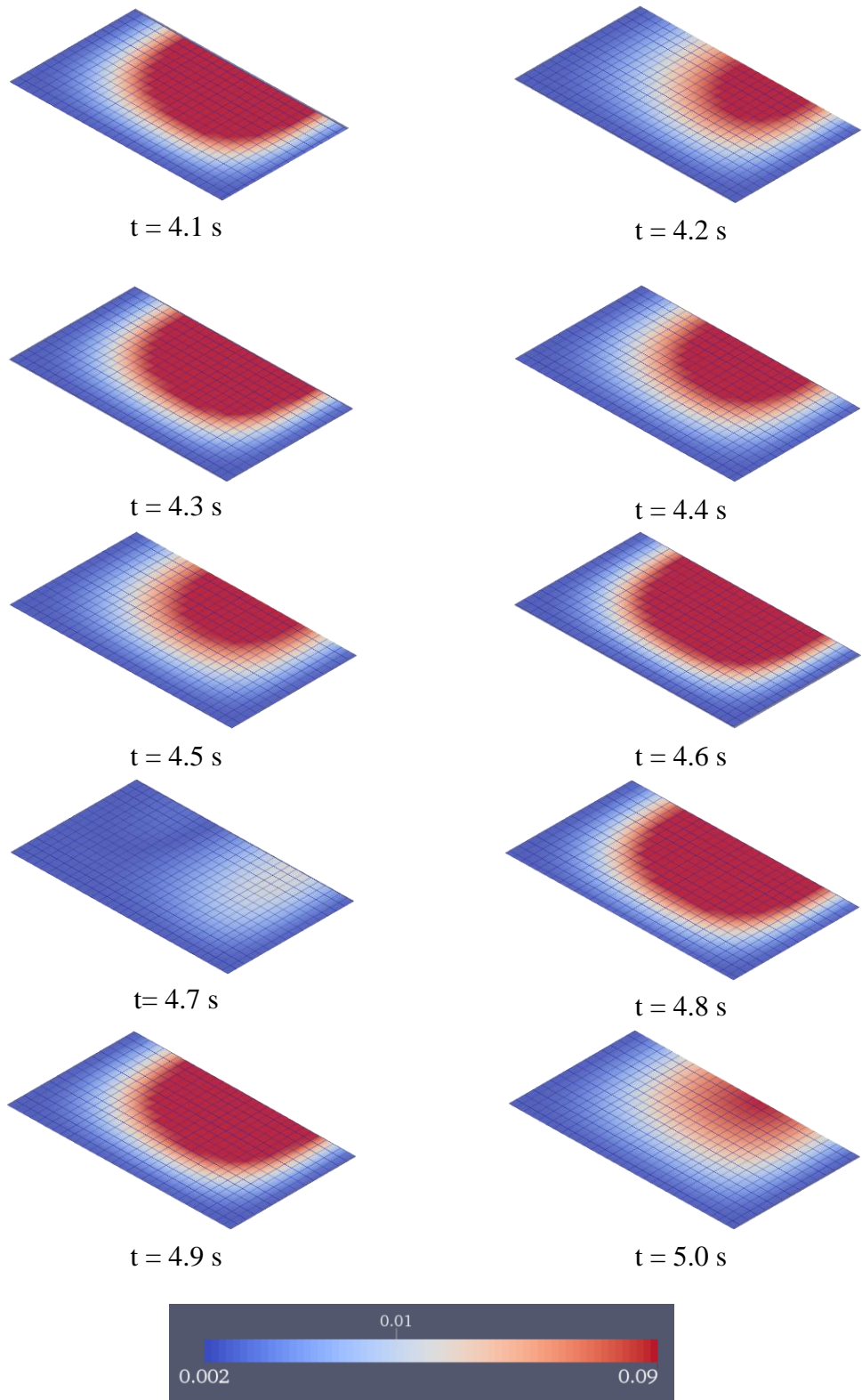


Figure 4.23: Deformed structure of the 3D membrane roof case in z-direction

4.5 Chapter Summary

Four benchmarking of two-dimensional and three-dimensional fluid-structure interaction has been studied in this chapter. Firstly, the 2D elastic tail deformation is induced by Von Karman vortex shedding from a square block. The results present a good agreement with other literature. In addition, the analysis of the fluid flow represents that the interaction between the fluid and the structure is done in both x- and y-directions. The periodic tail deformation is excited by the unsteady vortices, which in turn interacts with the new vortices that are generating from the neighbouring fluid at the tip of the flexible tail, and then the Von Karman vortex shedding street is distributed from the path.

The second testing case model is a hanging roof in the 2D space. This case is simulated for the wind flow effects on the slenderness structures. Taking into account that all needs arising from the physical model, the fsiFoam solver is applied to both fluid flow and solid.

Then, in other two benchmarking, the 3D numerical models of the thin-walled structures exposed to the aerodynamic flows. The first 3D model is the elastic cantilever plate attached to a solid support and the second is the 3D case is an extension of the 2D hanging membrane roof problem have been investigated.

Review of the past conducted simulations, all of the studied cases share a common trait: oscillatory motion imposed by fluid flow on various shapes and sizes of solids, due to the vortices which were generated and released in the wake of the flow, as the fluid interacted with the solid and vice-versa.

However, there were some situations where the modelling limitations revealed that even if the flow pattern is considered a laminar flow (due to the small Reynolds number and/or due to the clear shaped vortices), turbulence modelling or fairly more refined meshes should be used in order to properly capture the generation and release of vortices. This could be seen in the 2D membrane roof top case and in the 3D stationary cylinder with a fixed plate on one end. As proof, this was revealed by the 2D flexible tail behind a block, when comparing the results of a coarse with a fine mesh, as well as in the VIV cases, where the mesh was always fine enough to represent the vortices on all simulations.

Another trait which was revealed is the need to carefully choose the modelling strategies which are implemented when performing simulations with fluid-solid interaction, when the solid can be deformed by the forces imposed by the fluid. More specifically:

- When modelling with the 2D and 3D with a symmetry plane or when using a complete 3D simulation, revealed that:
 - A 2D mesh/geometry should only be used for studying the main behaviour of a solid, if and only if the influence of the flow in the missing 3rd dimension is not relevant enough to the study, as witnessed with the 2D and 3D representations of the flexible cantilever plate attached to a solid support.
 - A full 3D simulation does not necessarily imply that it will always reveal non-symmetrical flow profiles, as also revealed in the elastic cantilever plate attached to the solid block, even though it should possibly have revealed that torsion should have occurred.
- Using a more refined mesh can be fairly prohibitive, especially when the solver cannot be used with parallelized processing, given how time consuming these types of simulations can be.
- Simulations using the coarser meshes which were used, should possibly have been performed with a turbulence model, instead of using laminar flow modelling. However, that was beyond the scope of this thesis.

Chapter 5

Wind-Membrane Interaction

This chapter presents a systematic investigation and realization of numerical simulation of fluid-structure interaction phenomena in the membrane structures case. A typical application is the four-point tent structure subjected to wind flow. Due to their behaviour of special load carrying, membranes are extremely light-weight and susceptible to flow induced effects. Thus, they are characterised by their flexibility and light-weight which makes them more sensitive to the wind. In this chapter, the flow-induced membrane deformation in a smooth wind flow is studied.

5.1 Numerical Example of Four-Point Tent Structure Subjected to the Wind

In the following numerical model, a four-point tent structure is studied under the wind loading as a common model in Oman. The tent structure example resembles a hyperbolic surface shape of the 10m × 10m membrane with 2.5 kN/m pre-stress. It is stabilised by four cables with a pre-stress of 50kN and supported by two masts with a thickness of 6 mm and a diameter of 88.9 mm. The material of the membrane is a PVC coated-polyester fabric of type I and its thickness is 1mm. The bracing contains two couples of cables which have 13.8 mm diameter and 41 kN pre-stressed force [187]. The case model with its dimensions is presented in Figure 5.1.

5.1.1 Meshing Set-Up

Planned on testing with the snappyHexMesh utility in OpenFOAM [158], but this has ended with some issues when generating meshes around thin curved surfaces. The known problems are as follows:

- Since the membrane is a thin surface, the meshing method can get a bit lost on where the surface starts and ends, because there is no sharp corner, only a sharp edge.
- Given the curvature of the tent surface; there would have been a lot of cells getting either sliced or getting severely deformed in trying to snap to the surface of the tent.

Along with these issues, the mesh is meant to be subjected to dynamic mesh motion, so that the tent surface can change its surface with the forces of the fluid around it. This meant that the majority of meshes done with snappyHexMesh would eventually result in some critical issues, such as a cell being too skewed or squished to be properly morphed, which would numerically corrupt the simulated flow fields.

Another mesh technique, cfMesh [203] was also a possibility for meshing, but it can suffer from the same issues as snappyHexMesh.

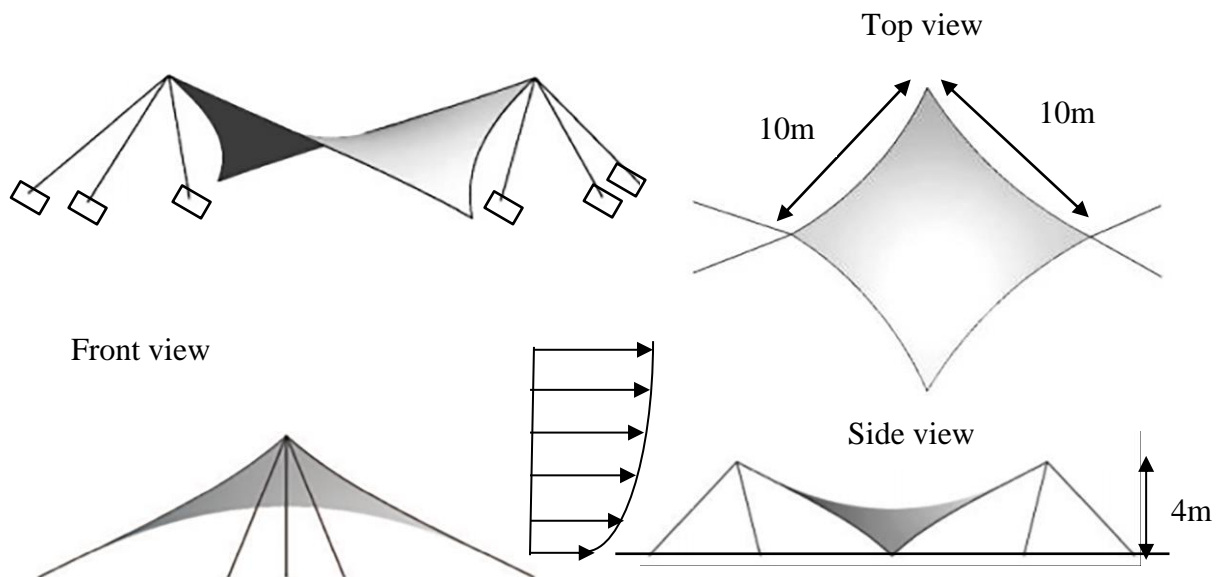


Figure 5.1: Geometry and dimension of the four-point tent structure

Therefore, the solution chosen was to design the mesh using other less conventional tools that OpenFOAM has got, namely:

- Using `blockMesh` to design a simple mesh that could be easily deformed, while also allow for it to adjust properly to the tent surface.
- Using `moveDynamicMesh` to morph the previous mesh onto the tent surface.
- Using any other tools that are necessary for finishing up the mesh, such as `mergeMeshes`, `stitchMesh`, `refineMesh` and anything else that was needed to get the job done.

The case structure is illustrated in Appendix 5.A and meshing steps for fluid and solid regions are shown in Appendices 5.B and 5.C, respectively, will outline what has been done and how it is currently working.

5.1.2 Important Logic Notes

There are a few important logic issues with this mesh set-up:

- The fluid region does not account for the solid thickness.
- This is because the fluid region was originally planned to only with the top of the membrane; otherwise, it would be had to use even more insane methods to manipulate the mesh.
- It was expected to work this way due to a previous case in Section 4.4 where only the top of the membrane meshed in the fluid region.
- Although the solid thickness is set to 1cm at the start of the solid mesh generation, specifically along the z-direction, which means that along the normal of the tent, it is likely only 0.707 of that, namely $\cos(45^\circ) = \sqrt{2}/2$.
- Only the membrane itself is modeled, while masts and cables influences are neglected.

However, limitations on this meshing strategy were revealed when attempting to use the solver with fluid-solid interaction turned on, which leads to the need to only mesh the fluid region on top of the membrane, without any fluid passing through the membrane.

In addition to these limitations, it was only possible to fixate the 4 edges of the tent membrane; it was not possible to fixate just the tips of the tent due to meshing limitations and how boundary conditions are defined in OpenFOAM, specifically due to only allowing the fixation of faces on the boundary mesh.

5.1.3 Simulation Set-Up

In this example, the transient analysis of a wind flow loaded tent is performed. The fluid and membrane properties are presented in Table 5.1. The strongly coupled simulation of a completely 3D set-up is accomplished using a hexahedral mesh for both fluid and solid. Prism layers were originally used for refinement mesh at the bottom of the membrane, but it was not possible to use the solver with those cells, due to a limitation on how the cells were stitched together. More specifically:

- Since the mesh had to be generated in 3 parts (2 fluid and 1 solid), where each part was meshed refined mostly independently, would result in not having the cell order match properly between the two fluid mesh regions.
- It was not possible to solve this with any of OpenFOAM/foam-extend meshing utilities, nor was it possible to use cyclic boundary conditions to artificially connect the two fluid regions (top mesh with the bottom mesh), therefore it was necessary to one of the mesh parts, which lead to discarding the bottom mesh.
- Furthermore, the only way to have a properly working solid region mesh, was to extrude the respective membrane top patch on the top fluid part onto the solid mesh part, so that the cells were properly ordered and matching; this was the other problem with meshing the bottom fluid part, given that it was necessary to extrude this mesh from the membrane bottom patch on the solid region.

The chosen fluid is air at 25°C, which enters the simulated region with a uniformly distributed wind flow along space and oscillating with time (see Figure 5.2), with its main flow direction is normal to inlet plane on the left side of the domain (bottom-left perspective in Figure 5.1).

Table 5.1: Material properties of the 3D four-point tent case

Domain	Parameter	Value	Unit
Fluid	ρ_f	1.25	kg m ⁻³
	v_f	0.08	m ² s ⁻¹
Solid	ρ_s	1,300	kg m ⁻³
	ν_s	0.4	-
	E_s	4.0 x 10 ⁹	Pa
Flow	Re	3,750	-

Besides examining the partitioned FSI approach, the aim of the tent structure simulation is the assessment in a qualitative way of the occurring influences and their magnitude, since the complete mesh could not be created and therefore, a direct comparison with the reference articles could not be done.

The dynamic behaviour of the model was studied in an unsteady FSI simulation for a wind flow in the y -direction. Regarding the fluid incompressibility and low mass of the structure, which in sequence because of low wind speed, the coupled simulation was studied in an implicit way. The stabilisation in the FSI partitioned approach between the incompressible fluid flow and light-weight structures is already discussed in Chapter 2 and demonstrated in Chapter 4. Moreover, the Aitken based under-relaxation method is applied to enhance the convergence.

Figure 5.2 behaviour shows that the deformed membrane is starting from the steady-state solution at 20 m/s, and the maximum wind-speed is changed within the range [10 m/s, 30 m/s]. The deformation of the membrane structure followed the wind-speed variation because of the little membrane mass and its pre-stress [187,204].

Given the limitations of the mesher and solver, the fluid could only go over and around the tent membrane. Therefore, the vortices developing on top of the membrane develop

downward and upward forces as times pass, as shown in Figures 5.3 and 5.4.

The simulation has taken roughly 6h of the CPU time, around 1h for each simulated second. The steps need for the simulation with fluid-solid interaction is all illustrated in the script file Allrun (Appendix 5.D).

These results are fairly differenced from the reference papers [187,204], given that it was not possible to mesh the fluid region below the tent membrane. This resulted in this simulation only handling the interaction between the fluid travelling on top and around the tent membrane, which associated to the 4 sides of the membrane being fixated, resulted in only occurring upward and downward forces on the centre of the membrane. In addition, the thickness of the membrane could not be specifically the 1mm thickness, due to the limitations with fluid-solid interaction solver that was used.

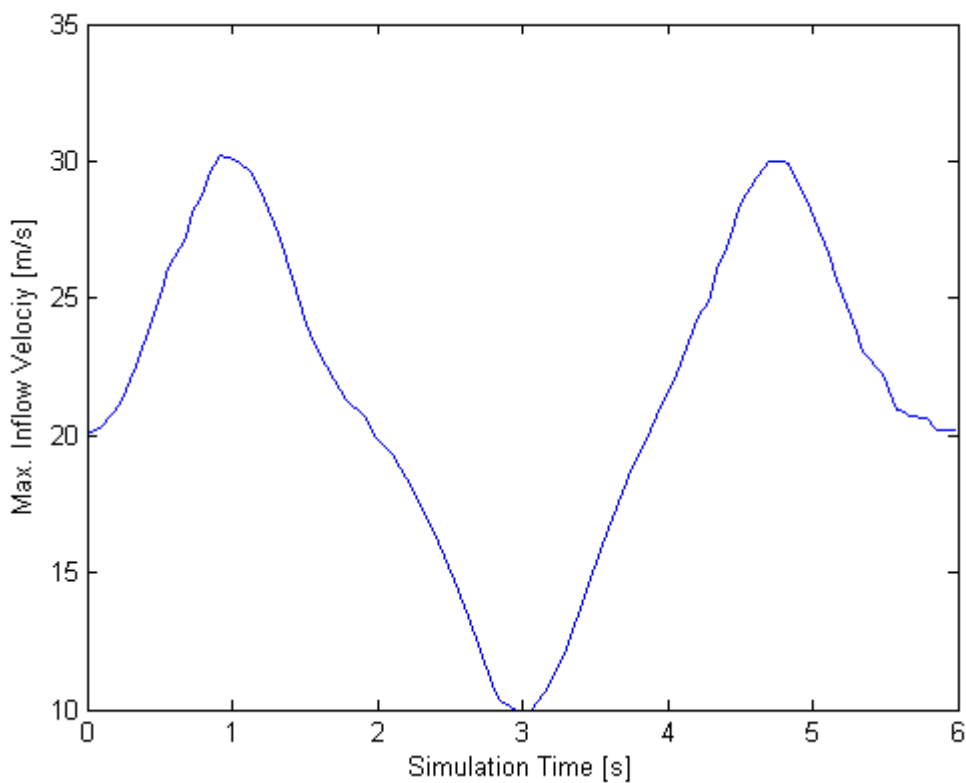


Figure 5.2: Maximum inflow velocity over simulation time

5.1.4 Tent Case Limitations

This case was simulated with the `fsiFoam` solver. Given that the fluid is a lot more viscous than air, then in order to get a distortion similar to using air, it would have been necessary to reduce the stiffness of the solid properties, so that the tent membrane bends a bit more than just roughly 2 millimeters or so. As already mentioned in the previous section, the main problem with the case not running had to do with how the solid mesh was being created. The current case is only working because of the following details:

- "membraneTop" is extruded from the fluid region onto the solid region;
- The fluid region below the membrane is removed;
- The four sides of the membrane are made to be fixed in their place;
- The wind profile is at the inlet, and the fluid domain is increased in all 5 directions: North, South, East, West, and Top. This was specifically done so that it would be similar to the previous 3D membrane case (Section 4.4) and would allow for the alleviation of the flow around the tent and the pressure fields could be better distributed around the tent.
- Not extending the fluid domain on all 5 directions would result in a flow profile similar to having the tent tightly stored inside a pipe, which would consequently increase drastically the fluid forces on top of the tent membrane; for example, the membrane would be severely distorted as it happened with the previous 2D membrane case in Section 4.2.

Otherwise, it was not possible to create a mesh in time for this to work as intended. This is due to there still being several limitations with the current technology, namely how the existing tools allowed the meshing to be done and how the `fsiFoam` solver works.

Having the fluid only going over the tent is not as accurate, but at least it can give comparable results to the previous 3D membrane. In other words, given that it was not possible to create a case identical to the reference articles [187,204], then it was attempted to compare this case to the previously studied 3D membrane; this way, the fluid will flow only over the top side of the membranes.

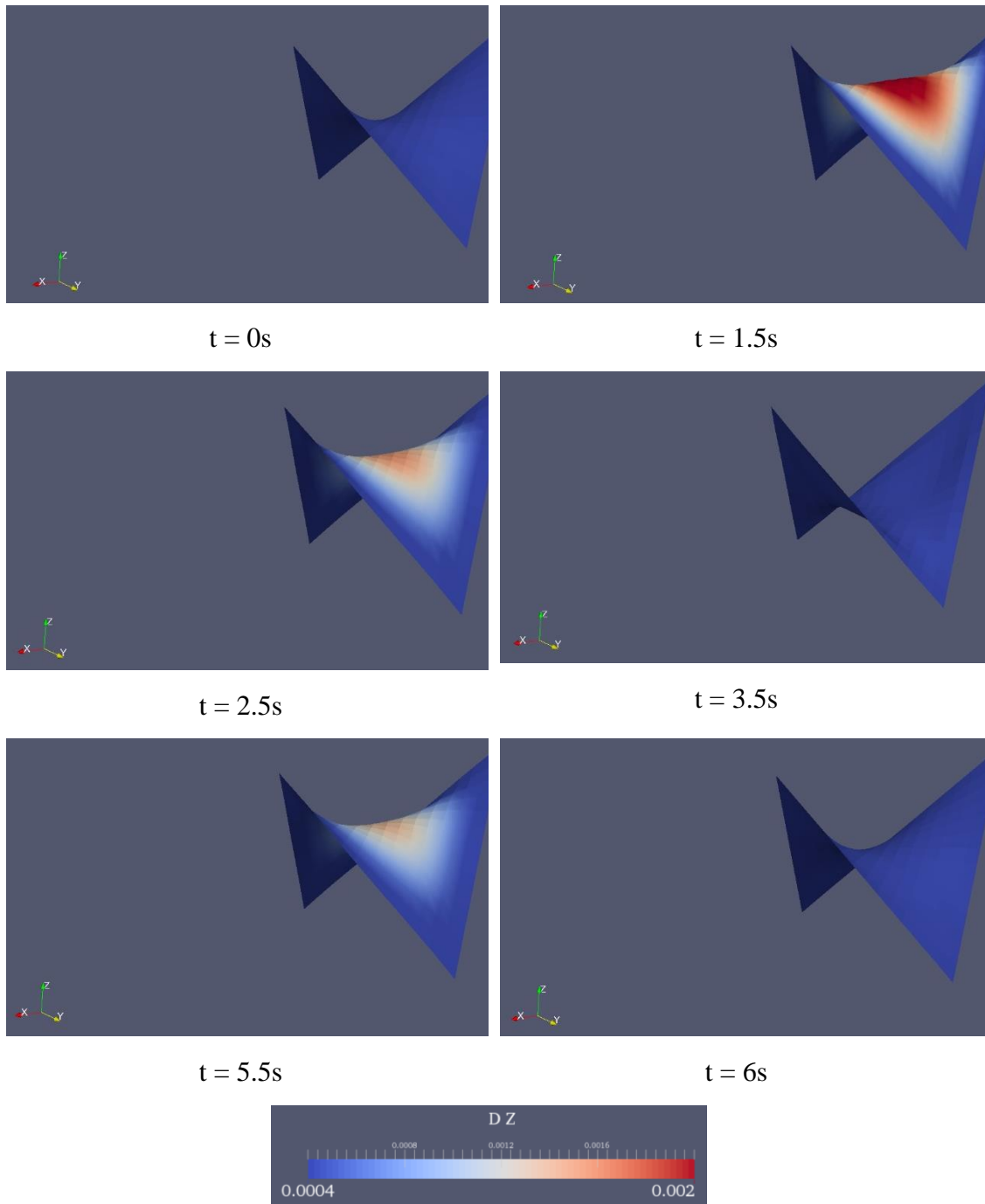


Figure 5.3: Response of the structure at different times with counter plot of the displacement dz in meter

It is very likely that the only way to easily generate a mesh for this case would be to use commercial meshers, such as Pointwise's mesh generators, which are fairly expensive for commercial use and it was not confirmed what the prices for academic use are. Furthermore, such a decision would have had to be taken a few months sooner than when work started on this tent membrane model, in order to get access and to get properly

familiar with those commercial mesh generators, for creating the final meshes for this model.

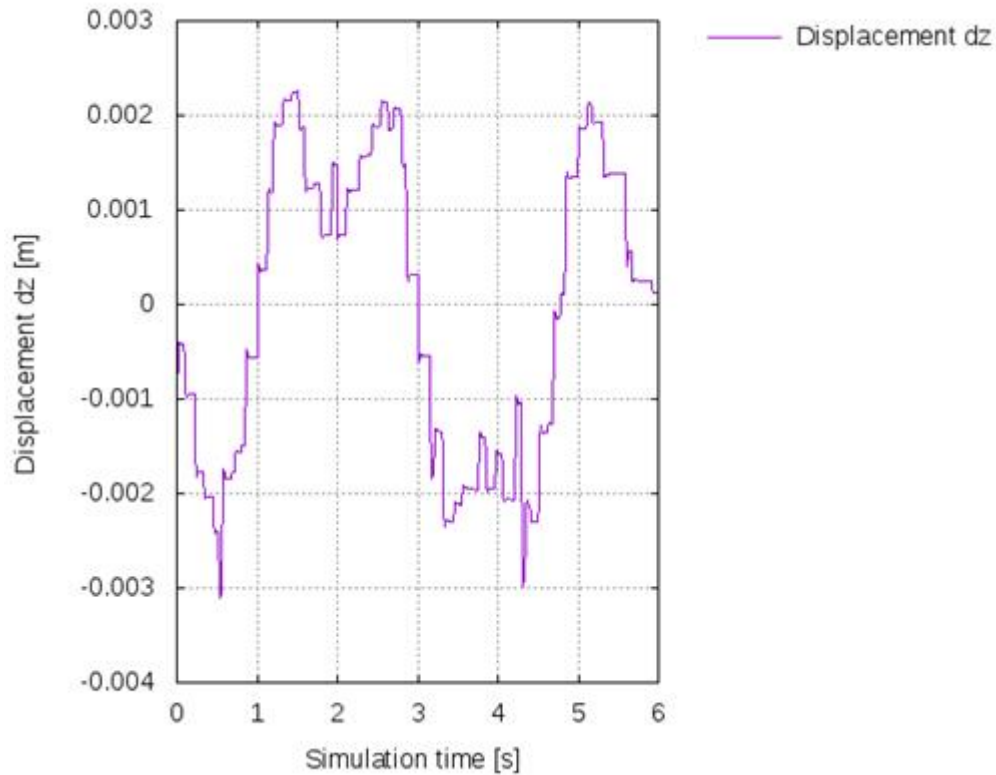


Figure 5.4: Deformation results along z-direction of four-point tent case in 6s simulation time

The current case only simulates the fluid flow over the tent membrane. This is because it was not possible to add the mesh for the fluid region that was meant to be below the tent membrane. Right now, it is not even certain how this can be achieved with OpenFOAM or foam-extend, given that even cyclic boundary conditions would not work for transferring flow between the top and bottom fluid regions. This means that this tent case is working in a similar way to the 3D membrane case studied in Section 4.4. Specifically, it is as if the building changed shape, but only the membrane roof was being simulated. In other words, it is as if the 3D membrane case was rotated 45° along Z-axis and the new North and South vortices of the membrane were lowered the ground, therefore morphing the flat membrane into the complex surface of the tent case.

The fluid properties are the same as the 3D membrane case in Section 4.4. The solid properties are the ones indicated for the PVC type I properties (Table 5.1). However, the

tent in this case, has a thickness of 1 cm and not 1 mm, because otherwise, the solver would crash due to numerical errors. The possibility of simulation with 1 mm of thickness would probably require a rather very well refined mesh on the solid region which was not possible to achieve due to the limited time available for this study.

The membrane is expected to only float up and down, like the 3D membrane case, given that there is no fluid flow going under the membrane and there is not enough deformation on the tent surface to cause vortices to be generated.

More specifically, it is possible to compare the characteristics of these two membranes, along with the fluid inlet velocity, to correlate the results reached in the two cases:

- The two membranes have similar surface areas (100 m^2 , when not distorted).
- The tent membrane is 4 times more rigid than the 3D membrane case, therefore it is expectable from this that the tent membrane has a displacement 4 times smaller than the achieved by the 3D membrane case.
- The inlet velocity for tent membrane has a maximum velocity of 30 m/s (uniformly distributed in space), which the 3D membrane case has a maximum velocity of 71.2 m/s (logarithmically distributed along the height), which leads to the tent case only receiving 50% of the fluid speed as the intake in the 3D membrane.
- The shape of the tent membrane allows for a substantial deflection of fluid flow to flow down and around the membrane, therefore reducing the displacement which would otherwise be necessary in the 3D membrane case, given that the latter would require the fluid to push the membrane down further before being able to flow back upwards and out of the membrane region.
- Therefore, taking these details into account, it could be expected to possibly have a displacement of at least 8 times smaller in the tent case (4 times more rigid and has at least half the fluid force imposed on it).
- In practice, the 3D membrane had a maximum displacement of roughly 0.045m (after the flow profile around it became periodic, i.e. after 3s of simulation), versus the 0.003m maximum which the tent membrane had, effectively making the tent

membrane displaced 15 times less than the 3D membrane.

- On the other hand, if the total displacement amplitude of the tent membrane is accounted for, namely 0.005m, this makes the tent membrane be displaced 9 times less than the 3D membrane, which is near the expected factor of 8 times.

The factor of 8 was aforementioned in the previous section, because the tent membrane has 4 times more stiffness than the 3D membrane, along with the wind velocity at the inlet on the 3D membrane is roughly twice faster than the maximum velocity at the inlet in the tent membrane case, making it the 3D membrane subjected to the estimated 8 times the scale of forces that are occurring in the tent membrane.

This analysis will be further addressed in Section 5.2, regarding the comparison of the two configurations of membranes.

Going back the case setup, the inverted V surfaces that are connected to the bottom of the fluid domain are defined with the slip boundary conditions. This was done because it was the most approximate boundary condition could use to try and partially replace the missing mesh underneath that region.

The `fsiFoam` solver and the Fluid-Structure Toolkit are still in development, and its limitations have restricted the scope of the simulations performed for the current thesis.

- Specially, that the cases could not be fully reproduced due to the limitations of both this toolkit and the experience that is needed to get these cases working properly.
- The tent case could not be reproduced both to limitations with the meshing requirements and solver limitations (foam-extend in this case as well).
- The reference case in [187,204] could not be 100% reproduced because the studies that have conducted for this thesis are all for laminar flow and the original model had the results for turbulence flow, which could not be performed with this solver and toolkit.

5.2 Learning from Membrane Structure Failure Cases

Despite the wide use of membrane structure for the construction's systems, some serious accidents have occurred on the part of or whole membrane roofs, particularly due to mistakes in the membranes design or improper installation process.

The flexibility behaviour of the membrane structure leads easily to induce instability in whole or part of its surface due to heavy rain or the wind during installation process before pre-tensioning. When it is raining, for example, if there is no good sloping pattern on the membrane for water flow, this is possible to cause a water basin inside the membrane, and that might exceed the designed load or stress. Then, the membrane structure may be damaged or even failure [200,205-206]. This can be observed in the results achieved with the tent membrane case and the 3D membrane case:

- When only air is flowing over the 3D membrane, the displacement plot reveals that the membrane will mostly oscillate into the building (mostly positive displacement), which will easily result in having a permanent water basin during heavy rain, with little chance of the membrane being displaced outwards and releasing the water.
- On the other hand, the tent membrane has two advantages, which will reduce or eliminate the occurrence of water basins:
 1. The centre of the membrane will oscillate in nearly equally proportion between a maximum position above the initial position and the minimum position below that initial centre position;
 2. The tent membrane already has sloping shape, specifically a saddle shape, which is nearly optimal for allowing for water and air to flow on top of the membrane.

The installation process of the membrane should be accomplished with a lot of care. For instance, during the installation step, the membrane surface had sometimes been poked by metallic equipment, and this could lead to slightly damaged. Thus, in this case, specific treatment must be immediately done because missing of that membrane surface treatment will decrease its load bearing capacity and finally lead to failure [200,207]. This was not possible to simulate with the tent case, but it could be tested in a future work.

In addition, mistakes in the membrane design or cutting pattern design can cause a higher stress concentration in a part of its surface, and that may lead to membrane cracks or failure. This is clearly demonstrated between the design of the 3D membrane versus the tent membrane, where the stress was substantially reduced, given that the tent membrane is displaced nearly half of the expected displacement in each direction (along Z). More specifically, the tent membrane was displaced between 0.002m and -0.003m, while the 3D membrane was displaced 0.045m in just one direction.

- More specifically, it is estimated that the 3D membrane would be displaced 0.0056m (maximum displacement observed was 0.045m, divided by the estimated factor of 8), if the 3D membrane was made of the same material as the tent membrane and simulated with the same wind speed at the inlet.
- This effectively means that the 3D membrane is subjected to nearly twice as much stress, than the tent membrane, when scaled accordingly, which would lead to a faster breakdown of the material due to stress.

The factor of 8 was aforementioned in the previous section, because the tent membrane has 4 times more stiffness than the 3D membrane, along with the wind velocity at the inlet on the 3D membrane is roughly twice faster than the maximum velocity at the inlet in the tent membrane case, making it the 3D membrane subjected to the estimated 8 times the scale of forces that are occurring in the tent membrane.

It is not easy sometimes to design the membrane surface in a homogenous stress in a service state. Therefore, in the design analysis results when this condition occurred, the high stressed membrane area should be established with more care, particularly, that area should be given by a local strengthening treatment [197,200]. This could possibly be simulated in a future work, for example, for assisting in the design of pre-stress distribution, when the membrane cannot be shaped directly to an optimum shape, such as the saddle shape.

5.3 Chapter Summary

Membrane structures are categorised by their flexibility and light-weight, which make

them relatively sensitive to wind flow. The partitioned approach is proposed to simulate this fluid-structure interaction problem. Moreover, due to the strong coupling in the membranes case and the almost negligible structure's mass, the Aitken based under-relaxation method is applied to accelerate convergence through the implicit fixed-point coupling.

The wind flow problem is presented by the incompressible Navier-Stokes equations while the membrane structure is modelled by the nonlinear elastodynamics equations. Both fluid and solid are simulated by the finite volume method (FVM), and an arbitrary Lagrangian-Eulerian (ALE) formulation is applied to capture the motion of the mesh.

The `fsiFoam` solver environment is using for the simulation of the wind flow effects on the slender membrane. The four-point tent case is taken as an example to present this work. One main challenge of the numerical wind simulation on constructions is the exact imitation of the natural wind conditions in the flow up-stream direction. This means that the inflow conditions of the fluid domain should be defined in accordance with the required task conditions. Generally, only the simplified wind profiles, which do not take into account the natural atmospheric turbulence, is used. However, studies of the experimental wind tunnel consist of these natural atmospheric turbulences.

For more application to wind-membrane interaction, the turbulence modelling such as large-eddy simulation (LES) and detached-eddy simulation (DES) methods are not available yet within `fsiFoam` solver in `foam-extend`.

Nonetheless, even though there were several limitations in the implemented models and available simulation capabilities, it was still possible to create a workflow that can be used for studying membrane designs, given that it was possible to achieve comparable results between the 3D membrane cases versus the tent case. The results which were achieved gave the clear result of why the tent membrane is overall a better design, given that it has reduced stress during the whole simulation, as well minimising/neutralising the occurrence of water basins on the membrane's surface.

Although, if time had permitted, intermediate simulations could have been performed for more easily comparing direct results and analysing design choices. For example, having both the tent membrane and the 3D membrane made of the same material, along with

having the same wind profile at the inlet, as well as having the 3D membrane rotated 45 degrees along the Z-axis, would have given a clear one-to-one comparison of the results.

Chapter 6

Conclusions and Future Research

The aim of the work as outlined in Section 1.6 was to investigate the impact of vortex-induced vibration (VIV) phenomena in two- and three-dimensional fluid-structure interaction (FSI) models. The thesis objective was achieved by studying two- and three-dimensional numerical models as discussed in Chapters 3 – 5. The partitioned approach was used in this study to analyse the strong coupling between fluid and structure. Different numerical models have been discussed in detail to demonstrate accuracy, stability, and convergence. The following sections summarise the achievements of this research effort, and present opportunities and directions for future work to improve upon and develop the work presented here.

6.1 Achievements Summary

Three different cases have been studied for the 2D flow past a circular cylinder example case; stationary cylinder, free vibrating cylinder, and forced vibrating cylinder. These three cases are simulated using `pimpleFoam` solver for the static cylinder and `pimpleDyMFoam` solver for the dynamic cylinder with PIMPLE algorithm in order to achieve pressure-velocity coupling. The results of the analysis are given as follows:

- In contrast to other hydrodynamic force coefficients for the static circular cylinder, both drag and lift coefficients represent a good agreement compared to open literature at Reynolds number equals 100 and 200. However, there is a significant difference in force coefficients at $Re = 1000$.
- An increasing of Reynolds number leads to an exponential increase of the lift coefficient amplitude.
- Compared to other experimental results, the relationship between Reynolds number and Strouhal number obtained from the simulation presents a good agreement.

- The solid body motion (free vibrating cylinder) was tested in one- and two-degrees-of-freedom system with linear spring and damping properties. Both affected the behaviour of the cylinder within the flow with some noticeable differences. More specifically, the response time of the cylinder and the drag coefficient were the most affected by the stiffness and damping coefficients used in the respective springs and dampers, where:
 - Reduced stiffness would lead to a smaller working range for the drag coefficient amplitude and take longer to reach to the periodic oscillation.
 - Reduced damping would lead to a slightly smaller working range for the drag coefficient amplitude, while also not working on the same exact frequency as the reference cases.
- From all tested simulations of the free vibration cylinder model, it is not clear what would happen if only 1 or 2 springs were used or if only one wire was used to hold the cylinder.
- In the free vibrating cylinder simulations, it was clear that the lift coefficient is barely affected by the springs and dampers, given that the cylinder would accompany the forces generated by the vortices that were being created and released in the cylinder's wake.
- With the forced vibrating cylinder simulation, it was clearer to infer how the drag and lift coefficients would correlate with the cylinder displacement and the vortices being generated in its wake, namely:
 - When the drag and lift coefficients are at a maximum and/ or the cylinder is at the extreme positions in displacement, and the flow is going over or under the cylinder for the respective extreme position of positive or negative displacements.
 - The “lock-in” phenomena will occur when the reduced velocity value is within the range [4, 12] and thus will cause the resonance.

A partitioned approach solver (fsiFoam) was applied to four models, two of them in the two-dimensional space and another two are their extensions in the three-dimensional space. In those four models, the strong coupling FSI was taken into account by applying block Gauss-Seidel implicit scheme with adaptive Aitken's under relaxation technique in

Sections 4.1, 4.2, and 4.4, and IQN-ILS technique in Section 4.3. That was essential especially in the partitioned approach model due to the different fluid flow and structural deformation solvers contact at the fluid-solid interface. The ALE formulation is applied to solve the Navier-Stokes equations as the internal mesh in the fluid flow domain deforms with the moving boundaries. Moreover, since the present benchmarking studies deal with either elastic tail or flexible membrane interacting with the viscous fluid flows, the Lagrangian formulation is used to discretise the St. Venant-Kirchhoff constitutive model which is the hyperelastic material model.

On the fluid-structure interface, the interaction between both solvers of the fluid flow and structure has been achieved. In addition, the solver of the mesh motion which uses the mesh diffusivity in the Laplace smoothing equation solves the motion of the internal mesh in the fluid flow domain and considers the displacement as the boundary condition at the fluid-solid interface. The conservation of mass and momentum of fluid flow and solid are provided by the coupling conditions for both velocities and boundary tractions. Space and time FVM discretisation implemented for both fluid and structure leads to a consistency of both continua.

The first benchmark that was used to validate the `fsiFoam` solver was the 2D flexible tail attached to a solid support block. The tail deformation is induced by the Von Karman vortex shedding from the solid square block. The analysis of the fluid flow displays the interaction in both directions. The tail deformation is excited by the unsteady vortices which in turn connect to the new vortices generating from the neighbouring fluid at the tip of the tail, and distributing the Von Karman vortex shedding street aside from its path.

Light-weight structures such as shells, membrane roofs and tents are presented for other examples models and applications. The well-known principle “great events often come from little causes” is very relevant in this context. This could be expressed by small changes in data causing an extremely important structural response. In the present work, this has been demonstrated by selecting a thin-walled membrane roof and a plate of medium slenderness (mass-less) ratio to show aerodynamic flow. The initial results provided some information about the difficulties in coupling, and ways or strategies to overcome them in the field of wind engineering.

A reliable structural design, such as a wind tunnel, is time-consuming and financially costly. For a strongly coupled system, numerical simulations may develop more efficient simulations, thereby reducing both time and cost. The presented frameworks are developed to examine the interaction of viscous fluid flow and slender structures in order to demonstrate the versatility and efficiency of the numerical scheme. The simulation provides an opportunity to demonstrate the coupled systems phenomena, and to analyse the reason for aeroelastic instabilities. For a viscous incompressible fluid, the natural wind is modelled at 25 °C.

However, as shown in this study, care must be taken when trying to replace wind tunnels with CFD simulation more so, when selecting meshing and modelling strategies, as well as the software and tools needed in these experiments. More specifically:

- 2D modelling can be used for initial experimentation with prototypes. However, this will unlikely fully represent real-life scenarios, given that a 2D simulation assumes that it is similar to a fully symmetrical flow and solid structure, happening in both directions of the third dimension. This was most visible in the 2D membrane case (Section 4.3) that was acting as a roof.
- Both 2D and 3D modelling requires that mesh studies are conducted, in order to determine the level of accuracy that is needed for a specific type of simulation, for example, for a specific flow profile at the inlet and a specific stiffness and elasticity of the membrane, given that both will affect how much fluid flow will interact with the solid membrane.
- Because of its limitation, turbulence modelling was not used in this study. This could have substantially hindered the scope of the simulations that were conducted. Namely, it could be possible to use coarser meshes for achieving similar results.
- The fluid-solid interaction toolkit which was used for this thesis (specifically the solver `fsiFoam`) was still a work in progress when this study was started. Thus, the ability to properly conduct a wide range of simulations was restricted due to being limited to a single CPU core when simulating. In other words, it was not possible to reduce the wall-clock time when dividing the mesh into sub-domains and assigning an independent CPU/core to each sub-domain, which is commonly done for most CFD software applications.

- Proper modelling of complex structures, such as the 3D tent membrane, requires advanced meshing software, which was not available during the time of this study. More specifically, the available strategies of meshing fluid regions in parts were used, then merging and stitching those parts together. This was a compromise, and it resulted in reaching limitations in foam-extend handling of multi-region meshes with fluid-solid interface surfaces as those in this study.

6.2 Suggestions for Future Work

In this study, the conducted investigations focused on Newtonian incompressible fluid flows interacting with flexible structures. The results documented in this study are significant. However, the following are suggestions and recommendations for future research:

- The two-dimensional and three-dimensional tests models rendering of turbulent and compressible flow.
- Applying the fluid-structure interaction modelling to more complex problems.
- Repeating the study accomplished in Chapter 5 using different membrane and structures shapes such as cone shape, and then apply to the net cable model.
- If the same software is to be used, it is strongly advised to work more closely with the developers of the foam-extend and the fluid-solid interaction toolkit projects, with the objective of improving the ability to handle the meshes needed for these types of simulations.

Bibliography

- [1] Billah, K.Y. and Scanlan, R.H. (1991). Resonance, Tacoma Narrows bridge failure, and undergraduate physics textbooks. *American Journal of Physics*, 59(2), pp.118-124.
- [2] Schäfer, F., Uffinger, T., Becker, S., Grabinger, J. and Kaltenbacher, M. (2008). Fluid-structure interaction and computational aeroacoustics of the flow past a thin flexible structure. *The Journal of the Acoustical Society of America*, 123(5), pp.3570-3575.
- [3] Gomes, J. and Lienhart, H. (2013). Fluid structure interaction-induced oscillation of flexible structures in laminar and turbulent flows. *Journal of Fluid Mechanics*, 715, pp.537-572.
- [4] Peng, Y., Mittal, R., Sau, A. and Hwang, R. (2010). Nested Cartesian grid method in incompressible viscous fluid flow. *Journal of Computational Physics*, 229(19), pp.7072-7101.
- [5] Pielhop, K., Klaas, M. and Schröder, W. (2013). Experimental analysis of the fluid structure interaction in finite-length straight elastic vessels. *European Journal of Mechanics - B/Fluids*, 50, pp.71-88.
- [6] Bearman, P. (2011). Circular cylinder wakes and vortex-induced vibrations. *Journal of Fluids and Structures*, 27(5-6), pp.648-658.
- [7] ESI-OpenCFD. (2004). OpenFOAM. *The Open Source CFD Toolbox*. Available at: <http://www.openfoam.com/>. Last accessed 21st July 2017.
- [8] Apacoglu, B., Paksoy, A. and Aradag, S. (2011). CFD analysis and reduced order modelling of uncontrolled and controlled laminar flow over a circular cylinder. *Engineering Applications of Computational Fluid Mechanics*, 5(1), pp.67-82.
- [9] Bayraktar, E., Mierka, O. and Turek, S. (2012). Benchmark computations of 3D laminar flow around a cylinder with CFX, OpenFOAM and FeatFlow. *International Journal of Computational Science and Engineering*, 7(3), pp.253-266.

- [10] The OpenFOAM Extend Project. (2014). Available at https://openfoamwiki.net/index.php/Extend-bazaar/Toolkits/Fluid-structure_interaction. Last accessed 21st July 2017.
- [11] Anagnostopoulos, P. (1994). Numerical investigation of response and wake characteristics of a vortex-excited cylinder in a uniform stream. *Journal of Fluids and Structures*, 8(4), pp.367-390.
- [12] Anagnostopoulos, P. (2000). Numerical study of flow past a cylinder excited transversely to the incident stream. Part I: Lock-in zone, hydrodynamic forces and wake geometry. *Journal of Fluids and Structures*, 14(6), pp.819-851.
- [13] Placzek, A., Sigrist, J. and Hamdouni, A. (2009). Numerical simulation of an oscillating cylinder in a cross-flow at low Reynolds number: Forced and free oscillations. *Computers & Fluids*, 38(1), pp.80-100.
- [14] Govardhan, R. and Williamson, C. (2000). Modes of vortex formation and frequency response of a freely vibrating cylinder. *Journal of Fluid Mechanics*, 420, pp.85-130.
- [15] Williamson, C. and Jauvtis, N. (2004). A high-amplitude 2T mode of vortex-induced vibration for a light body in motion. *European Journal of Mechanics - B/Fluids*, 23(1), pp.107-114.
- [16] Bourguet, R., Karniadakis, G. and Triantafyllou, M. (2011). Vortex-induced vibrations of a long flexible cylinder in shear flow. *Journal of Fluid Mechanics*, 677, pp.342-382.
- [17] Mittal, S. (2005). Excitation of shear layer instability in flow past a cylinder at low Reynolds number. *International Journal for Numerical Methods in Fluids*, 49(10), pp.1147-1167.
- [18] Dahl, J.M., Hover, F.S. and Triantafyllou, M.S. (2006). Two-degree-of-freedom vortex-induced vibrations using a force assisted apparatus. *Journal of Fluids and Structures*, 22(6-7), pp.807-818.

- [19] Dahl, J.M., Hover, F.S., Triantafyllou, M.S. and Oakley, O.H. (2010). Dual resonance in vortex-induced vibration at subcritical and supercritical Reynolds numbers. *Journal of Fluid Mechanics*, 643, pp.395-424.
- [20] Wang, X., Su, B. and Tan, S. (2012). Experimental study of vortex-induced vibrations of a tethered cylinder. *Journal of Fluids and Structures*, 34, pp.51-67.
- [21] Wang, Z.J. and Zhou, Y. (2005). Vortex-Induced Vibration Characteristics of an Elastic Square Cylinder on Fixed Supports. *Journal of Fluids Engineering.*, 127(2), pp.241-249.
- [22] Williamson, C. and Roshko, A. (1998). Vortex formation in the wake of an oscillating cylinder. *Journal of Fluids and Structures*, 2(4), pp.355-381.
- [23] Carberry, J., Sheridan, J. and Rockwell, D. (2005). Controlled oscillations of a cylinder: forces and wake modes. *Journal of Fluid Mechanics*, 538, pp.31-69.
- [24] Bearman, P.W. (2009). Understanding and predicting vortex-induced vibrations. *Journal of Fluid Mechanics*, 634, p.1-4.
- [25] Bearman, P.W. (2011). Circular cylinder wakes and vortex-induced vibrations. *Journal of Fluids and Structures*, 27(5-6), pp.648-658.
- [26] Schäfer, F., Uffinger, T., Becker, S., Grabinger, J. and Kaltenbacher, M. (2008). Fluid-structure interaction and computational aeroacoustics of the flow past a thin flexible structure. *Journal of Acoustical Society of America*, 123(5), pp.3570-3575.
- [27] Brika, D. and Laneville, A. (1993). Vortex-induced vibrations of a long flexible circular cylinder. *Journal of Fluid Mechanics*, 250, pp.481-508.
- [28] Carberry, J., Sheridan, J. and Rockwell, D. (2003). Controlled oscillations of a cylinder: a new wake state. *Journal of Fluids and Structures*, 17(2), pp.337-343.
- [29] Williamson, C.H.K. and Govardhan, R. (2004). Vortex-induced vibrations. *Annual Review of Fluid Mechanics*, 36, pp.413-455.

- [30] Blackburn, H.M. and Karniadakis, G.E. (1993, January). Two-and Three-Dimensional Simulations of Vortex-Induced Vibration Or a Circular Cylinder. In *The Third International Offshore and Polar Engineering Conference*. International Society of Offshore and Polar Engineers.
- [31] Morse, T.L. and Williamson, C.H. (2009). Prediction of vortex-induced vibration response by employing controlled motion. *Journal of Fluid Mechanics*, 634, pp.5-39.
- [32] Leong, C. and Wei, T. (2008). Two-degree-of-freedom vortex-induced vibration of a pivoted cylinder below critical mass ratio. In *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 464(2099), pp.2907-2927.
- [33] Srinil, N., Zanganeh, H. and Day, A. (2013). Two-degree-of-freedom VIV of circular cylinder with variable natural frequency ratio: Experimental and numerical investigations. *Ocean Engineering*, 73, pp.179-194.
- [34] Xie, F., Deng, J., Xiao, Q. and Zheng, Y. (2012). A numerical simulation of VIV on a flexible circular cylinder. *Fluid Dynamics. Research*, 44(4), p.045508.
- [35] Xie, F.F, Jian, D.E. and Zheng, Y. (2011). Multi-mode of vortex-induced vibration of a flexible circular cylinder. *Journal of Hydrodynamics, Ser. B*, 23(4), pp.483-490.
- [36] Zhou, C.Y., So, R.M.C. and Lam, K. (1999). Vortex-induced vibrations of an elastic circular cylinder. *Journal of Fluids and Structures*, 13(2), pp.165-189.
- [37] Li, T., Zhang, J.Y., Zhang, W.H. and Zhu, M.H. (2009). Vortex-induced vibration characteristics of an elastic circular cylinder. *World Academy of Science, Engineering and Technology*, 60, pp.56-65.
- [38] Anagnostopoulos, P. and Bearman, P.W. (1992). Response characteristics of a vortex-excited cylinder at low Reynolds numbers. *Journal of Fluids and Structures*, 6(1), pp.39-50.
- [39] Khalak, A. and Williamson, C.H. (1997). Investigation of relative effects of mass and damping in vortex-induced vibration of a circular cylinder. *Journal of Wind Engineering and Industrial Aerodynamics*, 69, pp.341-350.

- [40] Papadakis, G. (2009). Coupling 3D and 1D fluid-structure-interaction models for wave propagation in flexible vessels using a finite volume pressure-correction scheme. *Communication Numerical Methods in Engineering*, 25(5), pp.533-551.
- [41] Tallec, P.L. and Mouro, J. (2001). Fluid-structure interaction with large structural displacements. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25), pp.3039-3067.
- [42] Dettmer, W.G. (2004). *Finite Element Modelling of Fluid Flow with Moving Free Surfaces and Interfaces Including Fluid-Solid Interaction*. Ph.D. Thesis, School of Engineering, University of Wales, Swansea.
- [43] Dettmer, W.G. and Perić, D. (2006). A computational framework for fluid-structure interaction: finite element formulation and applications. *Computer Methods in Applied Mechanics and Engineering*, 195(41-43), pp.5754–5779.
- [44] Von Scheven, M. and Ramm, E. (2011). Strong coupling schemes for interaction of thin-walled structures and incompressible flows. *International Journal for Numerical Methods in Engineering*, 87(1-5), pp.214-231.
- [45] Habchi, C., Russeil, S., Bougeard, D., Harion, J.L., Lemenand, T., Ghanem, A., Della Valle, D. and Peerhossaini, H. (2013). Partitioned solver for strongly coupled fluid–structure interaction. *Computers & Fluids*, 71, pp. 306-319.
- [46] Glück, M., Breuer, M., Durst, F., Halfmann A. and Rank, E. (2001). Computation of fluid–structure interaction on lightweight structures. *Journal of Wind Engineering and Industrial Aerodynamics*, 89(14), pp.1351–1368.
- [47] Glück, M., Breuer, M., Durst, F., Halfmann A. and Rank, E. (2003). Computation of wind-induced vibrations of flexible shells and membranous structures. *Journal of Fluids and Structures*, 17(5), pp.739–765.
- [48] Simiu, E. and Scanlan, R.H. (1996). Wind effects on structures: Fundamentals and application to design. *Book published by John Willey & Sons Inc*, 605.

- [49] Bagn eris, M. (2009). Innovative Surface Structures, Technologies and Applications: A Review of Martin Bechthold's Book. *International Journal of Space Structures*, 24(1), pp.59-61.
- [50] Bungartz, H.J., and Sch afer, M. eds. (2006). *Fluid-Structure Interaction: Modelling, Simulation, Optimisation (Vol. 53)*. Springer Science & Business Media.
- [51] Yang, Q., Wang, J. and Wang L. (2003). Interaction of Wind with Fabric Structures. *Spatial Structures*, 9(1), pp.20–24.
- [52] Droll, P., Sieber, R., Cozzani, A. and Schafer, M. (2002). HAUPTAUFSATZ- Numerical investigation of the performance of a deep space antenna under environmental loads. *Bauingenieur*, 77(1), pp.7-11.
- [53] Elashkar, I. and Novak, M. (1983). Wind Tunnel Studies of Cable Roofs. *Journal of Wind Engineering and Industrial Aerodynamics*, 13, pp.407–419.
- [54] Kawamura, S. and Kiuchi, T. (1986). An Experimental Study of a One-membrane Type Pneumatic Structure –Wind Load and Response. *Journal of Wind Engineering and Industrial Aerodynamics*, 23, pp.127–140.
- [55] Takeda, T., Kageyama, M. and Homma, Y. (1986). Experimental Studies on Structural Characteristics of a Cable-reinforced Air-supported Structure. Shells. In *Membrane and Space Frames, Proceedings of IASS Symposium*, Osaka, pp.141–148.
- [56] Ishii, K. (1997). Membrane Structures in Japan —Technologies for Supporting Membrane Structures. In *IASS International Symposium on Shell & Spatial Structures*, Singapore, pp.15–26.
- [57] Daw, D.J. and Davenport, A.G. (1989). Aerodynamic damping and stiffness of a semi-circular roof in turbulent wind. *Journal of Wind Engineering and Industrial Aerodynamics*, 32(1-2), pp.83-92.
- [58] Novak, M. and Kassem, M. (1990). Free Vibration of Light Roofs Backed by Cavities. *Journal of the Engineering Mechanics Division*, 116(3), pp.549–564. T,

- [59] Il'chenko, A.V. and Temnenko, V.A. (1993). Oscillations of a membrane that is orthogonal to a flow: Asymptotics of large stresses. *Journal of Mathematical Sciences*, 65(2), pp.1521-1525.
- [60] Kawai, H., Yoshie, R., Wei, R. and Shimura, M. (1999). Wind-induced Response of a Large Cantilevered Roof. *Journal of Wind Engineering and Industrial Aerodynamics*, 83, pp.263–275.
- [61] Yang, Q., Wu, Y. and Zhu, W. (2010). Experimental study on interaction between membrane structures and wind environment. *Earthquake Engineering and Engineering Vibration*, 9(4), pp.523-532.
- [62] Hessenthaler, A., Gaddum N. R., Holub, O., Sinkus, R., Röhrle, O. and Nordsletten, D. (2017). Experiment for validation of fluid-structure interaction models and algorithms. *International journal for numerical methods in biomedical engineering*, 33(9).
- [63] Jasak, H. (1996). *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. Ph.D. Thesis, University of London Imperial College.
- [64] Donea, J., Huerta, A., Ponthot, J.P. and Rodriguez-Ferran, A. (2004). Arbitrary Lagrangian – Eulerian Methods. In: Stein, E., de Borst, R., and Hughes, T. J. *Encyclopedia of Computational Mechanics*. Chichester: John Wiley, pp.413-433.
- [65] Olivier, M., Dumas, G. and Morissette, J. (2009). A fluid–structure interaction solver for nano-air-vehicle flapping wings. In *Proceedings of the 19th AIAA Computational Fluid Dynamics Conference*. San Antonio, USA, pp.1–15.
- [66] Tuković, Ž. and Jasak, H. (2007). Updated Lagrangian finite volume solver for large deformation dynamic response of elastic body. *Transaction of FAMENA*, 31(1), pp.1-16.
- [67] Jasak, H. and Weller, H.G. (2000). Application of the finite volume method and unstructured meshes to linear elasticity. *International Journal of Numerical Methods Engineering*, 48(2), pp.267-287.

- [68] Hirsch, C. (2007). *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Butterworth-Heinemann.
- [69] Muzaferija, S. (1994). *Adaptive Finite Volume method for flow prediction using unstructured meshes and multigrid approach*, Ph.D. thesis, Imperial College, University of London.
- [70] Marić, T., Höpken, J. and Mooney, K. (2014). *The OpenFOAM Technology Primer*. Germany: sourceflux UG (haftungsbeschränkt).
- [71] Issa, R.I. (1986). Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computers and Physics*, 62(1), pp.40-65.
- [72] Patankar, S.V. (1980). *Numerical heat transfer and fluid flow*. Hemisphere, New York, pp.25-73.
- [73] Rhie, C.M. and Chow, W.L. (1983). Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal (ISSN 0001-1452)*, 21, pp.1525-1532.
- [74] Moukalled, F., Mangani, L. and Darwish, M. (2016). *The Finite Volume Method in Computational Fluid Dynamics. An Advanced Introduction with OpenFOAM and Matlab*. Switzerland: Springer International Publishing Switzerland, pp.85-364.
- [75] Perić, M. (1985). *A Finite Volume Method for The Prediction of Three-dimensional Fluid Flow in Complex Ducts*. Ph.D. Thesis, University of London.
- [76] Yao, H.D. (2014). Simulation of Fluid-Structural Interaction using OpenFOAM. *Simulation*, (1/37).
- [77] Tuković, Ž. and Jasak, H. (2009). *FVM for FSI with large structural displacements*. Available at: http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2009/FSIslides.pdf. Last accessed 8th July 2017.
- [78] Michler, C., Hulshoff, S.J., van Brummelen, E.H. and de Borst, R. (2004). A monolithic approach to fluid–structure interaction. *Computers & Fluids*, 33(5-6), pp.839-848.

- [79] Degroote, J., Bathe, K.J. and Vierendeels, J. (2009). Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction, *Computers & Structures*, 87, pp.793-801.
- [80] Jasak, H. and Tuković, Ž. (2010). Dynamic mesh handling in OpenFOAM applied to fluid-structure interaction simulations. In *Proceedings of the V European Conference on Computational Fluid Dynamics ECCOMAS CFD*. Lisbon, Portugal, pp.14-17.
- [81] Bonet, J. and Wood, R.D.(2005). *Nonlinear Continuum Mechanics for Finite Element* . 6th ed. New York: Cambridge University Press.
- [82] Yigit, S., Stenel, D.D. and Schäfer, M. (2007). Efficiency of fluid-structure interaction simulations with adaptive underrelaxation and multigrid acceleration. *The International Journal of Multiphysics*, 1(1), pp.85-99.
- [83] Degroote, J., Haelterman, R., Annerel, S., Swillens, A., Segers, P. and Vierendeels, J. (2009). An interface quasi-Newton algorithm for partitioned simulation of fluid-structure interaction. In *International Workshop on Fluid-Structure Interaction. Theory, Numerics and Applications* (p. 55). kassel university press GmbH.
- [84] Bathe, K.J. and Baig, M.M.I. (2005). On a composite implicit time integration procedure for nonlinear dynamics. *Computers & Structures*, 83(31-32), pp.2513–2524.
- [85] Dwight, R.P. (2006). Robust mesh deformation using the linear elasticity equations. In: Deconinck, H. and Dick, E. *Computational Fluid Dynamics*. Berlin, Heidelberg: Springer, pp.401-406.
- [86] Jasak, H. and Tuković, Ž. (2006). Automatic mesh motion for the unstructured finite volume method. *Transaction of FAMENA*, 30(2), pp.1–20.
- [87] Jasak, H. and Tuković, Ž. (2007). Automatic mesh motion for the unstructured finite volume method. *Transaction FAMENA*, 30(2), pp.1-18.

- [88] Farhat, C., Van der Zee, K.G. and Geuzaine, P. (2006) Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity. *Computer Methods in Applied Mechanics and Engineering*, 195(17), pp.1973–2001.
- [89] Forster, C., Wall, W.A. and Ramm, E. (2007). Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows. *Computer methods in applied mechanics and engineering*, 196(7), pp.1278–1293.
- [90] Vyzikas, T., Nilsson, H. and Andric, J. (2015). *CFD with OpenSource software: The implementation of interFoam solver as a flow model of the fsiFoam solver for strong fluid-structure interaction*.
- [91] Cardiff, P. (2012). *Development of the finite volume method for hip joint stress analysis*. Ph.D. Thesis, School of Mechanical and Materials Engineering, University College Dublin, Dublin.
- [92] Tuković, Ž. (2005). *Finite volume method on domains of varying* Ph.D. Thesis, Faculty of mechanical engineering and naval architecture, University of Zagreb, Croatia.
- [93] Tuković, Ž., Cardiff, P., Karač, A., Jasak, H. and Ivankovic, A. (2014a). *Parallel unstructured finite- volume method for fluid-structure interaction*. [Manuscript]. At: Zagreb: Faculty of mechanical engineering and naval architecture, University of Zagreb.
- [94] Tuković, Ž., Cardiff, P., Karač, A., Jasak, H. and Ivankovic, A. (2014b). OpenFOAM library for fluid-structure interaction. In *9th OpenFOAM workshop* (Vol. 2014) Faculty of mechanical engineering and naval architecture, University of Zagreb, Croatia.
- [95] Badia, S., Quaini, A. and Quarteroni, A. (2008). Modular vs. non-modular preconditioners for fluid–structure systems with large added-mass effect. *Computer Methods in Applied Mechanics and Engineering*, 197, pp.4216-4232.

- [96] Bazilevs, Y., Takizawa, K. and Tezduyar, T.E. (2013). *Computational Fluid-Structure Interaction. Methods and Applications*. United Kingdom: John Wiley & Sons Ltd.
- [97] Causin, P., Gerbeau, J.F. and Nobile, F. (2005). Added-mass effect in the design of partitioned algorithms for fluid–structure problems. *Computer Methods in Applied Mechanics and Engineering*, 194, pp.4506-4527.
- [98] Crosetto, P., Reymond, P., Deparis, S., Kontaxakis, D., Stergiopoulos, N. and Quarteroni, A. (2011). Fluid-structure interaction simulation of aortic blood flow. *Computers & Fluids*, 43(1), pp.46-57.
- [99] Greenshields, C.J. and Weller, H.G. (2005). A unified formulation for continuum mechanics applied to fluid–structure interaction in flexible tubes. *International Journal of Numerical Methods Engineering*, 64(12), pp.1575-1593.
- [100] Heil, M., Hazel, A. and Boyle, J. (2008). Solvers for large–displacement fluid–structure interaction problems: segregated versus monolithic approaches. *Computers & Mechanics*, 43, pp.91-101.
- [101] Walhorn, E., Kolke, A., Hubner, B. and Dinkler, D. (2005). Fluid–structure coupling within a monolithic model involving free surface flows. *Computers & Structures*, 83(25–26), pp.2100-2111.
- [102] Cirak, F., Deiterding, R. and Mauch, S.P. (2007). Large-scale fluid–structure interaction simulation of viscoplastic and fracturing thin-shells subjected to shocks and detonations. *Computers & Structure*, 85(11–14), pp.1049-1065.
- [103] Karac, A., Blackman, B.R.K, Cooper, V., Kinloch, A.J., Sanchez, S.R., Teo, W.S. and Ivankovic, A. (2011). Modelling the fracture behaviour of adhesively-bonded joints as a function of test rate. *Engineering Fracture Mechanics*, 78(6), pp.973-989.
- [104] Degroote, J. Haelterman, R., Annerel, S., Bruggeman, P. and Vierendeels, J. (2010). Performance of partitioned procedures in fluid-structure interaction. *Computers & Structures*, 88(7–8), pp.446-457.

- [105] Gerbeau, J.F., Vidrascu, M. and Frey, P. (2005). Fluid–structure interaction in blood flows on geometries based on medical imaging. *Computers & Structures*, 83(2–3), pp.155-165.
- [106] Kassiotis, C., Ibrahimbegovic, A., Niekamp, R. and Matthies, H. (2011a). Nonlinear fluid-structure interaction problem. Part I: implicit partitioned algorithm, nonlinear stability proof and validation examples. *Computers & Mechanics*, 47, pp.305-323.
- [107] Olivier, M. and Dumas, G. (2009). Non-linear aeroelasticity using an implicit partitioned finite volume solver. In *Proceedings of the 17th Annual Conference of the CFD Society of Canada*. Ottawa, Canada.
- [108] Patankar, S. and Spalding, D. (1972). A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10), pp.1787-1806.
- [109] Wall, W.A., Genkinger, S. and Ramm, E. (2007). A strong coupling partitioned approach for fluid–structure interaction with free surfaces. *Computers & Fluids*, 36(1), pp.169-183.
- [110] Kuttler, U. and Wall, W. (2008). Fixed-point fluid–structure interaction solvers with dynamic relaxation. *Computers & Mechanics*, 43, pp.61-72.
- [111] Wood, C., Gil, A., Hassan, O. and Bonet, J. (2010). Partitioned block-Gauss–Seidel coupling for dynamic fluid–structure interaction. *Computers & Structures*, 88(23–24), pp.1367-1382.
- [112] Fernández, M.A. and Moubachir, M. (2003). An exact Block-Newton algorithm for the solution of implicit time discretised coupled systems involved in fluid–structure interaction problems. In Bathe, K.J. *Computational Fluid and Solid Mechanics*, Oxford: Elsevier Science Ltd, pp.1337-1341.
- [113] Fernández, M.A. and Moubachir, M. (2005). A Newton method using exact Jacobians for solving fluid–structure coupling. *Computers & Structures*, 83, pp.127-142.

- [114] Gallinger, T. and Bletzinger, K. (2010). Comparison of algorithms for strongly coupled partitioned fluid–structure interaction – efficiency versus simplicity. In *Proceedings of the V European Conference on Computational Fluid Dynamics ECCOMAS CFD*. Lisbon, Portugal, pp.1-20.
- [115] Yvin, C. (2010). Partitioned fluid-structure interaction with open-source tools. 12ème Journées de l’Hydrodynamique. *Nantes, France*.
- [116] Irons, B.M. and Tuck, R.C. (1969). A version of the Aitken accelerator for computer iteration. *International Journal of Numerical Methods Engineering*, 1(3), pp.275-277.
- [117] Bos, F. (2010). *Numerical simulations of flapping foil and wing aerodynamics – mesh deformation using radial basis functions*. Ph.D. Thesis, Technical University of Delft, Netherlands.
- [118] Donea, J., Giuliani, S. and Halleux, J.P. (1982). An arbitrary Lagrangian–Eulerian finite element method for transient dynamic fluid–structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33, pp.689-723.
- [119] Souli, M., Ouahsine, A. and Lewin, L. (2000). ALE formulation for fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 190(5), pp. 659-675.
- [120] Bathe, K. and Hahn, W. (1979). On transient analysis of fluid–structure systems. *Computers & Structures*, 10(1–2), pp.383-391.
- [121] Razzaq, M., Damanik, H., Hron, J., Ouazzi, A. and Turek, S. (2012). FEM multigrid techniques for fluid-structure interaction with application to hemodynamics. *Applied Numerical Mathematics*, 62(9), pp.1156-1170.
- [122] Sathe, S., Benney, R., Charles, R., Doucette, E., Miletti, J., Senga, M., Stein, K. and Tezduyar, T.E. (2007). Fluid–structure interaction modelling of complex parachute designs with the space–time finite element techniques. *Computers & Fluids*, 36(1), pp.127-135.

- [123] Bos, F., van Oudheusden, B.W. and Bijl, H. (2008). Moving and deforming meshes for flapping flight at low Reynolds numbers. *Delft University of Technology*, 2, p.19.
- [124] Kassiotis, C. (2009). *Nonlinear fluid-structure interaction: a partitioned approach and its application through component technology*. Ph.D Thesis, Université Paris-Est, France.
- [125] OpenFOAM Foundation. (2013). *OpenFOAM User Guide, version 2.2.2*.
- [126] Weller, H.G., Tabor, G., Jasak, H. and Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers & Physics*, 12, pp.620-631.
- [127] Jasak, H., Jemcov, A. and Tukovic, Z. (2007). OpenFOAM: a C++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics* (Vol.1000). IUC Dubrovnik, Croatia, pp.1-20.
- [128] Kassiotis, C. (2009). *Nonlinear fluid-structure interaction: a partitioned approach and its application through component technology*. Ph.D Thesis, Université Paris-Est, France.
- [129] Kassiotis, C., Ibrahimbegovic, A. and Matthies, H. (2010). Partitioned solution to fluid–structure interaction problem in application to free-surface flows. *European Journal of Mechanics – B/Fluids*, 29(6), pp.510-521.
- [130] Lohner, R. and Yang, C. (1996). Improved ale mesh velocities for moving bodies. *Communication Numerical Methods Engineering*, 12(10), pp.599-608.
- [131] Deparis, S. (2004). *Numerical analysis of axisymmetric flows and methods for fluid-structure interaction arising in blood flow simulation*. Ph.D. Thesis, EPFL, Lausanne.
- [132] Ferziger, J.H. and Perić, M. (2012). *Computational Methods for Fluid Dynamics*. Springer Science & Business Media.
- [133] Stokes, G.G. (1851). *On the effect of the internal friction of fluids on the motion of pendulums* (Vol. 9, p. 8). Cambridge: Pitt Press.

- [134] Sumer, B.M. and Fredsøe, J. (1997). *Hydrodynamics Around Cylindrical Structures*. World Scientific.
- [135] Techet, A.H. (2005). *13.42 Lecture: Vortex Induced Vibrations*. Massachusetts Institute of Technology, Open Courseware, 21.
- [136] Lo, S., Hoffmann, K. and Dietiker, J. (2005). Numerical Investigation of High Reynolds Number Flows Over Square and Circular Cylinders. *Journal of Thermophysics and Heat Transfer*, 19(1), pp.72-80.
- [137] Friehe, C.A. (1980). Vortex shedding from cylinders at low Reynolds numbers. *Journal of Fluid Mechanics*, 100(2), pp.237-241.
- [138] Mittal, S. and Tezduyar, T. (1992). A finite element study of incompressible flows past oscillating cylinders and aerofoils. *International Journal for Numerical Methods in Fluids*, 15(9), pp.1073-1118.
- [139] Anagnostopoulos, P. (2000). Numerical study of the flow past a cylinder excited transversely to the incident stream. Part 1: Lock-in zone, hydrodynamic forces and wake geometry. *Journal of Fluids and Structures*, 14(6), pp.819-851.
- [140] Govardhan, R.N. and Williamson, C.H. (2006). Defining the modified Griffin plot in vortex-induced vibration: revealing the effect of Reynolds number using controlled damping. *Journal of Fluid Mechanics*, 561, pp.147-180.
- [141] Raghavan, K. (2007). *Energy extraction from a steady flow using Vortex Induced Vibration*. Ph.D Thesis, University of Michigan, Ann Arbor.
- [142] Raghavan, K. and Bernitsas, M. (2011). Experimental investigation of Reynolds number effect on vortex induced vibration of rigid circular cylinder on elastic supports. *Ocean Engineering*, 38(5-6), pp.719-731.
- [143] Koushan, K. (2009). *Vortex Induced Vibrations of Free Span Pipelines*. Ph.D. Thesis, Norwegian University of Science and Technology, Trondheim, Norway.
- [144] Mittal, S. and Kumar, V. (1999). Finite element study of vortex-induced cross-flow and in-line oscillations of a circular cylinder at low Reynolds numbers. *International Journal for Numerical Methods in Fluids*, 31(7), pp.1087-1120.

- [145] Norberg, C. (2001). Flow around a circular cylinder: aspects of fluctuating lift. *Journal of Fluids and Structures*, 15(3-4), pp.459-469.
- [146] Okajima, A., Nakamura, A., Kosugi, T., Uchida, H. and Tamaki, R. (2004). Flow-induced in-line oscillation of a circular cylinder. *European Journal of Mechanics - B/Fluids*, 23(1), pp.115-125.
- [147] Ribeiro, J.D. (1992). Fluctuating lift and its spanwise correlation on a circular cylinder in a smooth and in a turbulent flow: a critical review. *Journal of Wind Engineering and Industrial Aerodynamics*, 40(2), pp.179-198.
- [148] Feng, C.C. (1968). *The measurement of vortex induced effects in flow past stationary and oscillating circular and d-section cylinders* Ph.D. Thesis, University of British Columbia.
- [149] Rahman, M.M., Karim, M.M. and Alim, M.A (2007). Numerical investigation of unsteady flow past a circular cylinder using 2-D finite volume method. *Journal of Naval Architecture and Marine Engineering*, 4(1), pp.27-42.
- [150] Lienhard, J.H. (1966). *Synopsis of lift, drag, and vortex frequency data for rigid circular cylinders (Vol. 300)*. Technical Extension Service, Washington State University.
- [151] Diana, G., Falco, M., Cigada, A. and Manenti, A. (2000). On the measurement of overhead transmission lines conductor self-damping. *IEEE Transactions on Power Delivery*, 15(1), pp.285-292.
- [152] Blackburn, H., Govardhan, R. and Williamson, C. (2001). A complementary numerical and physical investigation of vortex-induced vibration. *Journal of Fluids and Structures*, 15(3-4), pp.481-488.
- [153] Gonçalves, R.A., Teixeira, P.R.D.F. and Didier, E.L. (2012). Numerical simulations of low Reynolds number flows past elastically mounted cylinder.
- [154] Reynolds, O. (1894). On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Proceeding of the Royal Society of London*, 56(336-339), pp.40-45.

- [155] Govardhan, R. and Williamson, C. (1999). Vortex-induced vibrations of a sphere. *Journal of Fluid Mechanics*, 531, pp.11-47.
- [156] Gavin, H.P. (2014). Vibrations of single degree of freedom systems. *Department of Civil and Environmental Engineering, Duke University*.
- [157] Khalak, A. and Williamson, C. (1999). Motions, forces and mode transitions in vortex induced vibrations at low mass-damping, *Journal of Fluids Structures*, 13, pp.813–851.
- [158] Greenshields, C.J. (2014). Openfoam user guide. *OpenFOAM Foundation Ltd, version, 2.2.x*
- [159] CFD Direct. (2017). *Mesh generation with blockMesh*. Available at: <http://cfdirect.com/openfoam/user-guide/blockmesh/>. Last Accessed 6th June 2017.
- [160] Anderson, J. D. (2001). *Fundamentals of aerodynamics*. 3rd edn. Boston: McGraw-Hill Series in Aeronautical and Aerospace Engineering.
- [161] Park, D.S., Ladd, D.M. and Hendricks, E.W. (1994). Feedback control of von Kármán vortex shedding behind a circular cylinder at low Reynolds numbers. *Physics of fluids*, 6(7), pp.2390-2405.
- [162] Mittal, S. and Raghuvanshi, A. (2001). Control of vortex shedding behind circular cylinder for flows at low Reynolds numbers. *International Journal for Numerical Methods in Fluids*, 35(4), pp.421-447.
- [163] Patnana, V.K., Bharti, R P., and Chhabra, R.P. (2009). Two-dimensional unsteady flow of power-law fluids over a cylinder. *Chemical Engineering Science*, 64(12), pp.2978-2999.
- [164] Lamas, I.M, Rodriguez, C.G. (2014). *CFD with OpenFOAM*. [Lecture to Technical Courses, Spain], 15th January 2014.
- [165] The OpenFOAM Foundation. (n.d). *Standard solvers*. Available at: <http://www.openfoam.org/features/standard-solvers.php>. Last accessed 15th June 2017.

- [166] OpenFOAMWiki. (2009). *pimpleFoam*. Available at: <https://openfoamwiki.net/index.php/PimpleFoam>. Last accessed 10th May 2017.
- [167] Fey, U., König, M., Eckelmann, H. (1998). A new Strouhal–Reynolds-number relationship for the circular cylinder in the range $47 < Re < 2 \cdot 10^5$ *Physics of Fluids*, 10(7), pp. 1547-1549.
- [168] Norberg, C. (2003). Fluctuating lift on a circular cylinder: review and new measurements. *Journal of Fluids and Structures*, 17(1), pp.57-96.
- [169] Pastò, S. (2008). Vortex-induced vibrations of a circular cylinder in laminar and turbulent flows. *Journal of Fluids and Structures*, 24(7), pp.977-993.
- [170] Berthelsen, P.A. and Faltinsen, O.M. (2007). A local directional ghost cell approach for incompressible viscous flow problems with irregular boundaries. *Journal of Computational Physics*, 227(2008), pp.4354–4397.
- [171] Calhoun, D. (2002). A Cartesian grid method for solving the two-dimensional stream function-vorticity equations in irregular regions. *Journal of Computational Physics*, 176(2), pp.231–275.
- [172] Franke, R., Rodi, W. and Schönung, B. (1990). Numerical calculation of laminar vortex shedding flow past cylinders. *Journal of Wind Engineering and Industrial Aerodynamics*, 35, pp. 237-257.
- [173] Herfjord, K. (1996). *A study of two-dimensional separated flow by a combination of the finite element method and Navier–Stokes equations*. Ph.D. Thesis, Department of Marine Hydrodynamics, Norwegian Institute of Technology, Trondheim: Norway.
- [174] Linnick, M.N and Fasel, H.F. (2005). A high-order immersed interface method for simulating unsteady compressible flows on irregular domains. *Journal of Computational Physics*, 204(1), pp.157–192.
- [175] Rajani, B. N., Kandasamy, A. and Majumdar, S. (2009). Numerical simulation of laminar flow past a circular cylinder. *Journal of Applied Mathematical Modelling*, 33(3), pp.1228-1247.

- [176] Russell, D. and Wang, Z. (2003). A cartesian grid method for modelling multiple moving objects in 2D incompressible viscous flow. *Journal of Computational Physics*, 191(1), pp.177-205.
- [177] Xu, S. and Wang, Z.J. (2006). An immersed interface method for simulating the interaction of a fluid with moving boundaries. *Journal of Computational Physics*. 216(2) pp.454–493.
- [178] Meyers, J.M., Fletcher, D.G. and Dubief, Y. (n.d). *Lift and Drag of an Airfoil*. ME 123: Mechanical Engineering Laboratory II: Fluids.
- [179] Wall, W.A. (1999). *Fluid–struktur interaktion mit stabilisierten finiten elementen*. Ph.D. Thesis, Institut für Baustatik und Baudynamik, Universität Stuttgart, Germany.
- [180] Dettmer, W.G. (2004). *Finite Element Modelling of Fluid Flow with Moving Free Surfaces and Interfaces Including Fluid-Solid Interaction*. Ph.D. Thesis, School of Engineering, University of Wales, Swansea.
- [181] Hübner, B., Wallhorn, E. and Dinkler, D. (2004). A Monolithic Approach to Fluid-Structure Interaction Using Space-Time Finite Elements. *Computer Methods in Applied Mechanics and Engineering*, 193(23), pp.2087-2104.
- [182] Matthies, H.G. and Steindorf, J. (2003). Partitioned strong coupling algorithms for fluid–structure interaction. *Computers & Structures*, 81(8), pp.805-812.
- [183] Olivier, M., Dumas, G. and Morissette, J. (2009). A fluid–structure interaction solver for nano-air-vehicle flapping wings. *In Proceedings of the 19th AIAA computational fluid dynamics conference, San Antonio, USA*. pp.1–15.
- [184] Razzaq, M. Hron, J. and Turek, S. (2010). Numerical Simulation of Laminar Incompressible Fluid-Structure Interaction for Elastic Material with Point *Advances in Mathematical Fluid Mechanics*, pp.451-472.
- [185] Wall, W.A. and Ramm, E. (1998). Fluid-structure interaction based upon a stabilized (ALE) finite element method. E and S Idelsohn. Eds. *In IV World Congress on Computational Mechanics (Vol. 170)*.

- [186] Wood, C. Gil, A., Hassan, O. and Bonet, J. (2010). Partitioned block-Gauss–Seidel coupling for dynamic fluid–structure interaction. *Computers & Structures*, 88(23), pp.1367-1382.
- [187] Wüchner, R., Kupzok, A., and Bletzinger, K.U. (2007). A framework for stabilized partitioned analysis of thin membrane-wind interaction. *International Journal for Numerical Methods in Fluids*, 54(6-8), pp.945-963.
- [188] Yvin, C. (2010). Partitioned fluid-structure interaction with open-source tools. 12ème Journées de l’Hydrodynamique, *Nantes, France*.
- [189] Walhorn, E., Hübner, B. and Dinkler, D. (2002). Space-time finite elements for fluid-structure interaction. *PAMM*, 1(1), pp.81-82
- [190] Kassiotis, C., Ibrahimbegovic, A., Niekamp, R. and Matthies, H.G. (2011b). Nonlinear fluid-structure interaction problem. Part II: Space discretization, implementation aspects, nested parallelisation and application examples. *Computational Mechanics*, 47(3), pp.335-357.
- [191] Von Scheven, M. (2009). *Effiziente Algorithmen für die Fluid-Struktur-Wechselwirkung*. Ph.D. Thesis, Institut für Baustatik und Baudynamik, Universität Stuttgart, Germany.
- [192] Taylor, R. (2013). *Finite Element Modelling of Three Dimensional Fluid-Structure Interaction*. Ph.D. Thesis, School of Engineering, Swansea University.
- [193] Hübner, B. (2003). *Simultane analyse von Bauwerks-Wind-Wechselwirkungen*. PhD Thesis. Institut für Statik, Technische Universität Braunschweig.
- [194] Forster, B. and Mollaert, M. (2004). *European Design Guide for Tensile Surface Structures: TensiNet*.
- [195] Michalski, A., Kermel, P.D., Haug, E., Lohner, R., Wüchner, R. and Bletzinger, K.U. (2011). Validation of the computational fluid–structure interaction simulation at real-scale tests of a flexible 29 m umbrella in natural wind flow. *Journal of Wind Engineering and Industrial Aerodynamics journal homepage*, 99(4), pp.400–413.

- [196] Kupzok, A.M. (2009). *Modelling the Interaction of Wind and Membrane Structures by Numerical Simulation*. PhD Thesis. Technical University of Munich, Germany.
- [197] Schweizerhof, K. and Ramm, E. (1984). Displacement dependent pressure loads in nonlinear finite element analyses. *Computers & Structures*, 18(6), pp.1099–1114.
- [198] Rank, E., Halfmann, A., Scholz, D., Glück, M., Breuer, M., Durst, F. Kaiser, U., Bergmann, D. and Wagner, S. (2005). Wind loads on lightweight structures: Numerical simulation and wind tunnel tests. *GAMM-Mitteilungen*, 28(1), pp.73–89.
- [199] Valdés, J.G., Oñate, E. and Miquel, J. (2016). Nonlinear analysis of orthotropic membrane and shell structures including fluid-structure interaction. *Monographs of the International Centre for Numerical Methods in Engineering (CIMNE)*.
- [200] Williams, C.J.K. (1997). The structural design of fabric structures to resist wind loading. *Research Paper-Health and Safety Executive London*, 38, pp.74-82.
- [201] Farhat, C. (2004). CFD-based nonlinear computational aeroelasticity. In *Encyclopedia of Computational Mechanics*. Stein, E., De Borst, R., and Hughes, T.J.R (eds.). pp.459–480.
- [202] Hartmann, S., Meister, A., Schäfer, M. and Turek, S. (2009). *International Workshop on Fluid-Structure Interaction Theory, Numerics and Applications*. Kassel: Kassel University press GmbH, Germany.
- [203] Juretić, F. (2015). cfMesh User Guide. *Creative Fields, Zagreb*.
- [204] Bletzinger, K.U., Wüchner, R. and Kupzok, A. (2006). Algorithmic treatment of shells and free form-membranes in FSI. *Fluid-structure interaction*, pp.336-355.
- [205] Hadipriono, F.C. (1985). Analysis of events in recent structural failures. *Journal of structural engineering*, 111(7), pp.1468-1481.
- [206] Supartono, F.X., Li, Z. and Wang, X. (2011). Membrane structure: A modern and aesthetic structural system. In *Seminar dan Pameran HAKI*.

- [207] Beccarelli, P. (2015). The design, analysis and construction of tensile fabric structures. In *Biaxial Testing for Fabrics and Foils*. Springer International Publishing. pp.9-33.
- [208] Drescher, H. (1947). *Model Testing Techniques. II. 6. Measurements of unsteady pressure*. AVA monographs D2. (Translation: Aero. Res. Council. Rep. no. 11, 391).

Appendices

Appendix 2.A: OpenFOAM Computational Pointers

The OpenFOAM structure of the linear algebraic solver starts by defining the classes that established from each solvers type of the algebraic matrix. These classified into three groups: solvers, preconditioners, and smoothers. Both preconditioners and smoothers are related to the differentiated fixed-point smoothers and then embedded them in the preconditioners framework. The linear algebraic solver's source codes in OpenFOAM located in “.../src/OpenFOAM /matrices/lduMatrix” for *solvers*, *preconditioners*, and *smoothers* sub-folders [74].

Firstly, *solvers* folder in OpenFOAM includes the following iterative solvers main codes

- *diagonalSolver* is a diagonal solver which used for symmetric and asymmetric matrices.
- *GAMG* named as a Generalized geometric-algebraic multi-grid solver or geometric agglomerated algebraic multi-grid solver. This solver applies the principle of creating a quick solution on a grid with the number of a small cell, mapping the solution onto a finer mesh; applying it as an initial presumption on the fine mesh to find an accurate solution.
- *ICC* defined as an incomplete Cholesky preconditioned conjugate gradient solver. This described for backward-completely and for its preference the *PCG* solver must utilise.
- *PCG* is a preconditioned conjugate gradient solver for symmetric LDU matrices.
- *PbiCG* is a preconditioned biconjugate gradient solver for asymmetric LDU matrices.
- *smoothSolver* is an iterative solver that is using smoother for both symmetric and asymmetric matrices depend on preconditioners.

The second folder is the *preconditioners* which consist of the different diagonal ILU implementations that denoted by

- `diagonalPreconditioner` is a diagonal preconditioner for both symmetric and asymmetric matrices. In spite of this preconditioner is not offer that faster propagation help through the grid, it is very good and easy for the first step.
- `DIC` and `DILU` are the diagonal-based incomplete Cholesky preconditioner for the symmetric and asymmetric matrices, respectively.
- `FDIC` is the faster version of the `DIC` preconditioner for the symmetric matrices where the preconditioned diagonal reciprocal and the upper coefficients of the matrix divided by the diagonal are both calculated and then stored.
- `GAMG` this is the Generalized geometric-algebraic multigrid preconditioner.
- `noPreconditioner` defines null preconditioner for symmetric and asymmetric matrices.

The last folder is the *smoothers*, and it contains

- `DIC` and `DILU` are simplified the diagonal-based incomplete Cholesky smoother for symmetric and asymmetric matrices, respectively.
- `DICGaussSeidel` and `DILUGaussSeidel` are combined smoother of `DIC` and `DILU`-Gauss-Seidel for both symmetric and asymmetric matrices where their smoothing is followed by Gauss-Seidel in order to ensure that if any “spikes” are created by their sweeps will be smoothed out.
- `DILU` also defined the LU smoother of a diagonal-based incomplete for asymmetric matrices.
- `GaussSeidel` is the Gauss-Seidel method that is used to solve both symmetric and asymmetric matrices. This method is considered as the improved method of the Jacobi method. It is defined on non-zero diagonal matrices, and its convergence is guaranteed by either diagonally predominant or symmetric and positive well-definite.

In OpenFOAM, those three classes which wrap the three categories are defined in the `lduMatrix` class as shown in the `lduMatrix.H` file listing in Figure 2.A.

```

class lduMatrix
{
    // private data

    //- LDU mesh reference
    const lduMesh& lduMesh_;

    //- Coefficients (not including interfaces)
    scalarField *lowerPtr_, *diagPtr_, *upperPtr_;
    ...
public:

    //- Abstract base-class for lduMatrix solvers
    class solver
    {
    protected:
    ...

    //- Abstract base-class for lduMatrix smoothers
    class smoother
    {
    protected:
    ...

    //- Abstract base-class for lduMatrix preconditioners
    class preconditioner
    {
    protected:
    ...

```

Figure 2.A: lduMatrix.H file in OpenFOAM for solver, smoother, and preconditioner classes

Appendix 3.A: Velocity (U) Boundary Conditions of the Circular Cylinder Case

```
/*-----*- C++ -*-----*\
|=====|
|\ / Field | foam-extend: Open Source CFD |
|\ / Operation | Version: 3.0 |
|\ / And | Web: http://www.extend-project.de |
|\ Manipulation |
\*-----*/

FoamFile
{
    version 2.0;
    format ascii;
    class volVectorField;
    location "0";
    object U;
}

// *****

dimensions [0 1 -1 0 0 0];

internalField uniform (0 0 0);

boundaryField
{
    bottom
    {
        type empty;
    }

    outlet
    {
        type pressureInletOutletVelocity;
        value uniform (0 0 0);
    }
}
```

```
top
{
  type      empty;
}

inlet
{
  type      fixedValue;
  value     uniform (1 0 0);
}

walls
{
  type      slip;
}

cylinder
{
  type      fixedValue;
  value     uniform (0 0 0);
}
}
// ***** //
```

Appendix 3.B: Pressure (p) Boundary Conditions of the Circular Cylinder Case

```
/*-----*- C++ -*-----*\
|=====|
|\ / Field | foam-extend: Open Source CFD |
| \ / Operation | Version: 3.0 |
| \ / And | Web: http://www.extend-project.de |
| \ / Manipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object p;
}
// ***** //

dimensions [0 2 -2 0 0 0];

internalField uniform 0;

boundaryField
{
    bottom
    {
        type empty;
    }

    outlet
    {
        type fixedValue;
        value uniform 0;
    }
}
```

```
top
{
  type      empty;
}

inlet
{
  type      zeroGradient;
}

walls
{
  type      slip;
}

cylinder
{
  type      zeroGradient;
}
}

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


Appendix 3.C: controlDict File of the Circular Cylinder Case

```
/*-----*- C++ -*-----*\
|=====|
|\ / Field | OpenFOAM: The Open Source CFD Toolbox |
|\ / Operation | Version: 1.6 |
|\ / And | Web: www.OpenFOAM.org |
|\ / Manipulation |
\*-----*/

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "system";
    object controlDict;
}

// *****

application pimpleFoam;

startFrom latestTime;

startTime 0;

stopAt endTime;

endTime 100;

deltaT 0.001;

writeControl adjustableRunTime;

writeInterval 1.0;

purgeWrite 0;

writeFormat binary;

writePrecision 12;

writeCompression compressed;
```

```

timeFormat    general;
timePrecision 12;
runTimeModifiable yes;
adjustTimeStep yes;
maxCo        0.2;
maxDeltaT    1.0;

functions
{
    forces
    {
        type    forceCoeffs;
        functionObjectLibs ( "libforces.so" );
        outputControl timeStep;
        outputInterval 1;
        patches
        (
            cylinder
        );
        directForceDensity no;

        pName    p;
        UName    U;
        rhoName  rhoInf;
        log      true;
        rhoInf  1000;
        CofR    (0.160 0 0); // origin for moment calculations
    }
}

```

```

liftDir ( 0 1 0 ); // lift direction (Parallel to U_inf)
dragDir ( 1 0 0 ); // drag direction (Normal to U_inf)
pitchAxis ( 0 0 0 ); // rotational moment axis
magUInf 1.0; // relative velocity between cylinder and fluid
lRef 1.0; // cylinder length (radius for cylinder) --> reference length
Aref 1.0; // reference area
}

```

```

fieldAverage1
{
    type fieldAverage;
    functionObjectLibs ("libfieldFunctionObjects.so");
    enabled true;
    outputControl outputTime;
    fields
    (
        U
        {
            mean on;
            prime2Mean on;
            base time;
        }

        p
        {
            mean on;
            prime2Mean on;
        }
    )
}

```

```
        base    time;
    }
);
}
}
```

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

Appendix 3.D: fvScheme File of the Circular Cylinder Case

```
/*-----*- C++ -*-----*\
|=====|
|\ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O p e r a t i o n | Version: 2.2.2 |
| \ \ / A n d | Web: www.OpenFOAM.org |
| \ \ M a n i p u l a t i o n |
\*-----*/

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    object fvSchemes;
}

// ***** //

ddtSchemes
{
    default Euler;
}

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

divSchemes
{
    default none;
    div(phi,U) Gauss linearUpwind grad(U);
}
```

```
div(phi,k) Gauss limitedLinear 1;  
div(phi,omega) Gauss limitedLinear 1;  
div((nuEff*dev(T(grad(U)))) Gauss linear;  
}
```

```
laplacianSchemes
```

```
{  
  default Gauss linear limited corrected 0.5;  
}
```

```
interpolationSchemes
```

```
{  
  default linear;  
}
```

```
snGradSchemes
```

```
{  
  default corrected;  
}
```

```
fluxRequired
```

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

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

Appendix 3.E: fvSolution File of the Circular Cylinder Case

```
/*-----*- C++ -*-----*\
|=====|
|\ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O p e r a t i o n | Version: 2.2.2 |
| \ \ / A n d | Web: www.OpenFOAM.org |
| \ \ M a n i p u l a t i o n |
\*-----*/

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    object fvSolution;
}

// *****

solvers
{
    pcorr
    {
        solver GAMG;
        tolerance 0.02;
        relTol 0;
        smoother GaussSeidel;
        nPreSweeps 0;
        nPostSweeps 2;
        cacheAgglomeration on;
        agglomerator faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels 1;
    }
}
```

```

p
{
  $pcorr
  tolerance 1e-7;
  relTol 0.01;
}

pFinal
{
  $p;
  tolerance 1e-7;
  relTol 0;
}

"(U|k|omega)"
{
  solver PBiCG;
  preconditioner DILU;
  tolerance 1e-06;
  relTol 0.1;
}

"(U|k|omega)Final"
{
  $U;
  tolerance 1e-06;
  relTol 0;
}

cellDisplacement
{
  solver GAMG;
  tolerance 1e-5;
  relTol 0;
  smoother GaussSeidel;
}

```



```

    cacheAgglomeration true;
    nCellsInCoarsestLevel 10;
    agglomerator faceAreaPair;
    mergeLevels 1;
}
}

```

PIMPLE

```

{
    correctPhi yes;
    nOuterCorrectors 2;
    nCorrectors 1;
    nNonOrthogonalCorrectors 0;
}

```

relaxationFactors

```

{
    fields
    {
        p 0.3;
    }
    equations
    {
        "(U|k|omega)" 0.7;
        "(U|k|omega)Final" 1.0;
    }
}

```

cache

```

{
    grad(U);
}

```

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

Appendix 3.F: dynamicMeshDict File

```

/*----- C++ -----*\
|=====|
|\ / Field | OpenFOAM: The Open Source CFD Toolbox |
|\ / Operation | Version: 2.2.2 |
|\ / And | Web: www.OpenFOAM.org |
|\ / Manipulation |
\*-----*/
FoamFile
{
  version 2.0;
  format ascii;
  class dictionary;
  object motionProperties;
}
// *****

```

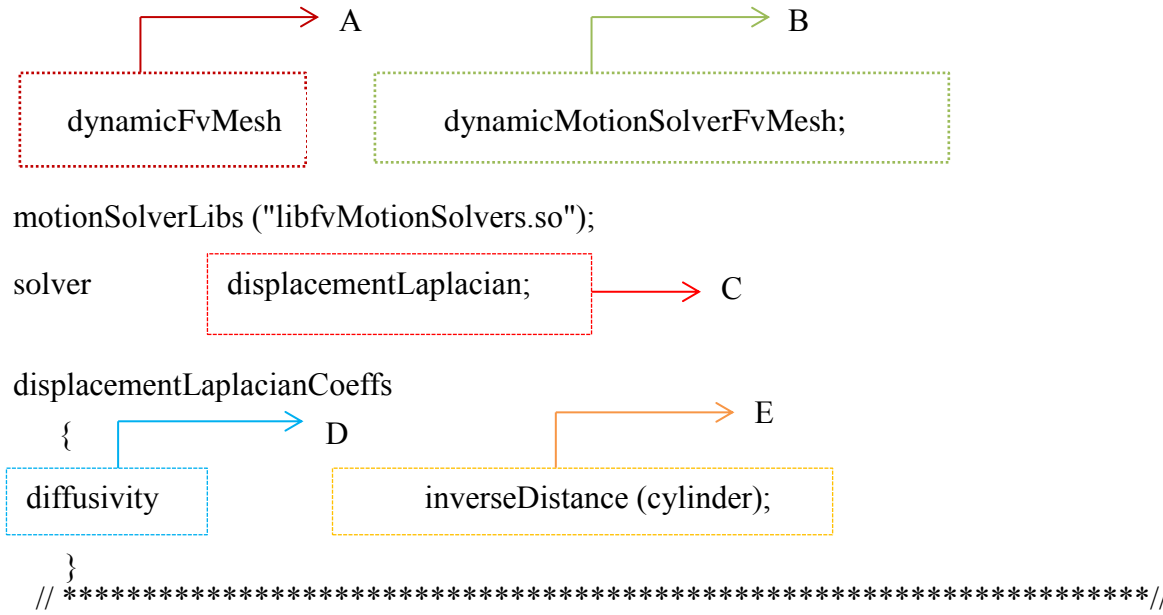


Table 3.F: dynamicMeshDict content explanation

Symbol	Definition
A	Automatic mesh motion where the mesh topology does not change
B	The simplest mesh motion solver type where the interval points motion is solved by using both the boundary conditions and the diffusivity models
C	The cell motion equations are solved based on the Laplacian on the cell displacement and on the diffusivity. The cell displacement is defined in the <code>pointDisplacement</code> file located in the <code>time</code> directories folder
D	One or more boundaries are specified and the diffusivity is based on the distance inverse from that boundary
E	Determines the way of the points movement when the cell equation is solved for each time-step

Appendix 3.G: Free Vibration Case - Scenario 1

```
/*-----*- C++ -*-----*\
|=====|
|\ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
|\ \ / O p e r a t i o n | Version: 2.2.2 |
|\ \ / A n d | Web: www.OpenFOAM.org |
|\ \ M a n i p u l a t i o n |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class pointVectorField;
    location "0.01";
    object pointDisplacement;
}
// *****

dimensions [0 1 0 0 0 0];

internalField uniform (0 0 0);

boundaryField
{
    cylinder
    {
        type sixDoFRigidBodyDisplacement;

        centreOfMass (4 0 0.5);
        momentOfInertia (1.4539 1.4539 1.24625);
        mass 9.97;
        orientation
        (
```

```

1 0 0
0 1 0
0 0 1
);

velocity (0 0 0);
acceleration (0 0 0);
angularMomentum (0 0 0);
torque (0 0 0);
gravity (0 -9.81 0);
rhoName rhoInf;
rhoInf 1024;
report on;

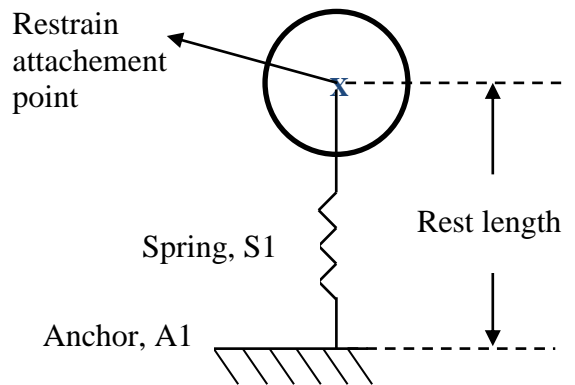
restraints
{
  S1
  {
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {
      anchor (3.5 0 0.5); //A1
      refAttachmentPt (4 0 0.5);
      stiffness 4;
      damping 2;
      restLength 0.5;
    }
  }

  S2
  {
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs

```



```

    {
        anchor      (4 0.5 0.5); //A2
        refAttachmentPt (4 0 0.5);
        stiffness    4;
        damping      2;
        restLength   0.5;
    }
}

S3
{
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {
        anchor      (4.5 0 0.5); //A3
        refAttachmentPt (4 0 0.5);
        stiffness    4;
        damping      2;
        restLength   0.5;
    }
}

S4
{
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {
        anchor      (4 -0.5 0.5); //A4
        refAttachmentPt (4 0 0.5);
        stiffness    4;
        damping      2;
        restLength   0.5;
    }
}

```

```

    }

}

constraints
{
    maxIterations    500;

    DontMoveOverZ
    {
        sixDoFRigidBodyMotionConstraint fixedAxis;
        tolerance    1e-06;
        relaxationFactor 0.7;
        fixedAxisCoeffs
        {
            axis      ( 0 0 1 );
        }
    }
}

value    uniform (0 0 0);
}

top
{
    type    empty;
}

bottom
{
    type    empty;
}

"*)"
{

```

```
    type    fixedValue;  
    value   uniform (0 0 0);  
  }  
}
```

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

Appendix 3.H: Free Vibration Case - Scenario 2

```
/*-----*- C++ -*-----*\
|=====|
|\ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
|\ \ / O p e r a t i o n | Version: 2.2.2 |
|\ \ / A n d | Web: www.OpenFOAM.org |
|\ \ M a n i p u l a t i o n |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class pointVectorField;
    location "0.01";
    object pointDisplacement;
}
// ***** //

dimensions [0 1 0 0 0 0];

internalField uniform (0 0 0);

boundaryField
{
    cylinder
    {
        type sixDoFRigidBodyDisplacement;

        centreOfMass (4 0 0.5);
        momentOfInertia (1.4539 1.4539 1.24625);
        mass 9.97;
        orientation
        (
```



```

1 0 0
0 1 0
0 0 1
);

velocity    (0 0 0);
acceleration (0 0 0);
angularMomentum (0 0 0);
torque      (0 0 0);
gravity     (0 -9.81 0);
rhoName     rhoInf;
rhoInf      1024;
report      on;
restraints
{
  S1
  {
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {
      anchor      (3.5 0 0.5); //A1
      refAttachmentPt (4 0 0.5);
      stiffness    4;
      damping      2;
      restLength   0.5;
    }
  }

  S2
  {
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {

```

```

        anchor      (4 0.5 0.5); //A2
        refAttachmentPt (4 0 0.5);
        stiffness    4;
        damping      2;
        restLength   0.5;
    }
}

S3
{
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {
        anchor      (4.5 0 0.5); //A3
        refAttachmentPt (4 0 0.5);
        stiffness    4;
        damping      2;
        restLength   0.5;
    }
}

S4
{
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {
        anchor      (4 -0.5 0.5); //A4
        refAttachmentPt (4 0 0.5);
        stiffness    4;
        damping      2;
        restLength   0.5;
    }
}

```

```

}

constraints
{
    maxIterations    500;

    DontMoveOverZ
    {
        sixDoFRigidBodyMotionConstraint fixedAxis;
        tolerance    1e-06;
        relaxationFactor 0.7;
        fixedAxisCoeffs
        {
            axis      (0 0 1);
        }
    }

    moveOnlyAlongY
    {
        sixDoFRigidBodyMotionConstraint fixedLine;
        tolerance    1e-9;
        relaxationFactor 0.7;
        fixedLineCoeffs
        {
            refPoint  (4 0 0.5);
            direction (0 1 0);
        }
    }

}

value    uniform (0 0 0);
}

```

```
top
{
  type      empty;
}

bottom
{
  type      empty;
}

"."
{
  type      fixedValue;
  value     uniform (0 0 0);
}
}

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

Appendix 3.I: Free Vibration Case - Scenario 3

```
/*-----*- C++ -*-----*\
|=====|
|\ \ / F ield | OpenFOAM: The Open Source CFD Toolbox |
| \ \ / O peration | Version: 2.2.2 |
| \ \ / A nd | Web: www.OpenFOAM.org |
| \ \ M anipulation | |
\*-----*/

FoamFile
{
    version 2.0;
    format ascii;
    class pointVectorField;
    location "0.01";
    object pointDisplacement;
}
// ***** //

dimensions [0 1 0 0 0 0];

internalField uniform (0 0 0);

boundaryField
{
    cylinder
    {
        type sixDoFRigidBodyDisplacement;

        centreOfMass (4 0 0.5);
        momentOfInertia (1.4539 1.4539 1.24625);
        mass 9.97;
        orientation
        (
```

```

    1 0 0
    0 1 0
    0 0 1
);

velocity    (0 0 0);
acceleration (0 0 0);
angularMomentum (0 0 0);
torque      (0 0 0);
gravity     (0 -9.81 0);
rhoName     rhoInf;
rhoInf      1024;
report      on;

restraints
{
  S1
  {
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {
      anchor      (3.5 0 0.5); //A1
      refAttachmentPt (4 0 0.5);
      stiffness    4;
      damping      0;
      restLength   0.5;
    }
  }

  S2
  {
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs

```

```

    {
        anchor      (4 0.5 0.5); //A2
        refAttachmentPt (4 0 0.5);
        stiffness    4;
        damping      0;
        restLength   0.5;
    }
}

S3
{
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {
        anchor      (4.5 0 0.5); //A3
        refAttachmentPt (4 0 0.5);
        stiffness    0;
        damping      2;
        restLength   0.5;
    }
}

S4
{
    sixDoFRigidBodyMotionRestraint linearSpring;

    linearSpringCoeffs
    {
        anchor      (4 -0.5 0.5); //A4
        refAttachmentPt (4 0 0.5);
        stiffness    0;
        damping      2;
        restLength   0.5;
    }
}

```

```

    }

}

constraints
{
    maxIterations    500;

    DontMoveOverZ
    {
        sixDoFRigidBodyMotionConstraint fixedAxis;
        tolerance    1e-06;
        relaxationFactor 0.7;
        fixedAxisCoeffs
        {
            axis      ( 0 0 1 );
        }
    }
}

value    uniform (0 0 0);
}

top
{
    type    empty;
}

bottom
{
    type    empty;
}

"*)"
{

```



```
    type    fixedValue;  
    value   uniform (0 0 0);  
  }  
}
```

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

Appendix 3.J: Forced Vibration Case

```
/*-----*- C++ -*-----*\
|=====|
|\ / Field | foam-extend: Open Source CFD |
|\ / Operation | Version: 3.0 |
|\ / And | Web: http://www.extend-project.de |
|\ / Manipulation |
\*-----*/
```

FoamFile

```
{
version 2.0;
format ascii;
class pointVectorField;
object pointDisplacement;

// ***** //

dimensions [0 1 0 0 0 0];

internalField uniform (0 0 0);

boundaryField
{
inlet
{
type fixedValue;
value uniform (0 0 0);
}
outlet
{
type fixedValue;
value uniform (0 0 0);
}
```

```

}
  top
  {
    type empty;
  }
  bottom
  {
    type empty;
  }
  walls
  {
    type slip;
  }

  cylinder
  {
    type oscillatingDisplacement;
    amplitude (0 0.25 0);
    omega 1.04929; // 2*Pi*f0 (f0 = 0.167 Hz)
    value uniform (0 0 0);
  }
}

// ***** /

```

Appendix 4.A: Mesh Generation- blockMeshDict of the 3D Elastic Cantilever Plate Attached to a Solid Block Case

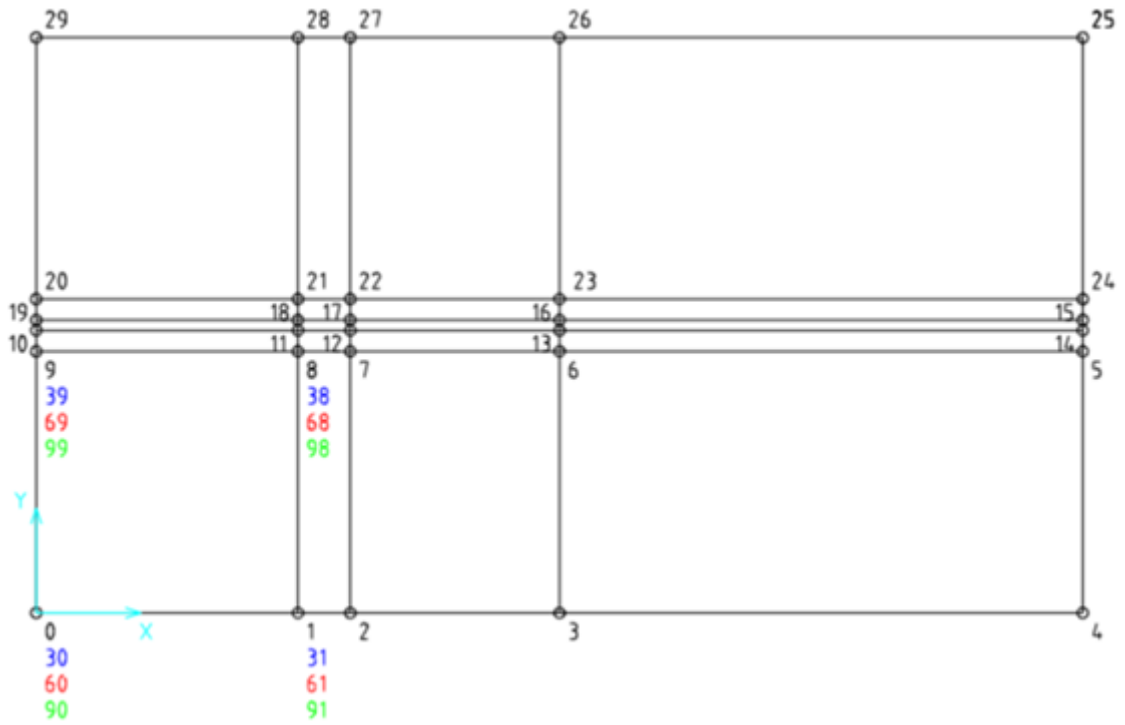


Figure 4.A: 3D elastic plate case points in blockMeshDict dictionary

```

/*-----*- C++ -*-----*\
|=====|
|\ / Field | OpenFOAM: The Open Source CFD Toolbox |
| \ / Operation | Version: 2.2.2 |
| \ / And | Web: www.OpenFOAM.org |
| \ / Manipulation |
\*-----*/

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    object blockMeshDict;
}
// ***** //

convertToMeters 0.01;

vertices
(
// First level
(0 0 0) //0
(5 0 0) //1
(6 0 0) //2
(10 0 0) //3
(20 0 0) //4
(20 5 0) //5
(10 5 0) //6
(6 5 0) //7
(5 5 0) //8
(0 5 0) //9
(0 5.47 0) //10
(5 5.47 0) //11
(6 5.47 0) //12

```

(10 5.47 0) //13
(20 5.47 0) //14
(20 5.53 0) //15
(10 5.53 0) //16
(6 5.53 0) //17
(5 5.53 0) //18
(0 5.53 0) //19
(0 6 0) //20
(5 6 0) //21
(6 6 0) //22
(10 6 0) //23
(20 6 0) //24
(20 11 0) //25
(10 11 0) //26
(6 11 0) //27
(5 11 0) //28
(0 11 0) //29

// Second level

(0 0 4)
(5 0 4)
(6 0 4)
(10 0 4)
(20 0 4)
(20 5 4)
(10 5 4)
(6 5 4)
(5 5 4)
(0 5 4)
(0 5.47 4)
(5 5.47 4)
(6 5.47 4)
(10 5.47 4)
(20 5.47 4)
(20 5.53 4)
(10 5.53 4)

(6 5.53 4)
(5 5.53 4)
(0 5.53 4)
(0 6 4)
(5 6 4)
(6 6 4)
(10 6 4)
(20 6 4)
(20 11 4)
(10 11 4)
(6 11 4)
(5 11 4)
(0 11 4)

// Third level

(0 0 7)
(5 0 7)
(6 0 7)
(10 0 7)
(20 0 7)
(20 5 7)
(10 5 7)
(6 5 7)
(5 5 7)
(0 5 7)
(0 5.47 7)
(5 5.47 7)
(6 5.47 7)
(10 5.47 7)
(20 5.47 7)
(20 5.53 7)
(10 5.53 7)
(6 5.53 7)
(5 5.53 7)
(0 5.53 7)
(0 6 7)

(5 6 7)

(6 6 7)

(10 6 7)

(20 6 7)

(20 11 7)

(10 11 7)

(6 11 7)

(5 11 7)

(0 11 7)

// Fourth level

(0 0 11)

(5 0 11)

(6 0 11)

(10 0 11)

(20 0 11)

(20 5 11)

(10 5 11)

(6 5 11)

(5 5 11)

(0 5 11)

(0 5.47 11)

(5 5.47 11)

(6 5.47 11)

(10 5.47 11)

(20 5.47 11)

(20 5.53 11)

(10 5.53 11)

(6 5.53 11)

(5 5.53 11)

(0 5.53 11)

(0 6 11)

(5 6 11)

(6 6 11)

(10 6 11)

(20 6 11)


```

(20 11 11)
(10 11 11)
(6 11 11)
(5 11 11)
(0 11 11)
);
blocks
(
// Bottom blocks
hex (0 1 8 9 30 31 38 39) (26 25 23) simpleGrading (0.16 0.154 0.2) //0
hex (1 2 7 8 31 32 37 38) (15 25 23) simpleGrading (1 0.154 0.2) //1
hex (2 3 6 7 32 33 36 37) (60 25 23) simpleGrading (1 0.154 0.2) //2
hex (3 4 5 6 33 34 35 36) (34 25 23) simpleGrading (11.4 0.154 0.2) //3
hex (6 5 14 13 36 35 44 43) (34 7 23) simpleGrading (11.4 1 0.2) //4
hex (7 6 13 12 37 36 43 42) (60 7 23) simpleGrading (1 1 0.2) //5
hex (8 7 12 11 38 37 42 41) (15 7 23) simpleGrading (1 1 0.2) //6
hex (9 8 11 10 39 38 41 40) (26 7 23) simpleGrading (0.16 1 0.2) //7
hex (10 11 18 19 40 41 48 49) (26 1 23) simpleGrading (0.16 1 0.2) //8
hex (11 12 17 18 41 42 47 48) (15 1 23) simpleGrading (1 1 0.2) //9
hex (12 13 16 17 42 43 46 47) (60 1 23) simpleGrading (1 1 0.2) //10
hex (13 14 15 16 43 44 45 46) (34 1 23) simpleGrading (11.4 1 0.2) //11
hex (16 15 24 23 46 45 54 53) (34 7 23) simpleGrading (11.4 1 0.2) //12
hex (17 16 23 22 47 46 53 52) (60 7 23) simpleGrading (1 1 0.2) //13
hex (18 17 22 21 48 47 52 51) (15 7 23) simpleGrading (1 1 0.2) //14
hex (19 18 21 20 49 48 51 50) (26 7 23) simpleGrading (0.16 1 0.2) //15
hex (20 21 28 29 50 51 58 59) (26 25 23) simpleGrading (0.16 6.5 0.2) //16
hex (21 22 27 28 51 52 57 58) (15 25 23) simpleGrading (1 6.5 0.2) //17
hex (22 23 26 27 52 53 56 57) (60 25 23) simpleGrading (1 6.5 0.2) //18
hex (23 24 25 26 53 54 55 56) (34 25 23) simpleGrading (11.4 6.5 0.2) //19
// Middle blocks
hex (30 31 38 39 60 61 68 69) (26 25 45) simpleGrading (0.16 0.154 1)
hex (31 32 37 38 61 62 67 68) (15 25 45) simpleGrading (1 0.154 1)
hex (32 33 36 37 62 63 66 67) (60 25 45) simpleGrading (1 0.154 1)
hex (33 34 35 36 63 64 65 66) (34 25 45) simpleGrading (11.4 0.154 1)
hex (36 35 44 43 66 65 74 73) (34 7 45) simpleGrading (11.4 1 1)

```

```

hex (37 36 43 42 67 66 73 72) (60 7 45) simpleGrading (1 1 1)
hex (39 38 41 40 69 68 71 70) (26 7 45) simpleGrading (0.16 1 1)
hex (40 41 48 49 70 71 78 79) (26 1 45) simpleGrading (0.16 1 1)
hex (43 44 45 46 73 74 75 76) (34 1 45) simpleGrading (11.4 1 1)
hex (46 45 54 53 76 75 84 83) (34 7 45) simpleGrading (11.4 1 1)
hex (47 46 53 52 77 76 83 82) (60 7 45) simpleGrading (1 1 1)
hex (49 48 51 50 79 78 81 80) (26 7 45) simpleGrading (0.16 1 1)
hex (50 51 58 59 80 81 88 89) (26 25 45) simpleGrading (0.16 6.5 1)
hex (51 52 57 58 81 82 87 88) (15 25 45) simpleGrading (1 6.5 1)
hex (52 53 56 57 82 83 86 87) (60 25 45) simpleGrading (1 6.5 1)
hex (53 54 55 56 83 84 85 86) (34 25 45) simpleGrading (11.4 6.5 1)
// Top blocks
hex (60 61 68 69 90 91 98 99) (26 25 23) simpleGrading (0.16 0.154 5)
hex (61 62 67 68 91 92 97 98) (15 25 23) simpleGrading (1 0.154 5)
hex (62 63 66 67 92 93 96 97) (60 25 23) simpleGrading (1 0.154 5)
hex (63 64 65 66 93 94 95 96) (34 25 23) simpleGrading (11.4 0.15 5)
hex (66 65 74 73 96 95 104 103) (34 7 23) simpleGrading (11.4 1 5)
hex (67 66 73 72 97 96 103 102) (60 7 23) simpleGrading (1 1 5)
hex (68 67 72 71 98 97 102 101) (15 7 23) simpleGrading (1 1 5)
hex (69 68 71 70 99 98 101 100) (26 7 23) simpleGrading (0.16 1 5)
hex (70 71 78 79 100 101 108 109) (26 1 23) simpleGrading (0.16 1 5)
hex (71 72 77 78 101 102 107 108) (15 1 23) simpleGrading (1 1 5)
hex (72 73 76 77 102 103 106 107) (60 1 23) simpleGrading (1 1 5)
hex (73 74 75 76 103 104 105 106) (34 1 23) simpleGrading (11.4 1 5)
hex (76 75 84 83 106 105 114 113) (34 7 23) simpleGrading (11.4 1 5)
hex (77 76 83 82 107 106 113 112) (60 7 23) simpleGrading (1 1 5)
hex (78 77 82 81 108 107 112 111) (15 7 23) simpleGrading (1 1 5)
hex (79 78 81 80 109 108 111 110) (26 7 23) simpleGrading (0.16 1 5)
hex (80 81 88 89 110 111 118 119) (26 25 23) simpleGrading (0.16 6.5 5)
hex (81 82 87 88 111 112 117 118) (15 25 23) simpleGrading (1 6.5 5)
hex (82 83 86 87 112 113 116 117) (60 25 23) simpleGrading (1 6.5 5)
hex (83 84 85 86 113 114 115 116) (34 25 23) simpleGrading (11.4 6.5 5)
);

```

edges

(
);

patches

```
(  
  patch inlet  
  (  
    (9 0 30 39)  
    (39 30 60 69)  
    (69 60 90 99)  
    (10 9 39 40)  
    (40 39 69 70)  
    (70 69 99 100)  
    (19 10 40 49)  
    (49 40 70 79)  
    (79 70 100 109)  
    (20 19 49 50)  
    (50 49 79 80)  
    (80 79 109 110)  
    (29 20 50 59)  
    (59 50 80 89)  
    (89 80 110 119)  
  )  
  patch outlet  
  (  
    (4 5 35 34)  
    (34 35 65 64)  
    (64 65 95 94)  
    (5 14 44 35)  
    (35 44 74 65)  
    (65 74 104 95)  
    (14 15 45 44)  
    (44 45 75 74)  
    (74 75 105 104)  
    (15 24 54 45)  
  )  
)
```

(45 54 84 75)
(75 84 114 105)
(24 25 55 54)
(54 55 85 84)
(84 85 115 114)

)
patch back

(
(28 29 59 58)
(58 59 89 88)
(88 89 119 118)
(27 28 58 57)
(57 58 88 87)
(87 88 118 117)
(26 27 57 56)
(56 57 87 86)
(86 87 117 116)
(25 26 56 55)
(55 56 86 85)
(85 86 116 115)

)
patch front

(
(0 1 31 30)
(30 31 61 60)
(60 61 91 90)
(1 2 32 31)
(31 32 62 61)
(61 62 92 91)
(2 3 33 32)
(32 33 63 62)
(62 63 93 92)
(3 4 34 33)
(33 34 64 63)
(63 64 94 93)

)

patch top

(

(90 91 98 99)

(91 92 97 98)

(92 93 96 97)

(93 94 95 96)

(96 95 104 103)

(97 96 103 102)

(98 97 102 101)

(99 98 101 100)

(100 101 108 109)

(101 102 107 108)

(102 103 106 107)

(103 104 105 106)

(106 105 114 113)

(107 106 113 112)

(108 107 112 111)

(109 108 111 110)

(110 111 118 119)

(111 112 117 118)

(112 113 116 117)

(113 114 115 116)

)

patch bottom

(

(0 1 8 9)

(1 2 7 8)

(2 3 6 7)

(3 4 5 6)

(6 5 14 13)

(7 6 13 12)

(8 7 12 11)

(9 8 11 10)

(10 11 18 19)

(11 12 17 18)
(12 13 16 17)
(13 14 15 16)
(16 15 24 23)
(17 16 23 22)
(18 17 22 21)
(19 18 21 20)
(20 21 28 29)
(21 22 27 28)
(22 23 26 27)
(23 24 25 26)

)
wall plate

(

(43 42 72 73)
(46 43 73 76)
(47 46 76 77)
(42 43 46 47)
(73 72 77 76)

)
wall cylinder //block

(

(38 41 71 68)
(41 48 78 71)
(48 51 81 78)
(37 38 68 67)
(42 37 67 72)
(52 47 77 82)
(51 52 82 81)
(38 37 42 41)
(41 42 47 48)
(48 47 52 51)
(67 68 71 72)

```
(72 71 78 77)
(77 78 81 82)
)
);
```

```
mergePatchPairs
(
);
```

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

Appendix 4.B: Implementation structure of 3D Elastic Cantilever Plate Attached to a solid Block

In order to implement that case in foam-extend 4.0, three different folders will be in the main case folder and discussed as follows:

- initialize
 - This is the primary case that runs for 2 seconds with coupled off.
 - It uses the boundary condition oscillatingInlet (**Appendix 4.C**), which has been custom made to use the inlet function that is defined in page 21 of the article [190].
 - Essentially this case does time = -2 to 0 seconds simulation that the article states, with the exception that the case was configured to run from 0 to 2 seconds, to avoid any possible errors associated with negative times, given that it's rare to do these kinds of simulations.
 - The folder "0.org" is only there as a backup, in case something happens to the time directory "0". This to say that the folder "0" is not deleted by the Allrun or Allclean scripts.
- oscillatingInlet
 - This has the source code library used for the boundary condition oscillatingInlet (see **Appendix 4.C**).
 - It's built automatically by the script Allrun that is in the previous case's subfolder fluid.
 - It uses the function they mention in [190], but it had to adjust to the different start time, so the function changed the term "(t+1)" to "(t-1)".
- solidMotion
 - This is the second case, which will run for 12 seconds, with coupled on.
 - The fluid and solid properties were set in this case folder.

- The inlet boundary condition is fixed to 1.0m/s using changeDictionary (**Appendix 4.D**) and the file system/changeDictionaryDict, but the velocity U and pressure p fields are first mapped from the time-step 2 from the first case (initialize), onto 0 on this case as presented in **Appendix 4.E**.
- The folder 0 is deleted by the Allclean and Allrun scripts for this case (**Appendix 4.F** and **Appendix 4.G**, respectively), because of how the case is set-up. Therefore, if ever need to change the initial boundary conditions, must change the files inside the folder 0.org.
- The file fluid/system/controlDict was changed the monitoring point. It now has three monitoring points, two at the two ends tip of the plate and another at the middle-end of the plate (**Appendix 4.H**).
- In both cases, the Allrun and Allclean scripts have been changed to properly handle each case.

Appendix 4.C: oscillatingInlet of the 3D Elastic Cantilever Plate Attached to a Solid Block Case

```
/*-----*\
\\ / Field | foam-extend: Open Source CFD
\\ / Operation |
\\ / And | For copyright notice see file Copyright
\\ Manipulation |
```

License

This file is part of foam-extend.

foam-extend is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

foam-extend is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with foam-extend. If not, see <<http://www.gnu.org/licenses/>>.

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

```
#include "oscillatingInletFvPatchField.H"
#include "addToRunTimeSelectionTable.H"
#include "fvPatchFieldMapper.H"
#include "volFields.H"
#include "surfaceFields.H
```

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

```
namespace Foam
```

```
{
```

```
    scalar oscillatingInletFvPatchVectorField::calculate()
```

```

{
  //Note: the formula in the paper is (t+1), but it's meant to be use for t = [-2,0]
  // We need t = [0,2], there for we substract the 2 seconds and get (t-1)
  return
    0.5
  * (
    sin(mathematicalConstant::pi*(this->db().time().value()-1.0)/2.0)
    + 1.0
  )
  // ***** /

```

Appendix 4.D: changeDictionaryDict of the 3D Elastic Cantilever Plate Attached to a Solid Block Case

```

/*-----*- C++ -*-----*\
|          |          |
|\ \ / Field | foam-extend: Open Source CFD |
| \ \ / Operation | Version: 3.1 |
| \ \ / And | Web: http://www.extend-project.de |
| \ \ Manipulation |
\*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    object changeDictionaryDict;
}
// ***** //

dictionaryReplacement
{
    U
    {
        boundaryField
        {
            inlet
            {
                type fixedValue;
                value uniform (1.0 0 0);
            }
        }
    }
}
// ***** //

```

Appendix 4.E: mappingFields File

```
/*-----*\
|=====|
|\ \ / Field | foam-extend: Open Source CFD |
| \ \ / Operation | Version: 4.0 |
| \ \ / And | Web: http://www.foam-extend.org |
| \ \ / Manipulation | For copyright notice see file Copyright |
\*-----*/
```

Build : 4.0-6de4e266aa6e

Exec : mapFields -consistent ../../initialize/fluid -sourceTime 2.0

Date : Feb 14 2017

Time : 11:07:34

Host : pnnode03

PID : 4923

CtrlDict....."/eng/cvcluster/egalmanthm/foam/egalmanthm-4.0/FluidSolidInteraction/run/fsiFoam/1stFeb/solidMotion11/fluid/system/controlDict"

Case.../eng/cvcluster/egalmanthm/foam/egalmanthm-4.0/FluidSolidInteraction/run/fsiFoam/1stFeb/solidMotion11/fluid

nProcs : 1

SigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).

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

Source: "../../initialize" "fluid"

Target: "/eng/cvcluster/egalmanthm/foam/egalmanthm-4.0/FluidSolidInteraction/run/fsiFoam/1stFeb/solidMotion11" "fluid"

Create databases as time

Source time: 2

Target time: 0

Create meshes

Source mesh size: 785700 Target mesh size: 785700

Consistently creating and mapping fields for time 2

interpolating p

interpolating U

interpolating U_0

interpolating cellMotionU

End

Appendix 4.F: Allclean Script File

```
#!/bin/sh

# Source tutorial clean functions
. $WM_PROJECT_DIR/bin/tools/CleanFunctions

cleanCase
\rm -f constant/polyMesh/boundary
\rm -rf history

\rm -f constant/solid/polyMesh/boundary
\rm -rf constant/solid/polyMesh/[c-z]*
\rm -rf ../solid/VTK
\rm -f *.ps
\rm -f *.pdf

\rm -rf 0

wclean libso ../hronTurekReport
wclean libso ../pointHistoryMod
```

Appendix 4.G: Allrun Script File

```
#!/bin/sh

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

# Get application name
application=`getApplication`

rm -rf 0
cp -r 0.org 0

ln -s ../../solid/0 0/solid

runApplication -l log.blockMesh.solid blockMesh -region solid
runApplication -l log.setSet.solid setSet -case ../solid -batch ../solid/setBatch
runApplication -l log.setToZones.solid setsToZones -case ../solid -noFlipMap

runApplication blockMesh
runApplication setSet -batch setBatch
runApplication setsToZones -noFlipMap

# Build hronTurekReport function object
#wmake libso ../hronTurekReport

# Build the modified point monitorin library
# wmake libso ../../pointHistoryMod

runApplication mapFields -consistent ../../initialize/fluid -sourceTime 2.0
cp 0.org/pointMotionU 0/
rm 0/U_0* 0/cellMotionU*
runApplication changeDictionary
runApplication $application

# ----- end-of-file
```


Appendix 4.H: controlDict File

```
/*-----*- C++ -*-----*\
|=====|
|\ \ / F ield | foam-extend: Open Source CFD |
| \ \ / O peration | Version: 3.0 |
| \ \ / A nd | Web: http://www.extend-project.de |
| \ \ M anipulation |
\*-----*/

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "system";
    object controlDict;
}
// ***** //

application fsiFoam;

startFrom startTime;

startTime 0;

stopAt endTime;

endTime 12;

deltaT 1e-3;

writeControl runTime;

writeInterval 0.01;
```

```

purgeWrite 0;

writeFormat binary;

writePrecision 6;

writeCompression compressed;

timeFormat general;

timePrecision 6;

runTimeModifiable yes;
adjustTimeStep no;

maxCo 0.2;

functions
(
  pointHistoryCentre
  {
    type pointHistory;
    functionObjectLibs
    (
      "libpointHistory.so"
    );

    region solid;

    refHistoryPoint (0.10 0.055 0.055);
  }

  pointHistoryTop
  {
    type pointHistory;

```

```

functionObjectLibs
(
  "libpointHistory.so"
);

region solid;

refHistoryPoint (0.10 0.055 0.07);
}

pointHistoryBottom
{
  type pointHistory;
  functionObjectLibs
  (
    "libpointHistory.so"
  );

  region solid;

  refHistoryPoint (0.10 0.055 0.04);
}

hronTurekReport
{
  type hronTurekReport;
  functionObjectLibs
  (
    "libhronTurekReport.so"
  );

  solidNames (cylinder plate);

  timeToActivateCoupling 0;
}

```

```
disableMeshMotionOnSolidForContinuingSim
{
  type disableMeshMove;
  functionObjectLibs
  (
    "libdisableMeshMove.so"
  );

  region solid;
}
);

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

Appendix 5.A: Tent Case Structure

The case structure was planned as follows:

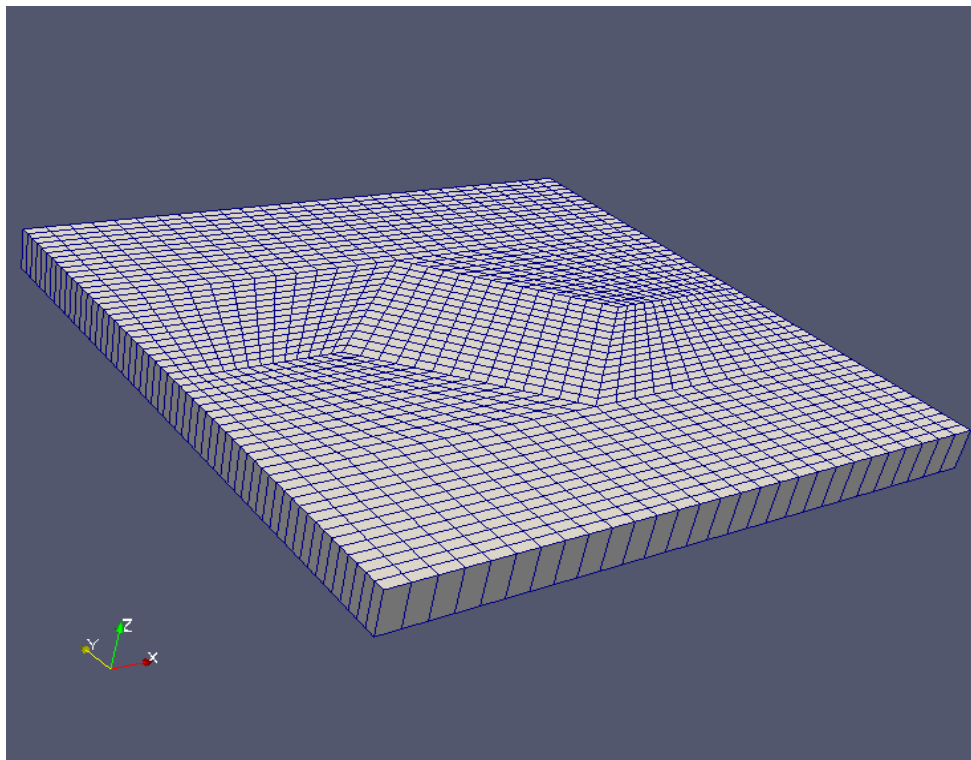
“case_tent_v?” - the main case folder is versioned, so that it’s easier to tell apart from previous iterations what has changed.

- “mesh” – case folder where the fluid and solid region meshes are created.
 - “fluid” – this is the base case folder for the fluid region, where the top part of the mesh is generated and then is assembled with the other fluid region to create the complete region.
 - “fluidBottom” – this is the case folder where only the mesh below the tent is generated. The case folder “fluid” will use the mesh generated here.
 - “solid” – this is the case folder where the solid region is meshed.
- “initialize” – case folder where the fluid region will have the flow initialized around the tent, to avoid sudden bursts of energy into the domain, which would blow the tent away.
- “solidMotion” – case folder where the fluid region uses the final data from “initialize” and has the solid region is operational with FSI.

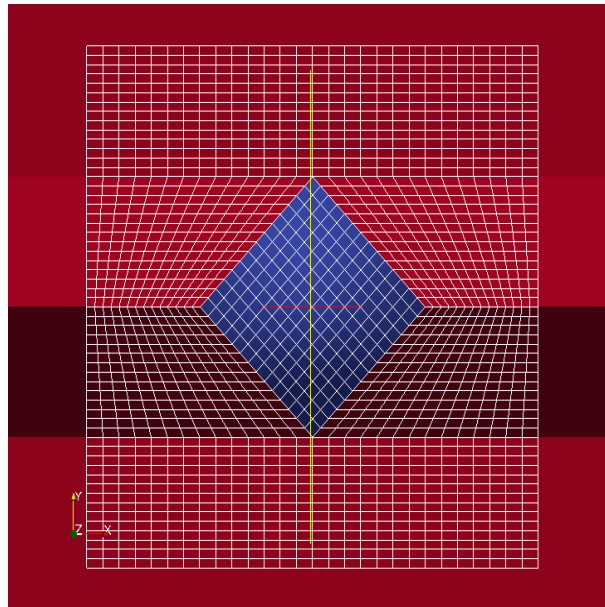
Appendix 5.B: Steps of Meshing the Fluid Region

The steps was done inside the script file “mesh/fluid/Allrun” are provided with images in the following:

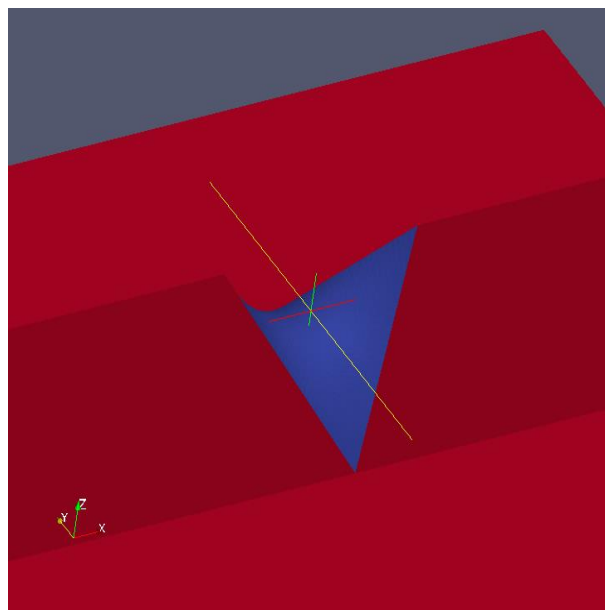
1. The folder “0.mesh.step1” contains the file “pointDisplacement”, which is explained better in step regarding moveDynamicMesh. The copy of this folder to “0” is so that the 0 folder is ready to be used when the time comes.
2. This step is divided into three staged
 - a. blockMesh is executed and creates the mesh shown below. There are 9 blocks drawn in this mesh and in the middle is the shape of the tent surface. The mesh only has 1 cell of thickness, because moveDynamicMesh was not able to properly move the mesh if it had cells along Z. Those will be added at a later step.



- b. The figure below is a top view to make it easier to see the mesh overlaid on the tent surface (in blue). The red surfaces are the extensions to the tent surface, on top of which was can move this mesh onto it.



- c. The figure below is shown what the surfaces look like. These are in the file “constant/triSurface/version6.stl”. Without the red surfaces, would not be able to generate a mesh, because would not be able to attach other mesh blocks onto the block that is going to be morphed onto the tent.

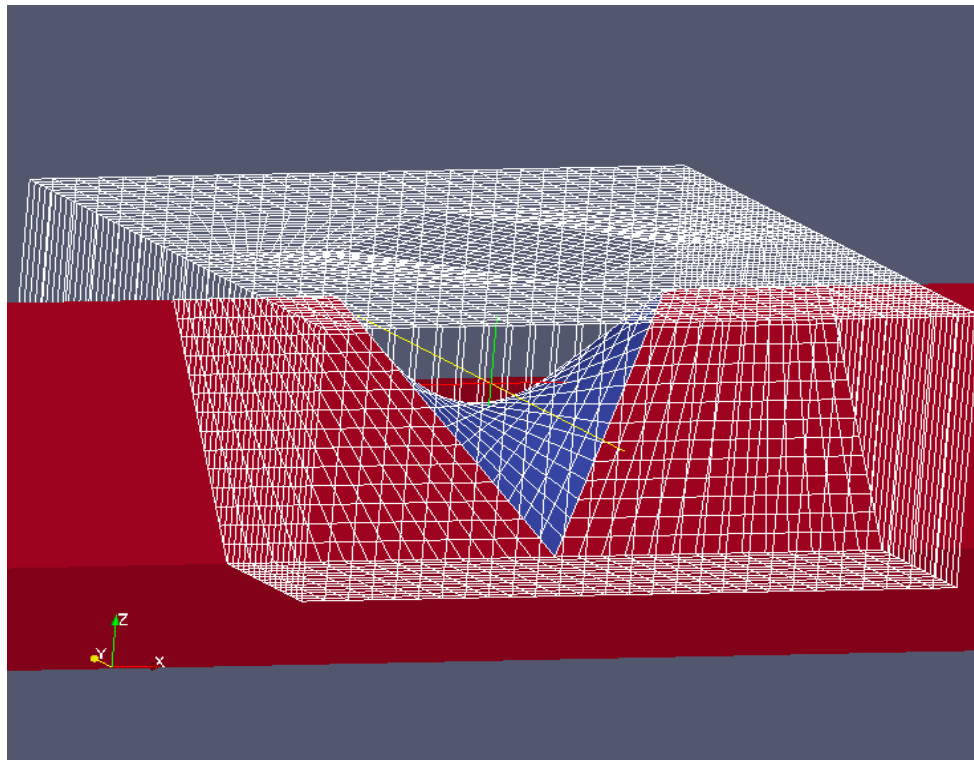


3. `moveDynamicMesh` is then executed to morph the mesh onto the surfaces, which as shown below, it does a perfect job.

The file “0/pointDisplacement” is used for this step and inside it is the boundary conditions for how the boundaries should be moved or fixated, namely:

- The top boundary stays where it is.
- The side boundaries are a special slip condition, where they can stretch/move/morph only along Z.
- The bottom boundaries moved so that they would snap onto the STL surface.

The mesh motion is time based, namely the bottom boundaries move at a maximum velocity of 10 m/s until they hit the surface. That is why the final mesh is then placed in the time folder “2”.



4. This command:

```
cp 2/polyMesh/points* constant/polyMesh/
```

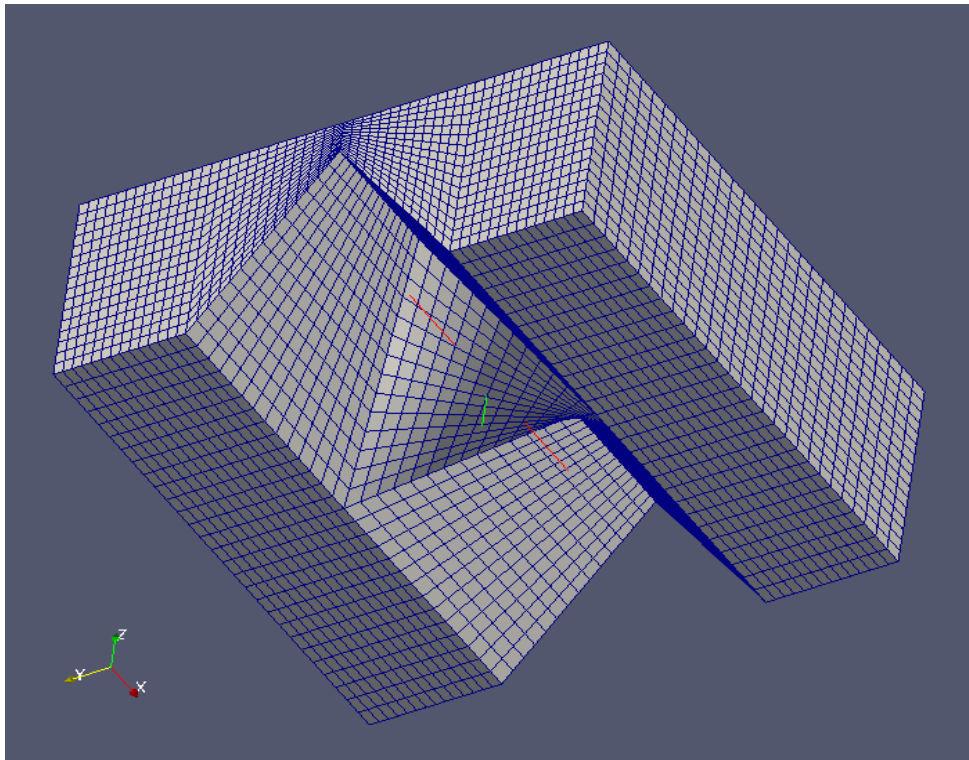
is for copying the points of the mesh that were moved by `moveDynamicMesh`.

The time-step folders “2” and “0” are then deleted, because they are no longer needed.

5. Next step is to mesh along Z, given that the mesh only has 1 cell of thickness at the moment. This is done in 4 iterations of refinement, by using `refineMesh` to split the cells in half along Z each time, resulting in 16 cells at the end.

The command `setSet` is used for selecting all cells of the mesh by relying on the file `setSet.selectAll.c0`, so that the associated selection name can be indicated to `refineMesh` in the file `system/refineMeshDict`.

The resulting mesh looks like as shown below, when seen from the bottom view. As it can be seen, this only handles the mesh on top. The next step is to do the mesh on the bottom side.

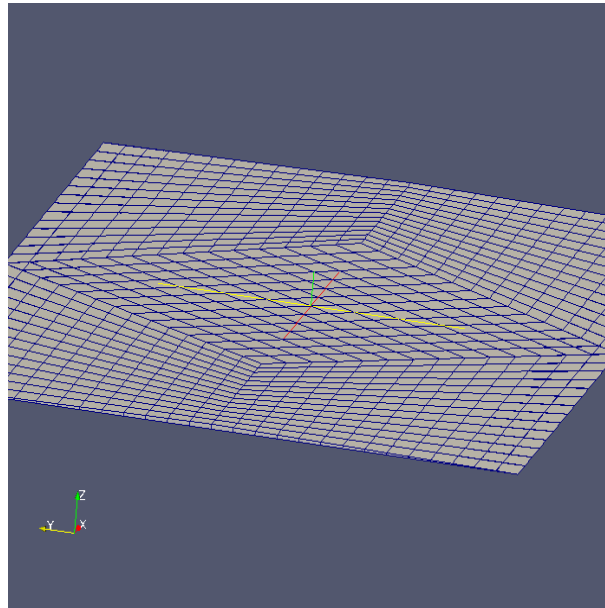


6. The next step is to run the script `mesh/fluidBottom/Allrun`, inside which the first step is to run `blockMesh`.

As shown below, this mesh only has 5 blocks and they are shaped so that the edges to the North and South of the mesh (left and right of this image) are squished. In other words, the cells on those edges are not hexahedrons and are instead triangle prisms.

This is so that the mesh can connect directly with the previous mesh, given that the bottom of the tent is fixed to the ground directly and does not give enough space for hexahedron cells.

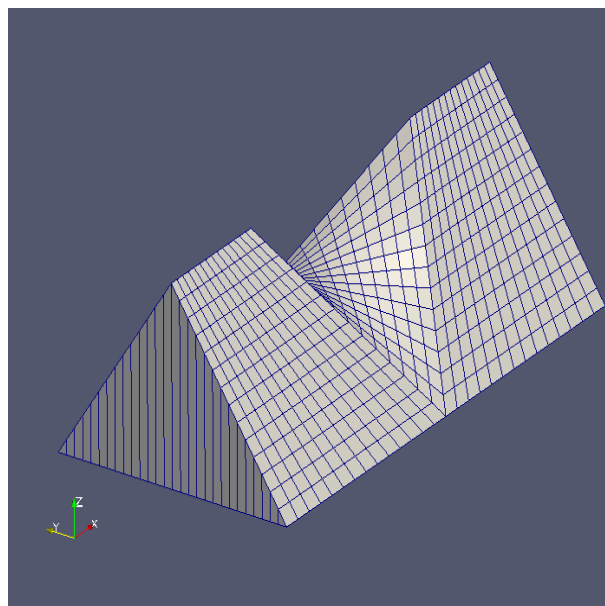
And again, this mesh only has 1 cell thickness along Z.



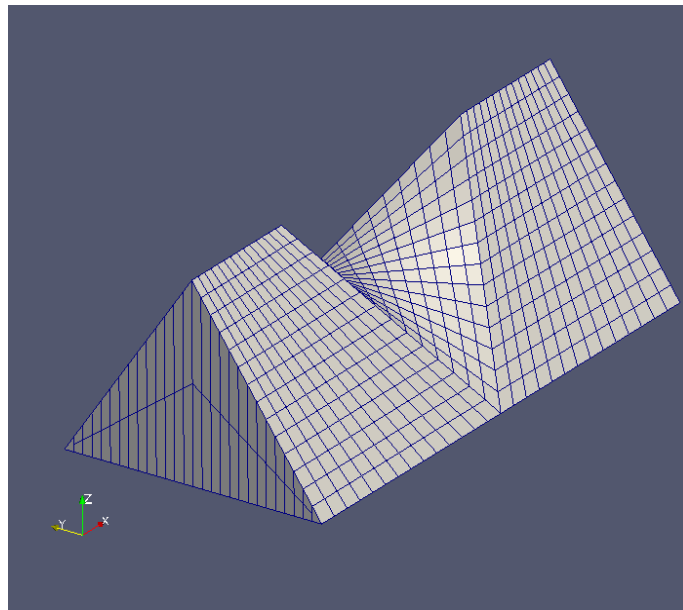
7. The next step is run `moveDynamicMesh`, which relies on the boundary conditions file “0/pointDisplacement”, similarly to as explained for the top region.

Notice the prism cells on the bottom edges to the North and South (left and right in the image).

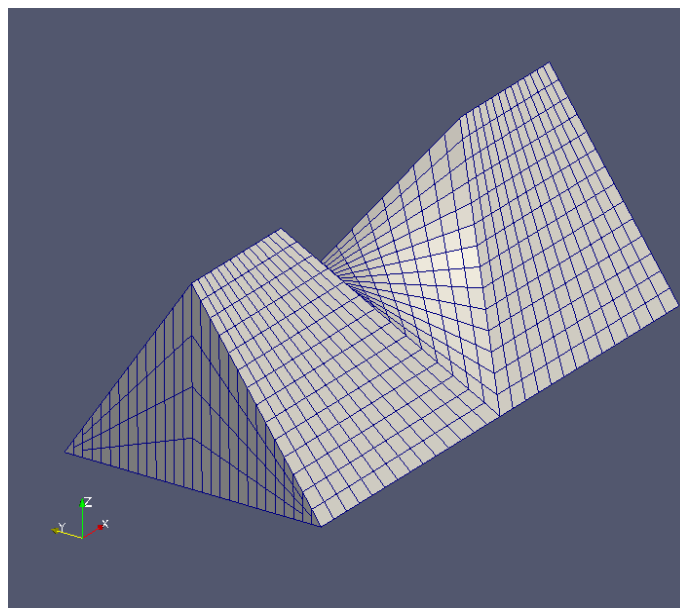
And again, the resulting mesh is placed in the folder “2”.



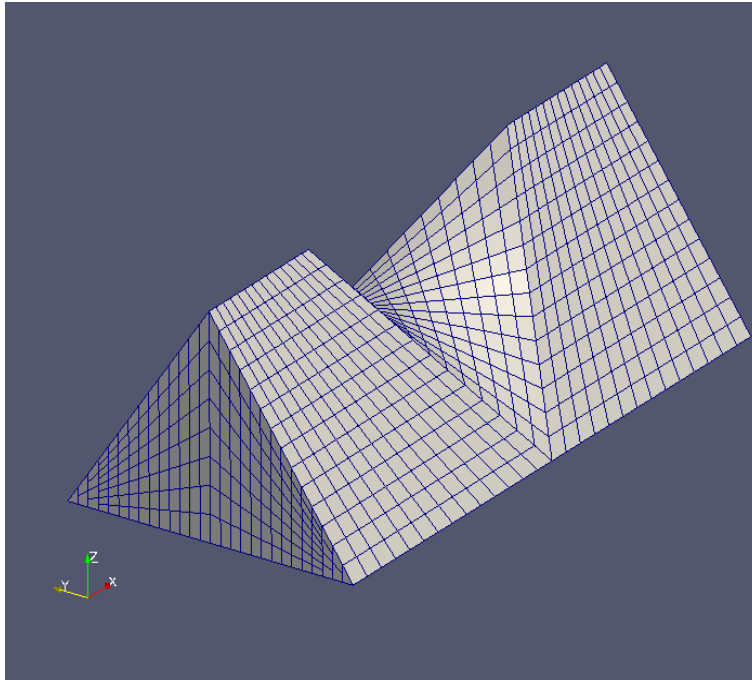
8. The “points” file is also copied from “2” to “constant” and the folder “2” is deleted.
- The mesh is then refined along Z, but in a somewhat different way from before. Instead of selecting all cells of the mesh, only some regions of the mesh are selected and refined, namely so that more elongated cells are divided in half more times than the shorter cells.
- This is done by using the files “setSet.selectAll.c0.?” with setSet mesh, where each file has a selection box that reduces the North/South limits with each step.
- Below is the result for the first step.



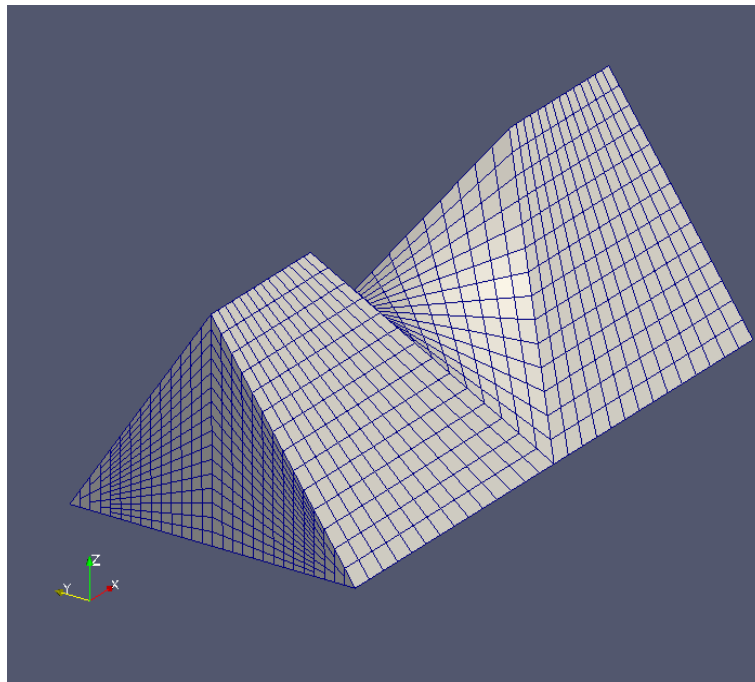
- Below is the result of the second step. Notice that to the North and South (left and right edges) the cells on the edges are not as refined.



c. Then, the next step is shown below

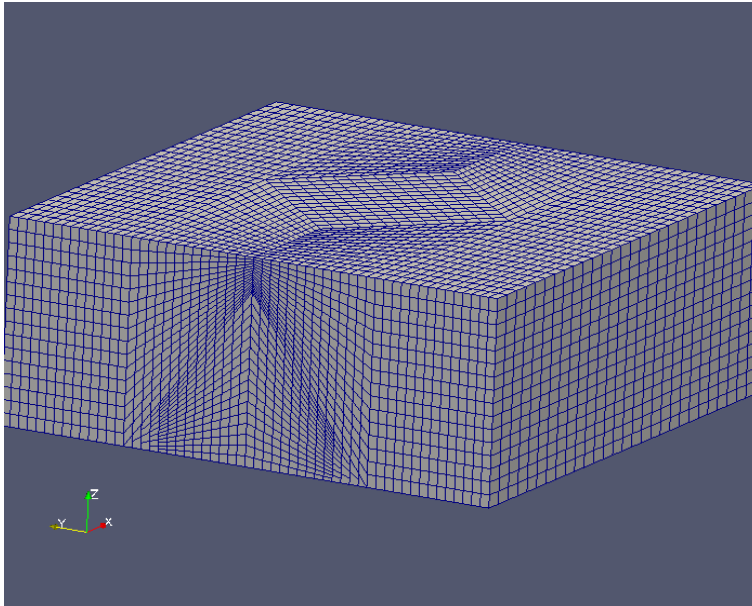


d. The final refinement step is in the next figure. Notice that on the middle axis, the refinement ended up with 16 cells along Z, as done for the top region. This completes the generation of the bottom mesh.



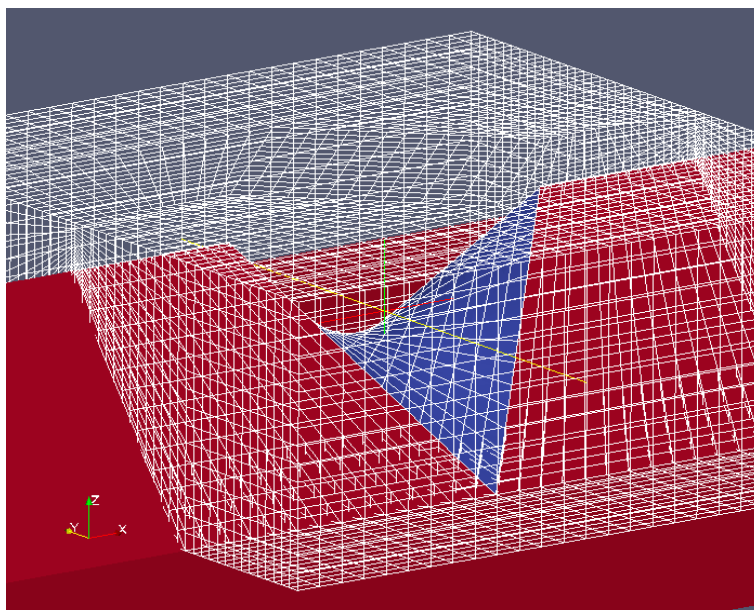
10.

a. Next, continue with the original script “mesh/fluid/Allrun”. The application mergeMeshes is used for getting the mesh from “mesh/fluidBottom” into the mesh in “mesh/fluid”.



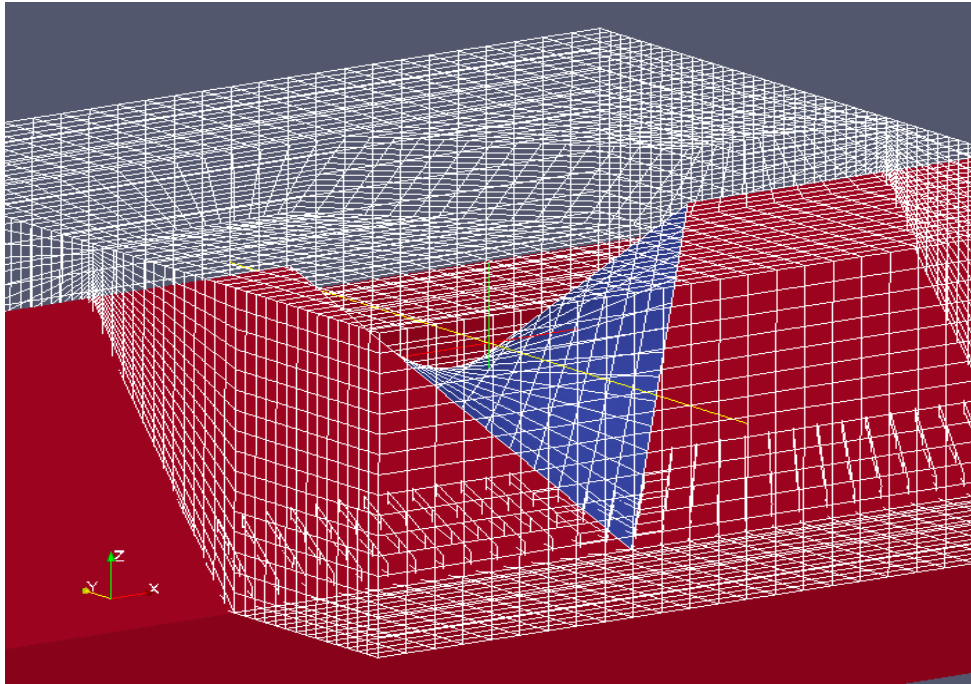
b. But this is not yet completed, because this only means that the two meshes are now in the same case, but they still have the old top and bottom boundaries that they had in the original states, as shown in the image below.

Notice that there are lines on the red surface within the mesh.



11. This step will now stitch some of the bottom boundaries from the original “fluid” mesh with some of the top boundaries that came from on the “fluidBottom” mesh.

As shown in the next image, this is the resulting mesh. There are a few rendering artifacts at the inclined-bottom of the mesh, but those are only some mild rendering issues with ParaView’s interpretation of the mesh and they can be ignored.



12. Finally, the time folder “0” is deleted once again, because some files were created there during the latest meshing operations, which can be ignored.

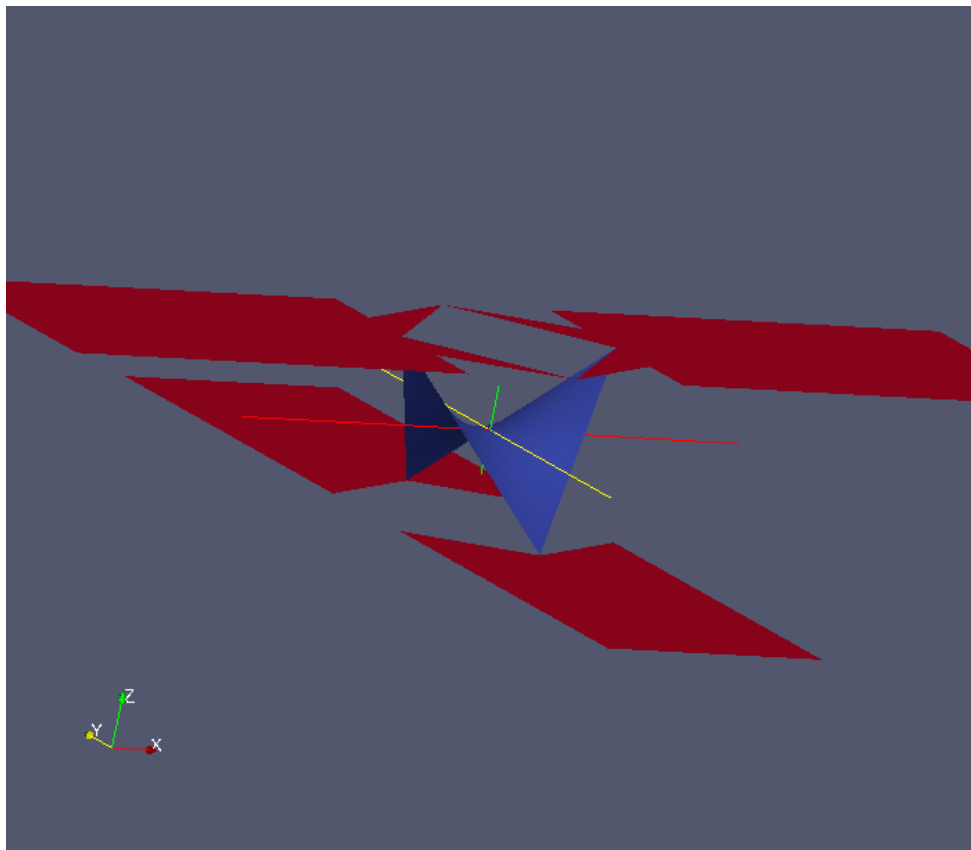
Appendix 5.C: Steps of Meshing the Solid Region

The following steps refer to the script “mesh/solid/Allrun”:

The very first step is the definition of the thickness of the tent. It was originally defined as 4 mm = 0.004m as an experimental thickness, until we could figure out what should be the correct thickness while also being able to simulate the case.

The idea is that the previous STL file (created using SolidWork) would be then copied and translated with a motion along Z with this thickness, so that we can have the mesh for the tent defined in-between.

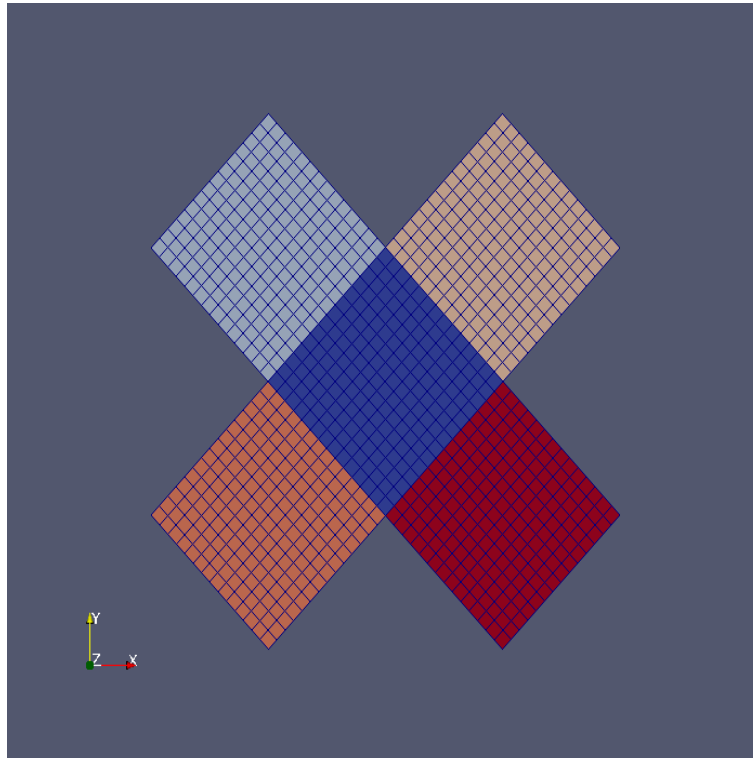
However, since the tent surface has to be fixated at the tips, it was necessary to create another STL (shown below) for this case folder, so that the morphed mesh would have cells that would be horizontally fixated. More details in the steps below.



surfaceTransformPoints is executed next, to do the copy and translation along Z of 4mm, as indicated in the previous step.

The next step is to run blockMesh.

Shown below is the resulting mesh. The different colors show that the mesh is divided into 5 odd-looking blocks. This is because will be doing several changes to the blocks in the diagonal corners, namely North-West, North-East, South-West and South-East.



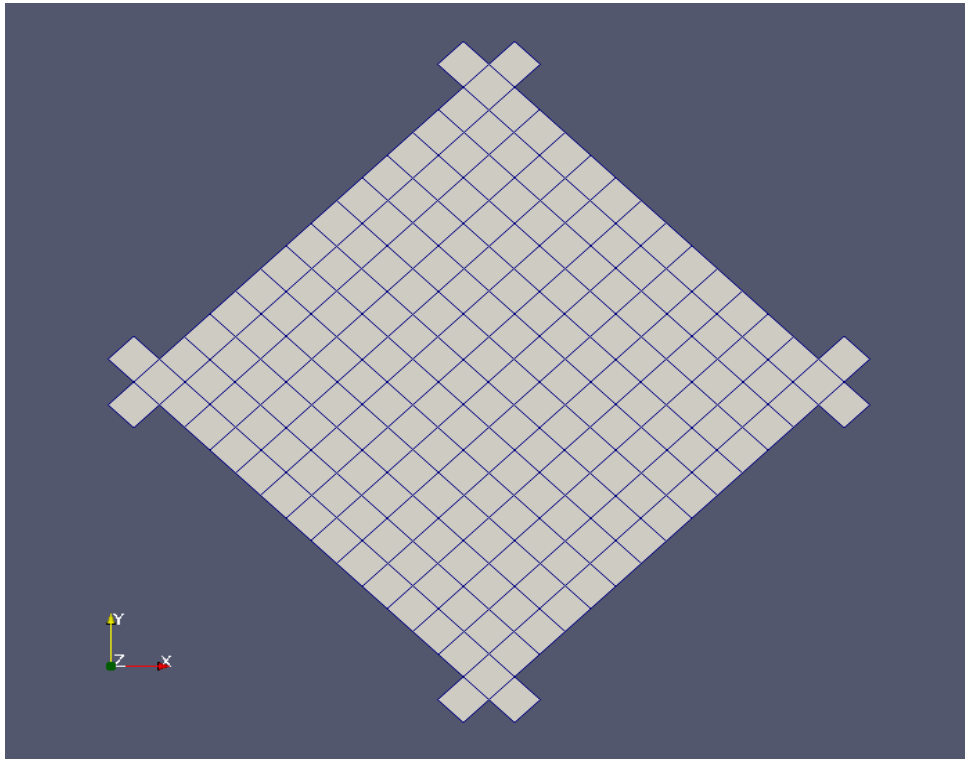
The next step uses the dictionary file “system/topoSetDict.removeExcessCells”, by running the topoSet utility with the “-dict” option. This will select all of the cells that want to keep in this mesh and create a cell set named “mainCellSet” that keeps the cells that matter.

The idea is that will be keeping 8 cells on the tips of the tent surface, 2 on each tip.

The application subsetMesh will then use that cell set to create a new mesh that only has the cells wanted, as shown in the below figure.

This seemingly insane strategy is because it’s not possible to create this mesh this way directly with blockMesh.

The 8 cells (2 each) on the tips will be the cells that will be holding the tips of the tent in place.



The next step is to use topoSet with “-dict system/topoSetDict.fixtures” to select the North, West, East and South faces on the 8 cells on the tips. It will be clearer in the next step what’s going on here.

Now the faces that were selected in the previous step are converted to patches with createPatch and the dictionary file “system/createPatchDict”.

4 patches are created this way:

westFixture

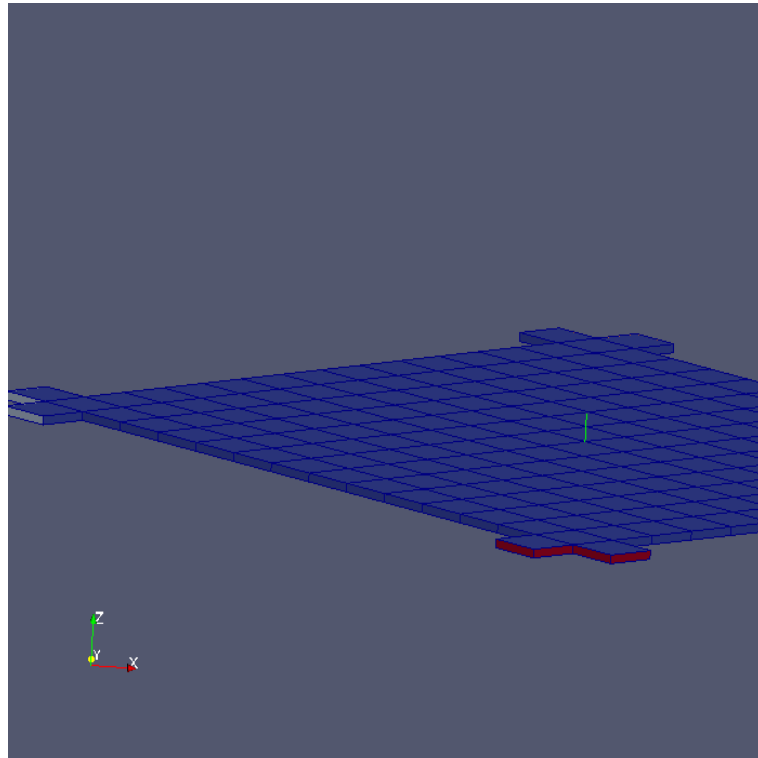
eastFixture

northFixture

southFixture

As shown below, the red and light blue faces are the South and West faces that have been created with the help of the previous step and this step. These faces will be fixated and will hold the tent tips in place.

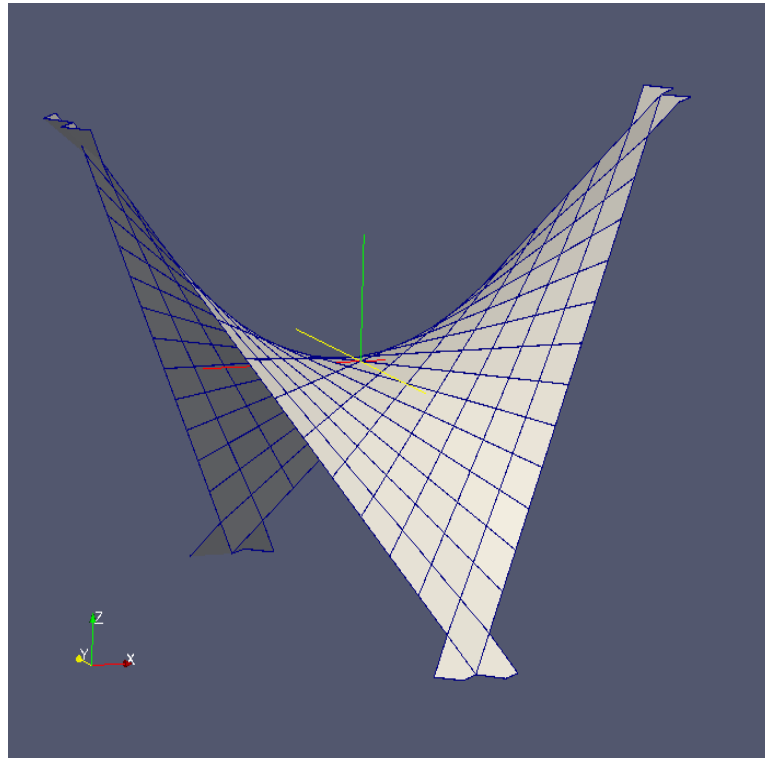
However, this is not yet the final mesh, since it does not yet have the shape of the tent.



The folder “0.meshMotion” contains the file “pointDisplacement”, which is explained better in step regarding moveDynamicMesh. The copy of this folder to “0” is so that the zero folder is ready to be used when the time comes.

The next step is run moveDynamicMesh, which relies on the boundary conditions file “0/pointDisplacement”, similarly to as explained for the fluid region.

Below is the resulting mesh. There is a bit of a rendering glitch, since the tips look weird, but the idea that matters is that the patches created in a previous step are horizontal and can hold the tent tips in place



The next steps are nearly identical to those done for the top fluid mesh, namely:

“points” files are moved from “2” to “constant”.

The mesh is refined along Z, but only split into 4 cells along Z (2 refinement iterations).

It’s not yet clear if this is enough or too much.

Finally, the “0” folder is deleted.

Appendix 5.D: Allrun Script File

```
#!/bin/sh
# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions
# Get application name
application=`getApplication`
./removeSerialLinks ./
(
  cd ..
  ./makeSerialLinks fluid solid
)
rm -rf 0
cp -r 0.org 0
ln -s ../../solid/0 0/solid
# Need to copy the meshes generated with OpenFOAM 2.4.*
if [ -e ../../mesh/fluid/constant/polyMesh/points ]
then
  cp -r ../../mesh/fluid/constant/polyMesh/* constant/polyMesh/
else
  echo "Error: Need the 'fluid' mesh to be generated first in the case folder '../../mesh/'"
  exit 1
fi
if [ -e ../../mesh/solid/constant/polyMesh/points ]
then
  cp -r ../../mesh/solid/constant/polyMesh/* constant/solid/polyMesh/
else
  echo "Error: Need the 'solid' mesh to be generated first in the case folder '../../mesh/'"
  exit 1
fi
runApplication -l log.setSet.solid \
setSet -case ../solid -batch ../solid/setBatch
runApplication -l log.setToZones.solid \
```

```
setsToZones -case ../solid -noFlipMap
runApplication setSet -batch setBatch
runApplication setsToZones -noFlipMap
runApplication mapFields ../../initialize/fluid -sourceTime 5.0
cp 0.org/pointMotionU 0/
rm 0/U_0* 0/cellMotionU*
runApplication $application

# ----- end-of-
```