Swansea University
Prifysgol Abertawe

College of Engineering
Civil and Computational Engineering Centre

Thesis of Master of Science in Computational Mechanics

A Weakly Coupled Strategy for Computational Fluid-Structure Interaction: Theory, Implementation and Application

Student: Sander Vaher

Supervisor: Dr. Wulf Georg Dettmer

Submitted to the University of Wales in fulfilment of the requirements for the Degree of Master of Science in Computational Mechanics

Swansea University
May 2013

Summary

This study investigates the implementation of weakly coupled fluid-structure interaction scheme into a real computer program.

Firstly, the introduction to theoretical background of finite element method with focus on linear elasticity and finite volume method with focus on fluid dynamics, is given. This is followed by the the description of the strategies for solving fluid-structure interaction problems, with the emphasis on staggered or weakly coupled schemes.

Staggered scheme provides the the basis of employing sophisticated sub-solvers for each domain. The individual sub-solvers in this work are an open source finite volume method based OpenFOAM and an in-house finite element method based MPAP2. As both of the solvers are written in the C++ programming language, the software are linked together by making OpenFOAM libraries available in MPAP2. This provides a fast and efficient data transfer between the sub-solvers. Few additional data structures and functions are programmed, which define the interaction between the sub-domains. The main steps of the implementation of the specific staggered scheme algorithm, developed by Dettmer and Perić [5], are described.

Finally, numerical examples are presented. The first part of examples focuses on classical problems of fluid flow and structural dynamics that are solved by employing each sub-solver individually. In second part, a two-dimensional fluid-structure interaction benchmark problem is solved for various meshes and solution parameters. A conclusion is driven based on the theory of added mass effect and on the results of numerical simulations.

Declarations and Statements

Declaration

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

Signed:	....
Date:	....

Statement 1

This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed:	....
Date:	....

Statement 2

I hereby give my consent for my dissertation, if relevant and 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:	....
Date:	....

Contents

Summary . . . i
Declarations and Statements . . . iii
Contents . . . vi
Acknowledgements . . . vii
List of Figures . . . ix
List of Tables . . . x
Abbreviations . . . x

1 Introduction 1
1.1 Motivation and Problem Description . . . 1
1.2 Layout of the Thesis . . . 2

2 The Basics of Finite Element Modelling 5
2.1 Mechanics of Deformable Solids . . . 5
2.1.1 The Concept of Lagrangian Description . . . 5
2.1.2 The Governing Equations for Solids . . . 6
2.2 The Weak Form . . . 7
2.3 Solution Procedure . . . 11

3 The Basics of Finite Volume Modelling 13
3.1 Governing Equations of Fluid Flow . . . 13
3.2 Spatial Discretisation Using Finite Volume Formulation . . . 16
3.2.1 Approximation of Surface Integrals . . . 18
3.2.2 Approximation of Volume Integrals . . . 19
3.2.3 Interpolation and Differentiation Schemes . . . 19
3.2.4 Diffusion Term and aspects of non-orthogonality . . . 22
3.3 Temporal discretisation and methods for unsteady problems . . . 23
3.4 Solution to Navier-Stokes Equations . . . 26
3.4.1 Derivation of pressure equation . . . 26
3.4.2 Solution to pressure-velocity coupling . . . 28
3.5 Calculation of traction forces . . . 29

4 Computational Fluid-Structure Interaction 31
4.1 Classification of Fluid-Structure Interaction Solvers . . . 32
4.2 Staggered Scheme for Fluid-Structure Interaction . . . 34
4.3 Stability of Staggered Scheme . . . 37

5 Implementation of the Weakly Coupled Scheme 39
5.1 MPAP2 . . . 39
5.2 OpenFOAM . . . 39
5.3 Assembly of Sub-solvers into Single Computer Program . . . 42

6 Numerical Examples 49
6.1 Fluid Dynamics - Channel Flow (study on grids) . . . 49
6.2 Solid Dynamics - Cantilever Beam . . . 51
6.3 Fluid Dynamics - Flow around cylinder . . . 55
6.4 Fluid Dynamics - Flow in a Channel with Prescribed Wall Motion 60
6.5 FSI - Flow Induced Oscillations of a Flexible Beam . . . 62

7 Conclusions 73

Acknowledgements

It is impossible to overvalue my supervisor, Dr. Wulf G. Dettmer, encouragement and time guiding me throughout this masters course. I am truly grateful to him for being my thesis supervisor.

I also would like to thank European Commission for sponsoring this Master of Science program and my studies in UPC Barcelona and Swansea University. It has helped me to widen the knowledge not just in Computational Mechanics, but also in various cultures by meeting open minded and enthusiastic young people from all over the world.

Special thanks to my undergraduate supervisor Dr. Hendrik Naar from Tallinn University of Technology for introducing the world of numerical methods.

Finally, I would like to thank my parents for their support and encouragement not just in my studies, but at any time and anywhere.

List of Figures

2.1 Motion of deformable body 6
2.2 Typical partial differential equation problem over a complex domain 8
3.1 Arbitrary 2D finite volumes grid with notations 17
3.2 Upwind Differencing 20
3.3 Central Differencing 21
3.4 Various approaches for non-orthogonality correction 23
3.5 Forward and backward Euler approximation of time integral . . . 24
3.6 Midpoint and trapezoidal rule for approximating time integral . . 25
4.1 Structure of a FSI solver [4] 31
4.2 Classification of FSI solvers 33
4.3 Interpolation of traction forces 36
5.1 Structure of typical pimpleDyMFoam case directory 41
5.2 Interface data transfer 47
6.1 Geometry of channel flow problem 49
6.3 Velocity field at outlet 50
6.2 Pressure distribution of 2D channel flow for various grids 51
6.4 Geometry of clamped pipe 52
6.5 Finite element model of the pipe 52
6.6 Oscillation of free end of pipe 54
6.7 Pipe end displacement at the end of the first period 55
6.8 Geometry and boundary conditions of flow around cylinder problem 56
6.9 Grid fragments of flow around cylinder problem 57
6.10 Comparison of drag coefficients 58
6.11 Comparison of lift coefficients 59
6.12 Pressure field of fully developed wake region (OpenFOAM fine grid) 59
6.13 Geometry and boundary conditions of channel with wall indentation 60
6.14 Fragment of the grid at $t = 13.5 \approx T/2$ 61
6.15 Vorticity (colour scale: $-6 \ldots +6$) 62
6.16 Geometry and boundary conditions of oscillating beam 63
6.17 Overview of the fine mesh 64
6.18 Coarse mesh at the connection of the bluff body and the beam . . 64
6.19 Oscillation of the beam tip (coarse mesh, $\beta = 0.05$) 68
6.20 Oscillation of the beam tip (coarse mesh, $\beta = 0.3$) 68
6.21 Oscillation of the beam tip (coarse mesh, $\beta = 0.5$) 69
6.22 Oscillation of the beam tip (coarse mesh, $\beta = 0.8$) 69
6.23 Flow induced oscillations (vorticity colour scale -150...+150) . . . 70
6.24 Oscillation of the beam tip (fine mesh, $\beta = 0.05$) 71
6.25 Oscillation of the beam tip (fine mesh, $\beta = 0.3$) 71
6.26 Oscillation of the beam tip (fine mesh, $\beta = 0.5$) 72
6.27 Oscillation of the beam tip (fine mesh, $\beta = 0.8$) 72

List of Tables

3.1 Classification of fluid flows based on various properties 13
3.2 Schemes of finite volume method CFD codes 25
6.1 Grid properties of channel flow model 50
6.2 Simulation properties for flow around cylinder problem 56
6.3 Comparison of lift and drag coefficients 58

6.4 Fluid grid properties of oscillating beam problem 63
6.5 Results of the coarse grid simulation 66
6.6 Results of the fine grid simulation ($\Delta t = 0.00025$) 66

Abbreviations

2D	two-dimensional
3D	three-dimensional
CD	central differencing
CFD	computational fluid dynamics
CM	control mass
CV	control volume
FSI	fluid-structure interaction
GUI	graphical user interface
LSE	linear system of equations
OS	computer operating system
UD	upwind differencing

1 Introduction

1.1 Motivation and Problem Description

Fluid-structure interaction (FSI) is an important area in the field of computational multiphysics. Problems, involving coupling of fluid flow with a solid structure, are common not just in structural engineering, but also in biology and medicine.

Often in ordinary engineering problems, the affect of the fluid-structure interaction is negligible, because of minute traction forces or a very stiff structure. For example, in most cases wind cannot deform a building at a scale, which changes its structural behaviour or changes the patten of air flow.

However, there is an unlimited number of cases where the fluid-structure interaction cannot be disregarded and an efficient fluid-structure interaction solver becomes useful. For instance,

Mechanical engineering - turbo-machinery, ship dynamics and wind turbines;
Aerospace engineering - aeroelasticity, flutter of wings, propellers and rotors;
Civil engineering - slender (suspension-)bridges, flutter of cables;
Biomechanical engineering - artificial organs (heart, lungs), blood flow in vessels

Named examples illustrate the need for a fluid-structure interaction solver, which should be robust, accurate and computationally efficient.

Computational fluid-structure schemes can be classified as strongly (equilibrium of kinematics and traction forces is satisfied exactly at the interface) and weakly coupled (in general traction forces equilibrium is not satisfied). Strongly coupled strategies have wider range of applicability, but are computationally more expensive compared to weakly coupled or staggered schemes. The restricted range of applicability of staggered schemes is due to instabilities occurring if the ratio of solid over fluid is small. In weakly coupled schemes the added mass effect is not accounted for and it causes inaccuracy of traction forces, which lead to inaccuracy in deformation of the solid structure and finally to instabilities.

Many strongly coupled schemes have been developed in last two decades, including partitioned/monolithic, exact/inexact Newton-Raphson procedures and Gauß-Seidel (or Dirichlet-Neumann or fixed point iteration schemes) iteration. A variety of problem-specific schemes are suggested to improve the computational efficiency of the strongly coupled schemes (see, for example, [21]), but in general the computation time for staggered scheme is one order of magnitude shorter [5].

Combining two individual solver into a single fluid-structure solver is convenient only with Gauß-Seidel and staggered schemes. In most of the cases the fixed point iteration strategy or weakly coupled scheme with compressible, explicit or first order accurate fluid sub-solver have been implemented to ensure the stability.

Examples of can be found of combining OpenFOAM and FEAP with the help of Component Tempered Library by Kassiotis *et al.* [12], [13]; OpenFOAM and DEAL.II by Johan Lorentzon [15]; OpenFOAM and Code_Aster by Romanelli *et al.* [20] *etc.*

The aim of this thesis is to develop a robust and efficient FSI solver by combining individual fluid and solid simulation codes into single software. As the fluid sub-solver of this project has the capabilities of turbulence modelling, the ultimate goal is to model FSI problems with high Reynolds number over a long period of time. The attention of current project is restricted to incompressible flows.

A open-source computational fluid dynamics (CFD) solver, OpenFOAM, will be employed as a fluid sub-solver in the final FSI computer program. In-house finite element software, MPAP2 (Multi-Physics Analysis Program), developed by Dr. Wulf G. Dettmer in Swansea University Civil and Computational Engineering Centre, will be used as solid sub-solver in the current project. The coupling will follow the strategy suggested by Dettmer and Perić in paper [5].

1.2 Layout of the Thesis

This thesis is divided into *theory* (Sections 2 - 4), *implementation* (Section 5) and *application* (Section 6) parts.

Theory. In Section 2 the basics of finite element analysis is presented on the basis of two typical boundary value problems - heat conduction and linear elasticity. The steps from governing equations to weak form and finally to system of algebraic equations, are given. The solution procedure of a typical finite element computer program closes the section on solid sub-solver.

Section 3 focuses on finite volume method on the basis of OpenFOAM. The most important spatial and temporal discretisation methods are described. Solution procedures for Navier-Stokes equations with the emphasis on pressure-velocity coupling using *SIMPLE* and *PISO* algorithms, are given. Finally the computation of traction forces of the fluid flow, are viewed.

Section 4 provides an introduction to computational fluid-structure interaction with the focus on weakly coupled schemes. Stability issues of a specific staggered strategy are discussed. This section closes the theory part of the thesis.

Implementation. In Section 5, firstly, the structure of both sub-solvers, MPAP2 and OpenFOAM is given. This is followed by instructions describing the steps to link OpenFOAM with MPAP2. A brief overview of new data structures and key points of transferring the data across the interface is provided.

Application. Section 6 provides a numerical examples of each sub-solver and finally the coupled fluid-structure interaction solver. Two simple computational fluid dynamics problems are solved in OpenFOAM and a single structural dynamics problem in MPAP2. This section will be closed by a numerical example on a fluid-structure interaction benchmark problem, which is solved for various meshes and control parameters.

Section 7 brings the conclusions of this work and suggests directions for further work.

2 The Basics of Finite Element Modelling

Many physical phenomena can be described by partial differential equations and boundary conditions. Analytical solutions may not be available due to the complexity of equations or the domain. Thus, numerical methods are employed to solve complex problems by exploiting computers. The most powerful, robust and widely used numerical solution technique nowadays, is the finite element method. Finite element analysis can be used with structured and unstructured meshes for various partial differential equation types.

In this section only the basics of finite element modelling for structural problems will be described. Firstly, the governing equations of solid mechanics will be visited. Secondly, the weak form of linear steady state problem will be presented. Finally a solution procedure on the basis of a real computer program will we viewed.

For additional information on finite element analysis, refer to any relevant textbook. (e.g Cook *et al* [2] or Reddy [19]).

2.1 Mechanics of Deformable Solids

2.1.1 The Concept of Lagrangian Description

Kinematics of solid mechanics are usually described in the Lagrangian setting (material description) as the displacements are small compared, for example, to fluids. It means that any particle position of physical property are described in referential coordinates and time. Usually the reference configuration is the configuration at $t = 0$.

Thus, two position fields for particles can be described - the initial position and current, as shown in Figure 2.1. The motion of body is described by mapping function ϕ from initial configuration to current,

$$\mathbf{x} = \phi(\mathbf{X}, t), \tag{2.1}$$

where $\mathbf{x}$ denotes the current and $\mathbf{X}$ the initial location.

Material derivative of the current position of particle, $\mathbf{x}$, is the instantaneous velocity $\mathbf{v}$ of that particle. Thus the velocity can be expressed as,

$$\mathbf{v} = \dot{\mathbf{x}} = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \frac{\partial \phi(\mathbf{X}, t)}{\partial t} \tag{2.2}$$

and the acceleration as,

$$\mathbf{a} = \dot{\mathbf{v}} = \ddot{\mathbf{x}} = \frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2} = \frac{\partial^2 \phi(\mathbf{X}, t)}{\partial t^2}. \tag{2.3}$$

Displacement is a vector pointing from point P in the initial configuration to

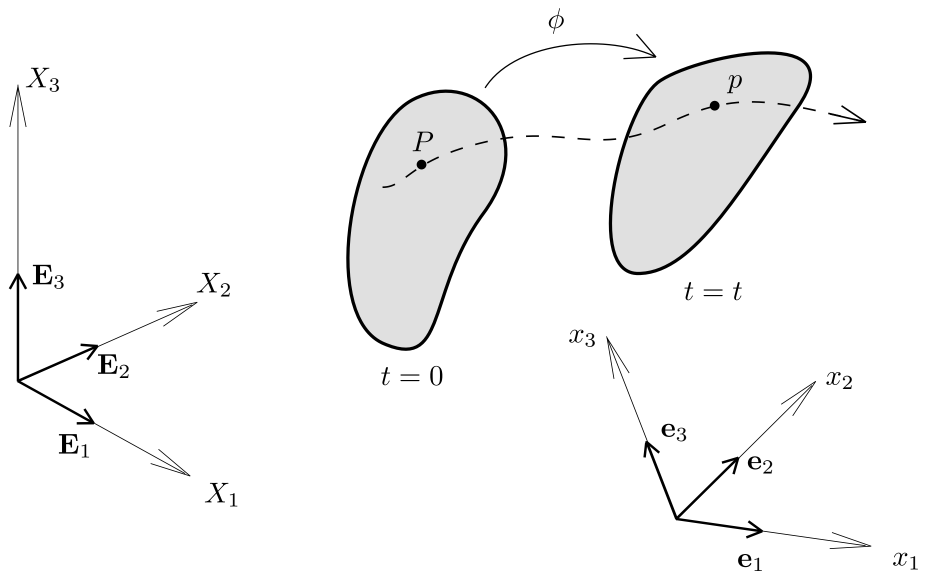


Figure 2.1: Motion of deformable body

point p in the final configuration and thus, a displacement field can be described for the body. In Lagrangian description it becomes,

$$\mathbf{u}(\mathbf{X}, \mathbf{t}) = \mathbf{u}_b + \mathbf{x}(\mathbf{X}, \mathbf{t}) - \mathbf{X}, \tag{2.4}$$

where $\mathbf{u}_b = \mathbf{0}$, if the coordinate systems for undeformed and deformed configuration coincide.

2.1.2 The Governing Equations for Solids

The equations that govern the mechanics of solid continuum, are the balance equations (mass, momentum, and energy) equations, kinematic relations and finally constitutive equations.

For simplification proposes, only the linear momentum conservation equation is given. The full set balance equations is given in any continuum or solid mechanics textbook (e.g Bonet and Wood [1] or Holzapfel [9]).

The conservation of linear momentum reads,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \rho\mathbf{v}\,\mathrm{d}V = \int_\Omega \rho\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t}\,\mathrm{d}V = \int_\Omega (\nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{b})\,\mathrm{d}V, \tag{2.5}$$

which in differential (local) form gives,

$$\rho\ddot{\mathbf{x}} = \nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{b}, \tag{2.6}$$

where $\boldsymbol{\sigma}(\mathbf{x}, \mathbf{t})$ is the Cauchy stress tensor and $\mathbf{b}$ is the acceleration vector for body forces.

If the displacements $\mathbf{u}$ are small, then the configuration $\mathbf{x}$ can be approximated by $\mathbf{X}$ and the settings for small strain (geometrically linear) analysis are obtained.

Consitutive law. In order to close the system one needs a relationships between the stresses, strains and material properties. In this work only Neo-Hookean material model is considered for the solid domain. The constitutive equation of this hyperelastic material is given as [3],

$$\boldsymbol{\sigma} = G\, J^{-\frac{5}{3}} \left(\mathbf{B} - \frac{1}{3}\mathrm{tr}(\mathbf{B})\mathbf{I}\right) + K\frac{J^2-1}{2J}\,\mathbf{I}, \tag{2.7}$$

where G and K are the material shear and bulk modulus, respectively and the left Cauchy-Green deformation tensor $\mathbf{B}$ is given as,

$$\mathbf{B} = \mathbf{F}\,\mathbf{F}^T, \tag{2.8}$$

where

$$\mathbf{F} = \frac{\partial \phi}{\partial \mathbf{X}} = \nabla_0 \phi\,. \tag{2.9}$$

Denoting the mapping as, $\mathbf{x} = \phi(\mathbf{X}) = \mathbf{x}(\mathbf{X})$, the Equation (2.9) can be expressed as,

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}. \tag{2.10}$$

In the framework of infinitesimal strains, Equation (2.7) reduces to

$$\boldsymbol{\sigma} = 2G\,\left[\frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right) - \frac{1}{3}(\nabla\cdot\mathbf{u})\,\mathbf{I}\right] + K(\nabla\cdot\mathbf{u})\,\mathbf{I}. \tag{2.11}$$

2.2 The Weak Form

To employ the finite element method solving partial differential equations (e.g governing equations presented in the preceding section) specific steps must be followed. The domain is divided into finite number of elements, the continuous equations are discretised and system of algebraic equations is built. The discretisation process involves defining the weak form of the governing equations, which is presented in this section.

Denoting a linear differential operator, of any order, as $\mathcal{L}(\bullet)$, the governing equations can be written as,

$$\mathcal{L}(u) = -f \qquad \forall\,\mathbf{x}\in\Omega \tag{2.12}$$

$$u = u_D \qquad \forall\,\mathbf{x}\in\Gamma_D \tag{2.13}$$

$$\mathbf{q}(u)\cdot\mathbf{n} = q_N \qquad \forall\,\mathbf{x}\in\Gamma_N, \tag{2.14}$$

where Ω is the domain of the problem, Γ_D and Γ_N denote the Dirchlet and Neumann type boundary condition part of the boundary Γ. Expression, $\Gamma = \Gamma_D \cup \Gamma_N$ holds at any time for the domain. The variable f denotes the source term in Ω. The quantities u_D and q_N represent prescribed boundary values of the primary variable and prescribed fluxes (secondary variable) through boundary, respectively. The secondary variable $\mathbf{q}(u)$ is a linear function of the first derivative of the primary variable u. And finally, vector $\mathbf{n}$ denotes the unit outward normal of the boundary Γ.

The graphical representation of this problem can be seen in Figure 2.2.

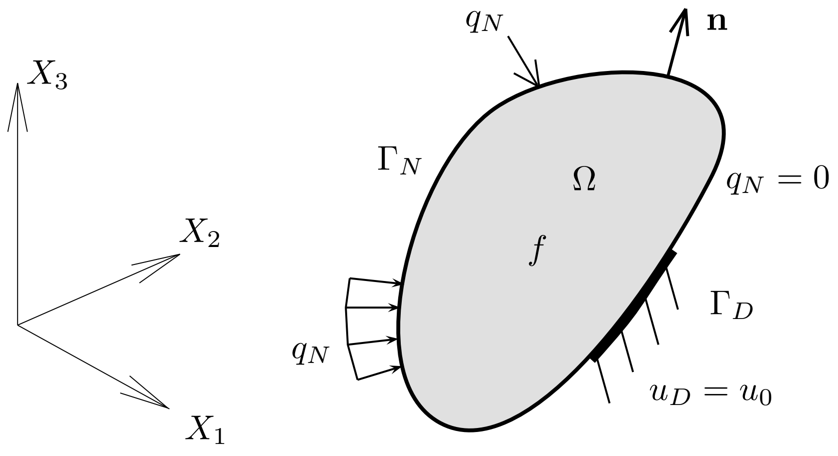


Figure 2.2: Typical partial differential equation problem over a complex domain

In this work, the typical examples of finite element analysis will be followed: heat conduction and linear elasticity (referring to equations in Section 2.1.2).

In case of heat transfer problem, the operator $\mathcal{L}$ denotes a Laplacian term in heat conduction equation

$$\nabla \cdot (\kappa \nabla u) = -f \qquad \forall\, \mathbf{x} \in \Omega \tag{2.15}$$

$$u = u_D \qquad \forall\, \mathbf{x} \in \Gamma_D \tag{2.16}$$

$$\mathbf{q}(u) \cdot \mathbf{n} = q_N \qquad \forall\, \mathbf{x} \in \Gamma_N, \tag{2.17}$$

where

κ	– thermal conductivity
u	– temperature
f	– heat source in Ω
u_D	– prescribed temperature on boundary
q_N	– prescribed heat flux on boundary (e.g $q_N = 0$ for isolated boundary).

As noted previously, the secondary variable is the derivative of the primary. Thus, $\mathbf{q}(u) = \kappa \nabla u$.

Such description is named the *strong form* of the problem.

Instead of enforcing the Equation (2.15) to hold at any point on the domain, the expression can be enforced *weakly* by multiplying the strong from by weighting function w and taking an integral over the domain. Thus, Equation (2.15) renders,

$$\int_\Omega w\,[\nabla \cdot (\kappa \nabla u)]\ \mathrm{d}V = -\int_\Omega wf\ \mathrm{d}V \tag{2.18}$$

Integrating by parts reduces the order of differential equations by one and employing Divergence theorem gives,

$$\int_\Gamma w(\kappa\ \nabla u \cdot \mathbf{n})\ \mathrm{d}A - \int_\Omega \kappa \nabla w \cdot \nabla u \mathrm{d}V = -\int_\Omega wf\ \mathrm{d}V. \tag{2.19}$$

The resulting expression is less strict compared to Equation (2.15) and is thus called the *weak form* of the problem.

Similarly, the weak form of solid mechanics problem can be derived. For simplification, a steady state linear elasticity problem is considered by neglecting time derivative and non-linear terms in Equations (2.6) and (2.7). In this problem, the operator $\mathcal{L}$ represents linear function for Cauchy stress tensor $\boldsymbol{\sigma}$, expressed in terms of displacement vector $\mathbf{u}$, $\mathcal{L}(\mathbf{u}) = \nabla \cdot \boldsymbol{\sigma}(\mathbf{u})$. The strong formulation of the problem gives,

$$\nabla \cdot \boldsymbol{\sigma}(u) = -\mathbf{b} \qquad \forall\, \mathbf{x} \in \Omega \tag{2.20}$$

$$\mathbf{u} = \mathbf{u}_D \qquad \forall\, \mathbf{x} \in \Gamma_D \tag{2.21}$$

$$\boldsymbol{\sigma}(\mathbf{u}) \cdot \mathbf{n} = \mathbf{t}_N \qquad \forall\, \mathbf{x} \in \Gamma_N, \tag{2.22}$$

$$\boldsymbol{\sigma}(\mathbf{u}) = \lambda(\nabla \cdot \mathbf{u})\ \mathbf{I} + G\left[\nabla \mathbf{u} + (\nabla \mathbf{u})^T\ \right], \tag{2.23}$$

where

$\boldsymbol{\sigma}(\mathbf{u})$	– stress tensor
$\mathbf{b}$	– body forces vector
λ, G	– Lamé constants
$\mathbf{u}_D$	– prescribed displacement vector
$\mathbf{t}_N$	– prescribed traction forces vector.

The derivation of the weak form follows the same steps as in heat conduction problem and the resulting equation can be given as,

$$\int_\Omega \nabla \mathbf{w} : \boldsymbol{\sigma}(\mathbf{u})\ \mathrm{d}V = \int_{\Gamma_D} \mathbf{w} \cdot \mathbf{t}\ \mathrm{d}A + \int_\Omega \mathbf{w} \cdot \mathbf{b}\ \mathrm{d}V, \tag{2.24}$$

where

$$\int_\Omega \nabla \mathbf{w} : \boldsymbol{\sigma}(\mathbf{u})\ \mathrm{d}V = \int_\Omega \lambda(\nabla \cdot \mathbf{w})(\nabla \cdot \mathbf{u}) + 2\,G\ \nabla \mathbf{w} : \left(\frac{1}{2}\left[\nabla \mathbf{u}(\nabla \mathbf{u})^T\right]\right) \mathrm{d}V. \tag{2.25}$$

Same equations can be derived by following the *principle of potential energy*, where the weighting function in heat conduction and linear elasticity problem are called *virtual temperature* and *virtual work*, respectively.

Having the integral form of the weighted functions, one needs to substitute approximation u^h into Equations (2.24) and (2.25). Function u^h is defined piecewise employing the shape functions and coefficients at the nodes. The approximation of u is given as,

$$u(\mathbf{x}) \approx u^h(\mathbf{x}) = \sum_{j=1}^{n} u_j N_j, \tag{2.26}$$

where u_j are the coefficients to be identified and N_j are the shape functions.

Various methods have been developed for choosing the best test and shape functions. In current work only the Galerkin approach is presented. For other methods, for instance, *point collocation, least squares* and *Petrov-Galerkin*, refer to [19].

Employing Galerkin approach ($w_i = N_i$), the approximation u^h and the discretised weak form, Equations (2.12) - (2.14) (general boundary value problem) for single element render,

$$\sum_{j=1}^{n} \left[\int_{\Omega} w_i \mathcal{L} N_j \, \mathrm{d}V \right] u_j = \int_{\Omega} w_i \, f \, \mathrm{d}V + \int_{\Gamma} w_i \, q \, \mathrm{d}A, \tag{2.27}$$

where u_j is a vector of coefficients to be identified. The term in brackets on the LHS forms the stiffness matrix, and RHS forms the external forces vector.

So far a single element has been considered. For the clarity, superscript ^{e} was not added to each term, but will be hereon for any variable on the elementary basis.

One could concentrate the terms under integral and write Equation 2.27 as,

$$\sum_{j=1}^{n} w_i \, k^e_{ij} \, u^e_j = w_i \, f^e_i \; + w_i \, Q^e_i \quad (i = 1, 2, ..., n) \tag{2.28}$$

Instead of presenting the discretisation process of heat conduction and linear elasticity equations, the final, elementary stiffness matrix and external forces vector is given.

For heat conduction,

$$\mathbf{k}^e_{AB} = \int_{\Omega_e} \kappa \, \nabla N_A \cdot \nabla N_B \, \mathrm{d}V \tag{2.29}$$

$$\mathbf{p}^e_A = \int_{\Gamma_N \cap \Gamma_e} N_A \, q_N \, \mathrm{d}A + \int_{\Omega_e} N_A \, f \, \mathrm{d}V \tag{2.30}$$

and for linear elasticity,

$$\mathbf{k}^e_{AB} = \int_{\Omega_e} \lambda \, \nabla N_A \otimes \nabla N_B + G \, (\nabla N_A \cdot \nabla N_B \, \mathbf{I} + \nabla N_B \otimes \nabla N_A) \mathrm{d}V, \tag{2.31}$$

$$\mathbf{p}^e_A = \int_{\Gamma_N \cap \Gamma_e} N_A \, \mathbf{t} \, \mathrm{d}A + \int_{\Omega_e} N_A \mathbf{b} \, \mathrm{d}V. \tag{2.32}$$

2.3 Solution Procedure

In Section 2.2 a discretised weak form of the problem for single element was derived.

Galerkin formulation for linear problems can be written,

$$\sum_{i=1}^{n_{el}} w_A \, k^e_{AB} \, u_B = \sum_{i=1}^{n_{el}} w_A \, p^e_A, \tag{2.33}$$

which in matrix form gives,

$$\mathbf{w} \cdot \mathbf{K} \, \mathbf{u} = \mathbf{w} \cdot \mathbf{P}, \tag{2.34}$$

where $\mathbf{w}$ and $\mathbf{u}$ are vectors of nodal values of the weighting w function and the trial solution u^h. Vector $\mathbf{P}$ holds the nodal values of external forces.

For the global linear system of equations (LSE), the elementary equations are assembled,

$$\mathbf{K} = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \mathbf{k}^e, \qquad \mathbf{P} = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \mathbf{p}^e. \tag{2.35}$$

The connectivity matrix $\mathbf{T}$ is employed to define the locations of local stiffness matrix entries in the global system. Since $\mathbf{w}$ arbitrary, Equation 2.34 renders,

$$\mathbf{Ku} = \mathbf{P}. \tag{2.36}$$

For linear problems the stiffness matrix, $\mathbf{K}$ is constant and calculated only once. In case of non-linear governing equations, new stiffness matrix is calculated at each iteration by computing the derivative of residual with respect to the sought variable u. This procedure is called Newton-Raphson method and employed to accelerate the convergence of the solution. Linear systems converge in single iteration as stiffness matrix $\mathbf{K}$ does not depend on solution $\mathbf{u}$.

In real computer programs isoparametric elements are introduced to employ numerical integration (e.g Gauß quadrature) over complex elements. Jacobian (for isoparametric transformation) is calculated for the mapping to transfer data between the reference and real element.

Before solving the system $\mathbf{Ku} = \mathbf{P}$, the Dirichlet boundary conditions must be applied, otherwise the system is not closed and $\mathbf{K}$ is singular. Suppressing rows-columns, *Lagrange multipliers* and *penalty method* are the strategies to enforce Dirichlet boundary conditions (including dependent degrees of freedom). One of the goals of these methods is to remain the symmetry of $\mathbf{K}$ to reduce the computational cost of solving the LSE. For more information, refer to [2] or [19].

The general procedure of a finite element analysis software is [6]:

1. Definition of the geometry (nodal coordinates $\mathbf{x}$, connectivity matrix $\mathbf{T}$) and reference element.
2. Generate linear system of algebraic equations:
   - Loop over elements
     - Loop over integration points
       → compute Jacobian of the isoparametric transformation
       → compute derivatives of shape functions N_i in reference coordinates
       → find derivatives of shape functions N_i in global coordinates
       → compute contribution of every integration point to $\mathbf{k}^e$ and $\mathbf{f^e}$
   - assemble system, $\mathbf{Ku} = \mathbf{P}$
3. Apply Dirichlet type boundary conditions
4. Solve system $\mathbf{Ku} = \mathbf{P}$

Usually the solution procedure in a real computer program is controlled by macros, which call specific functions and procedures. The same method is employed in solid sub-solver of the current project. For more information on solution control in MPAP2, refer to Section 5.1.

Depending on the software, a preprocessor and postprocessor may be included to the main part of the finite element analysis computer program. Preprocessor is exploited to create geometry, generate the mesh, and define the boundary and initial conditions of the problem. Postprocessor is employed to compute the solution (using to nodal values and shape functions) and the derivative of it at desired points of the domain.

3 The Basics of Finite Volume Modelling

All the fluid flow simulations in this study have characteristic length and time-scales that are considerably larger compared to the scales of molecular structure of the matter. Thus, it is possible to express any macroscopic physical property of the matter as a continuous function.

The most distinct property of the fluid compared to solid, is its non-resistance to external shear forces. Even the smallest force causes deformation, which allows the fluid to flow.

Fluid flows are caused by externally applied loads:

- *surface forces*, including pressure and shear stress caused by sliding of a boundary wall
- *body forces* (e.g gravity)

Fluid flows can be classified according to many properties of fluid and the nature of flow as shown in the following Table, 3.1. Obviously, the list is not complete and it is not the only classification.

Flow types					Property
laminar		↔		turbulent	Reynolds number
incompressible		↔		compressible	Mach number
subsonic	↔	supersonic	↔	hypersonic	Mach number
viscous		↔		inviscid	distance from a wall
non-Newtonian		↔		Newtonian	viscosity dependence on the shear rate

Table 3.1: Classification of fluid flows based on various properties

In these flow types, some terms in the Navier-Stokes equations become dominant and others negligible. Thus many simplification and approximations of Navier-Stokes equations are developed, which correspond to a specific flow types, e.g incompressible flow, inviscid (Euler) flow, Potential flow, creeping (Stokes) flow, Boussinesq approximation and boundary layer approximation. These simplifications are not just in favour of finding an analytical solution in case of simple geometry of the domain, but also may decrease the computational cost as well as the need for the required computational resources (e.g memory, CPU).

The derivation of equations and algorithms in following sections is mainly based on the book “Computational Methods for Fluid Dynamics” [8] by Joel H. Ferziger and Milovan Perić, and on Hrvoje Jasak PhD thesis “Error Analysis and Estimation for Finite Volume Method with Applications to Fluid Flows” [10].

3.1 Governing Equations of Fluid Flow

In solid mechanics usually the conservation laws of extensive properties (mass, momentum and energy) are applied on a control mass (CM). In fluid mechanics it is more convenient to examine particular spatial region of flow, rather than to follow a parcel of matter. This is due to the deformations in fluid flow, which are usually much larger compared to solid mechanics.

In this study only incompressible flows (liquids or gases with Mach number below 0.3) are under investigation and thus only the conservation of mass and momentum will be viewed in this section. The conservation law of a property relates the rate of change of the amount of that property in a given control mass to externally determined effects. As mass cannot be created nor destroyed, the conservation equation gives

$$\frac{\mathrm{d}(m)}{\mathrm{d}t} = 0 \; . \tag{3.1}$$

But the momentum can be changed by action of external forces according to Newton's second law of motion

$$\frac{\mathrm{d}(m\mathbf{v})}{\mathrm{d}t} = \sum \mathbf{f}, \tag{3.2}$$

where

t	– time
m	– mass
$\mathbf{v}$	– velocity
$\mathbf{f}$	– forces acting on the control mass.

These conservation laws must be transformed into control volume (CV) form as one is regarding to a specific spatial region.

If one considers ϕ as any conserved intensive property ($\phi = 1$ for mass conservation and $\phi = \mathbf{v}$ for momentum conservation), then the corresponding extensive property Φ can be expressed as,

$$\Phi = \int_{\Omega_{CM}} \rho\phi \; \mathrm{d}\Omega_{CM}, \tag{3.3}$$

where Ω_{CM} is the volume occupied by the control mass.

Using Reynolds' transport theorem, each conservation equation can be written for a control volume as,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CM}} \rho\phi \; \mathrm{d}\Omega_{CM} = \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CV}} \rho\phi \; \mathrm{d}\Omega_{CV} + \int_{S_{CV}} \rho\phi(\mathbf{v} - \mathbf{v}_b)\cdot\mathbf{n} \; \mathrm{d}S_{CV}, \tag{3.4}$$

where

Ω_{CV}	– volume of CV
S_{CV}	– surface of CV
$\mathbf{n}$	– unit outward normal of CV
$\mathbf{v}$	– fluid velocity
$\mathbf{v}_b$	– velocity of the CV surface.

This Equation (3.4) states that the rate of change of the amount of a physical property in the control mass is equal to the rate of change of the property in the CV plus the net flux of this property trough the CV surface.

Considering control volumes hereon, the notation will be for clarification changed as follows: Ω_{CV} and S_{CV} to Ω and S, respectively. It is also assumed that the control volume is fixed in the space and thus $\mathbf{v}_b = 0$. This assumption also allows to replace the full time derivative (first term on the RHS) of the Equation (3.4)) with partial time derivative $\frac{\partial}{\partial t}$.

By setting $\phi = 1$, one obtains the integral form of the equation of mass conservation

$$\frac{\partial}{\partial t}\int_{\Omega} \rho \; \mathrm{d}\Omega + \int_{S} \rho\mathbf{v}\cdot\mathbf{n} \; \mathrm{d}S = 0, \tag{3.5}$$

which in differential form is

$$\frac{\partial}{\partial t}\rho + \nabla\cdot\rho\mathbf{v} = 0 \; . \tag{3.6}$$

To obtain the integral form of the momentum conservation equation, one needs to replace the arbitrary physical property ϕ by velocity $\mathbf{v}$ and use the Newton's second law of motion (Equation (3.2)),

$$\frac{\partial}{\partial t}\int_{\Omega} \rho\mathbf{v} \; \mathrm{d}\Omega + \int_{S} \rho\mathbf{v}\mathbf{v}\cdot\mathbf{n} \; \mathrm{d}S = \sum \mathbf{f}. \tag{3.7}$$

As stated before, the forces, acting on the fluid, are divided into surface and body forces. The surface forces can be expressed using stress tensor on the boundary, which for Newtonian fluid can be written as,

$$\mathbf{T} = -(p + \frac{2}{3}\mu\nabla\cdot\mathbf{v})\mathbf{I} + 2\mu\mathbf{D}, \tag{3.8}$$

where

μ	– dynamic viscosity
$\mathbf{I}$	– unit tensor
p	– static pressure
$\mathbf{D}$	– rate of deformation tensor,

which is given as,

$$\mathbf{D} = \frac{1}{2}[\nabla\mathbf{v} + (\nabla\mathbf{v})^T] \; .$$

Body forces per unit of mass are expressed by vector $\mathbf{b}$. Thus the overall integral form of the momentum equation becomes,

$$\frac{\partial}{\partial t}\int_\Omega \rho\mathbf{v}\,\mathrm{d}\Omega + \int_S \rho\mathbf{v}\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = \int_S \mathbf{T}\cdot\mathbf{n}\,\mathrm{d}S + \int_\Omega \rho\mathbf{b}\,\mathrm{d}\Omega. \tag{3.9}$$

Momentum conservation equation in differential form gives

$$\frac{\partial(\rho\mathbf{v})}{\partial t} + \nabla\cdot(\rho\mathbf{v}\mathbf{v}) = \nabla\cdot\mathbf{T} + \rho\mathbf{b}\ . \tag{3.10}$$

Mass and momentum conservation form the crucial part of Navier-Stokes equations. The full set of Navier-Stokes equations includes also energy conservation, which is not considered in this work as the flow is assumed to be isothermal and incompressible.

3.2 Spatial Discretisation Using Finite Volume Formulation

The starting point for FV methods are the integral form of the conservation equations (see Chapter 3.1). The domain is divided into a finite number of control volumes, on which the conservation equations are applied. The actual computational node, where the variable values are calculated, lies at the centroid of each CV. To evaluate the variable at a surface of a CV, interpolation of two adjacent CVs is used. Quadratures are used to find a surface and volume integrals of a variable over a CV. As a result, an algebraic formulae containing the number of adjacent CVs respect to CV under investigation, is derived for each cell in the domain.

Finite volume method is conservative as the convective and diffusive fluxes (surface integrals) are equal for cells that share the same face. The number of flat surfaces forming a control volume is not limited, allowing to generate mesh for complex domains with reasonably good quality.

The advantages of the FV method is its conservativeness (on local as well as on global level) and that all the approximations have physical meaning (e.g surface integral - flux).

The main disadvantage of FV methods is related to difficulties of achieving higher, than second order accuracy, due to three levels of approximation: interpolation, differentiation and integration (especially with unstructured grids).

Many domain discretisation methods have been developed for finite volumes method (see for example page 71 in "Computational Methods for Fluid Dynamics" [8]), but in this work the focus is on discretisation schemes, used in OpenFOAM. The nodes on the grid are placed at the centroid of the control volume (other approach would place nodes to equal distance from face). Moreover, OpenFOAM uses *collocated variable arrangement*, which means that all the variables are calculated at the same location - at the cell centroids. This arrangement became popular in early 1980's, when semi-implicit methods for pressure-velocity coupling were developed. Another common variable arrangement places the pressure variables in the centre of the cell and the velocity in the face centres or to vertices of the cell. That variable arrangement is known as *staggered approach.*

A typical finite two-dimensional (2D) non-uniform volume grid can be seen in the Figure 3.1.

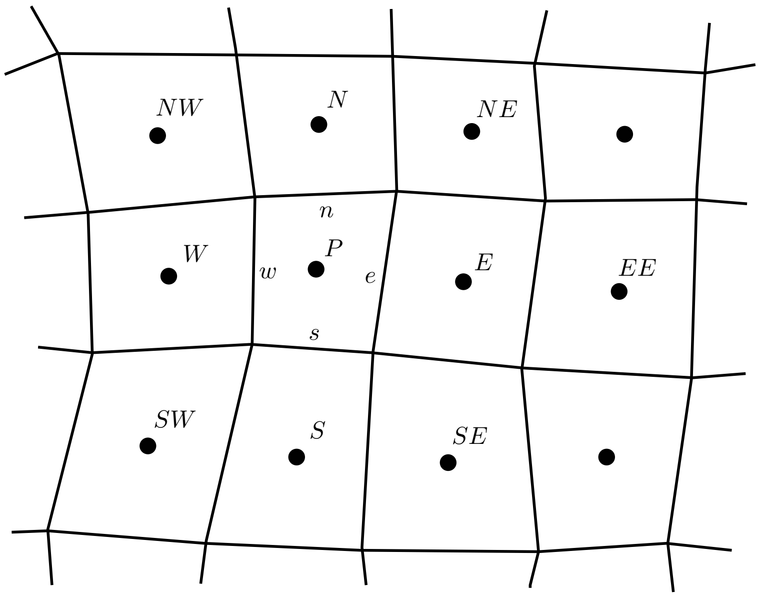


Figure 3.1: Arbitrary 2D finite volumes grid with notations

All the variables are calculated at the filled circles. The cell under investigation is noted P and the adjacent cells (cells that share faces with cell P) are marked either N, S, W or E, according to their position in respect to cell P. Capitals denote cells and lower case letters, faces (n – north, s – south, w – west, e – east).

Before finding the solution for Navier-Stokes equations, one needs to look at the discretisation methods of following terms (operators): gradient, divergence and laplacian. Generic scalar transport equation is considered as a good example for conservation equations. Following this, general transport equation approach, it is assumed that the fluid velocity field and the fluid properties are known. The velocity and pressure field can be found by solving Navier-Stokes equations, which is viewed after describing the discretisation methods in Sections 3.2.1 - 3.3. General transport equation is exploited to describe the idea of finite volume discretisation. This equation has all the terms that appear in a conservation equations, but it is simpler, because of the linearity and the transported physical quantity is scalar.

A generic scalar transport equation reads,

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\phi \, \mathrm{d}\Omega + \int_{S}\rho\phi\mathbf{v}\cdot\mathbf{n}\, \mathrm{d}S = \int_{S}\Gamma_{\phi}\nabla\phi\cdot\mathbf{n}\, \mathrm{d}S + \int_{\Omega}S_{\phi}\, \mathrm{d}\Omega, \tag{3.11}$$

where S_ϕ is a source or a sink and Γ_ϕ is the diffusivity of ϕ.

The differential form of an arbitrary scalar transport Equation (3.11) gives,

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla\cdot(\rho\phi\mathbf{v}) = \nabla\cdot(\Gamma_\phi\nabla\phi) + S_\phi. \tag{3.12}$$

3.2.1 Approximation of Surface Integrals

Equation (3.11) contains many surface integral terms and thus an approximation of these terms must be found.

The net flux through the boundary of a CV is sum of integrals over all the faces of the cell

$$\int_S f\, \mathrm{d}S = \sum_k \int_{S_k} f\mathrm{d}S, \tag{3.13}$$

where f is the normal to CV face component of flux - convective ($\rho\phi\mathbf{v}\cdot\mathbf{n}$) or diffusive ($\Gamma_\phi\nabla\phi\cdot\mathbf{n}$) and k, number of faces of the cell.

To evaluate the Equation (3.13) exactly, one has to know the value of the integrand f at every point at the surface S_k. As only the values at CV centres are known, an approximation has to be carried out. It involves two levels:

- the integral is approximated using one (or more) discrete values on the face
- the face values are calculated using nodal values of adjacent cells

The simplest face integral approximation multiplies the single value at the face centre (approximation of the mean value over the face) by the area of the face (mid-point rule). Thus the surface integral for face e becomes

$$F_e = \int_{S_e} f\mathrm{d}S = \bar{f}_e S_e \approx f_e S_e, \tag{3.14}$$

where

S_e	– area of the face
$\bar{f}_e$	– mean value of integrand
f_e	– component of flux vector at face centre in direction of face normal.

This approximation is second-order accurate only if the variable f_e at face centre has been found using scheme, which is at least second order accurate. Interpolation is used to find the value of the variable at face centre and its algorithm will be given later.

Higher order approximations of surface integral are available if the variable is known at more locations on a single face (e.g at vertices of the face). As this is not used in OpenFOAM, the schemes will not be presented in this paper. For more information, please refer to page 74 in the textbook [8].

3.2.2 Approximation of Volume Integrals

Referring again to Equation (3.11), one can see the need for approximation of volume integrals.

Similarly to surface integral, the simplest second-order volume integral approximation is a product of the mean value of integrand and the volume of the cell. Given that the variable at point P (see Figure 3.1) is an approximation of the mean over a cell, the overall approximation of the volume integral can be given as,

$$Q_P = \int_\Omega q \, \mathrm{d}\Omega = \bar{q} \; V_\Omega \approx q_P V_\Omega, \tag{3.15}$$

where q_P is the value of q at cell centre. Replacing mean value $\bar{q}$ by q_P generates an exact scheme for the volume integral only if the distribution of q is constant or linear over the cell.

Higher order volume integral approximations require values of q at more locations than just at the centre of the cell. OpenFOAM uses only single value at the cell centre and thus higher order schemes are not discussed here. For more information of the higher order schemes, one can refer to page 75 in the textbook [8].

3.2.3 Interpolation and Differentiation Schemes

The integrand f used in Equations (3.13) and (3.14) denotes convective ($f^c = \rho\phi\mathbf{v}\cdot\mathbf{n}$) or diffusive ($f^d = \Gamma_\phi \nabla\phi\cdot\mathbf{n}$) flux in the general transport equation (Eq. 3.11)

In order to evaluate these fluxes, one has to know the value of ϕ and its gradient normal to the cell-face. Interpolation is used to express ϕ at a face in terms of cell nodal values.

Numerous schemes have been developed for the best approximation of ϕ at the face. As higher than second order schemes include values not just from the adjacent cells, but from cells further away, the implementation for unstructured grids becomes complex and will not be discussed here.

Upwind Differencing (UD) is the simplest bounded scheme available. It is first order accurate and takes its face value from cell centre that is located upstream

$$\phi_e = \begin{cases} \phi_P & \text{if}(\mathbf{v}\cdot\mathbf{n})_e > 0; \\ \phi_E & \text{if}(\mathbf{v}\cdot\mathbf{n})_e < 0 \end{cases} \tag{3.16}$$

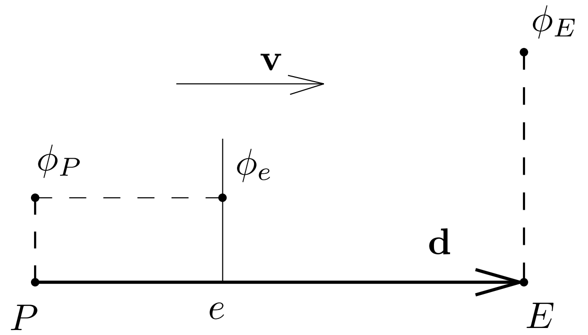


Figure 3.2: Upwind Differencing

The unconditional stability is achieved by numerical diffusion, which the scheme contains. To find the magnitude of the diffusion, one has to compare Equation (3.16) to Taylor expansion around P,

$$\phi_e = \phi_P + (x_e - x_P)\left(\frac{\partial \phi}{\partial x}\right)_P + \frac{(x_e - x_P)^2}{2}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_P + H, \tag{3.17}$$

where H denotes higher order terms.

It can be seen that the approximation equals just the first term on RHS of Taylor expansion and the leading truncation error is diffusive, which is given,

$$f_e^d = \Gamma_e \left(\frac{\partial \phi}{\partial x}\right)_e,$$

where $\Gamma_e = (\rho u)_e \Delta x/2$ (for one-dimensional problem).

Linear interpolation or *central differencing* (CD) is the next common approximation for the variable at face centre. It assumes linear distribution between two adjacent cell centres and is given as,

$$\phi_e = \phi_E \lambda_e + \phi_P(1 - \lambda_e), \tag{3.18}$$

where λ_e is the distance ratio given as,

$$\lambda_e = \frac{x_e - x_P}{x_E - x_P}, \tag{3.19}$$

where $x_E - x_P = |\mathbf{d}|$ considering the notation in Figure 3.3.

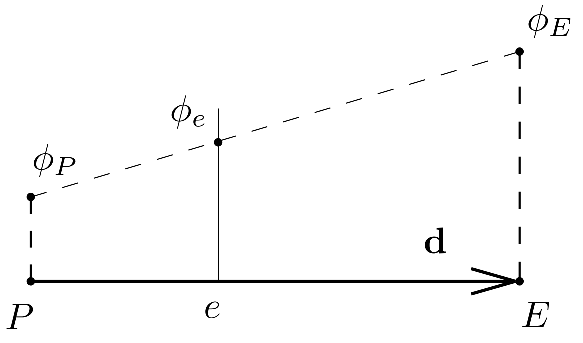


Figure 3.3: Central Differencing

Taylor series expansion of ϕ_E about point x_P gives

$$\phi_E = \phi_P + (x_E - x_P)\left(\frac{\partial \phi}{\partial x}\right)_P + \frac{(x_E - x_P)^2}{2}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_P + H. \tag{3.20}$$

This expression can be used to evaluate the derivative at x_P

$$\begin{aligned}\left(\frac{\partial \phi}{\partial x}\right)_P &= \frac{\phi_E - \phi_P}{x_E - x_P} - \frac{x_E - x_P}{2}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_P + H \\ &= \lambda_e \frac{\phi_E - \phi_P}{x_e - x_P} - \frac{x_E - x_P}{2}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_P + H.\end{aligned} \tag{3.21}$$

By substituting the derivative in Equation (3.17) gives

$$\phi_e = \phi_E \lambda_e + \phi_P(1 - \lambda_e) - \frac{(x_e - x_P)(x_E - x_e)}{2}\left(\frac{\partial^2 \phi}{\partial x^2}\right)_P + H, \tag{3.22}$$

which proves central differencing being second order accurate.

Blended differencing is a linear combination of two previous schemes with an aim to maintain the boundedness of upwind differencing and accuracy of central differencing. It can be written using face values from previous schemes

$$\phi_e = (1 - \gamma)(\phi_e)_{UD} + \gamma(\phi_e)_{CD} \tag{3.23}$$

or in terms of the nodal values

$$\begin{aligned}\phi_e =& [(1 - \gamma)max(sgn(\mathbf{v} \cdot \mathbf{n}), 0) + \gamma\lambda_e]\phi_P + \\ & [(1 - \gamma)min(sgn(\mathbf{v} \cdot \mathbf{n}), 0) + \gamma(1 - \lambda_e)]\phi_E,\end{aligned} \tag{3.24}$$

where γ, $0 < \gamma < 1$, is blending factor, which defines the amount of numerical diffusivity in the scheme.

Many variations of upwind and central differencing schemes have been developed. One can find all available interpolation schemes of OpenFOAM from the software's user guide [18].

Higher order schemes that involve larger computational molecule than just adjacent cells, are described in the literature. Most of them are developed having structured (sometimes even uniform) grids in mind. Some of these algorithms are also available in OpenFOAM (see user guide [18]). Anyhow, the overall accuracy remains second order, because of the scheme used for approximation of the surface integrals.

3.2.4 Diffusion Term and aspects of non-orthogonality

Assuming linear variation of ϕ between nodes P and E, to evaluate the diffusive term, $f^d = \Gamma_\phi \nabla\phi \cdot \mathbf{n}$, the simplest approximation reads

$$\left(\frac{\partial \phi}{\partial x}\right)_e \approx \frac{\phi_E - \phi_P}{x_E - x_P} = \frac{\phi_E - \phi_P}{|\mathbf{d}|}. \tag{3.25}$$

It can be proven that the approximation (3.25) is second order accurate only if the face is exactly at the midway between the nodes P and E. Moreover, one has to bear in mind that particular expression defines the gradient between the nodes, not normal to the face.

The alternative is to find cell-centred gradients of cells (exploiting Green-Gaußprocedure) shearing the face as,

$$(\nabla\phi)_P = \frac{1}{V_P}\sum_k \int_{S_k} \phi_f \ \mathrm{d}\mathbf{S} = \frac{1}{V_P}\sum_k \phi_{f_k}\mathbf{S}_k \tag{3.26}$$

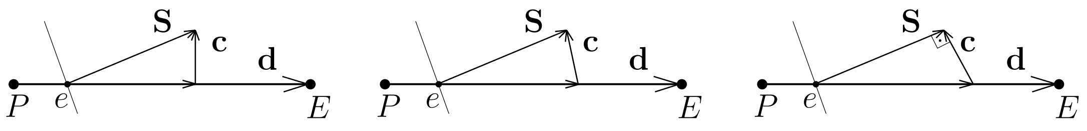
and then use linear interpolation:

$$(\nabla\phi)_e = \lambda_e(\nabla\phi)_P + (1-\lambda_e)(\nabla\phi)_N. \tag{3.27}$$

The latter method is less accurate, but can be used also for non-orthogonal grids.

The optimal solution, which is also implemented in OpenFOAM, would be to use both of Equations (3.25) and (3.27).

This method is based on the decomposition of face area vector into orthogonal and non-orthogonal part. Three different decomposition strategies can be employed, which are shown in Figure 3.4



(a) Minimum correction (b) Ortohogonal correction (c) Over-relaxed

Figure 3.4: Various approaches for non-orthogonality correction

All these approaches are second order accurate and satisfy the decomposition expression,

$$\mathbf{S} = \mathbf{b} + \mathbf{c}, \tag{3.28}$$

where $\mathbf{S}$ is the area vector of a face.

The approximation that is more accurate (Eq. 3.25) is used together with vector $\mathbf{b}$ which is parallel to vector $\mathbf{d}$ and the approximation, based on cell-centred gradients will be used as corrector together with vector $\mathbf{c}$. Thus the overall face normal gradient becomes

$$(\nabla\phi \cdot \mathbf{S})_e = |\mathbf{b}|\frac{\phi_E - \phi_P}{|\mathbf{d}|} + \mathbf{c} \cdot (\nabla\phi)_e, \tag{3.29}$$

where $(\nabla\phi)_e$ is calculated using Equation (3.27).

This procedure ensures optimal solution, as less accurate approximation is used together with shorter(in general) vector $\mathbf{c}$.

3.3 Temporal discretisation and methods for unsteady problems

The main classification of the discretisation methods for the time derivative terms are the *explicit* and the *implicit* method. The first of them uses just the values from the past, while implicit method includes also values from the next time-step and can only be solved iteratively. There are four main discretisation methods for time derivatives. These are the source for many other combined methods, which are not described in the scope of this study. One can refer to any numerical methods textbook for further information.

Lets consider a first order ordinary differential equation with an initial condition

$$\frac{\mathrm{d}\phi(t)}{\mathrm{d}t} = f(t, \phi(t)); \qquad \phi(t_0) = \phi^0 \tag{3.30}$$

The aim is to find value of ϕ after a small time increment Δt. The simplest way to construct algorithm for ϕ^{n+1} is to integrate Equation (3.30) form time instant t_n to time instant t_{n+1}.

$$\int_{t_n}^{t_{n+1}} \frac{\mathrm{d}\phi}{\mathrm{d}t}\mathrm{d}t = \phi^{n+1} - \phi^n = \int_{t_n}^{t_{n+1}} f(t, \phi(t))\mathrm{d}t \tag{3.31}$$

As ϕ^{n+1} is unknown, one can not substitute previous expression directly to transport equation, but needs to find an approximation of ϕ^{n+1}.

Four most basic and well-known methods are presented in following paragraphs and shown graphically in Figures 3.5 and 3.6.

If one uses only the value of the integrand at previous time-step, one achieves method known as *explicit* or *forward Euler*,

$$\phi^{n+1} = \phi^n + f(t_n, \phi^n)\Delta t. \tag{3.32}$$

This scheme becomes unstable as Courant number becomes larger than unity. Courant number can be calculated as,

$$Co = \frac{\mathbf{v}_e \cdot \mathbf{d}}{\Delta t}, \tag{3.33}$$

where $\mathbf{v}_e$ is the fluid velocity at face e, $\mathbf{d}$ is the vector from node P to node E and Δt is the time-step size.

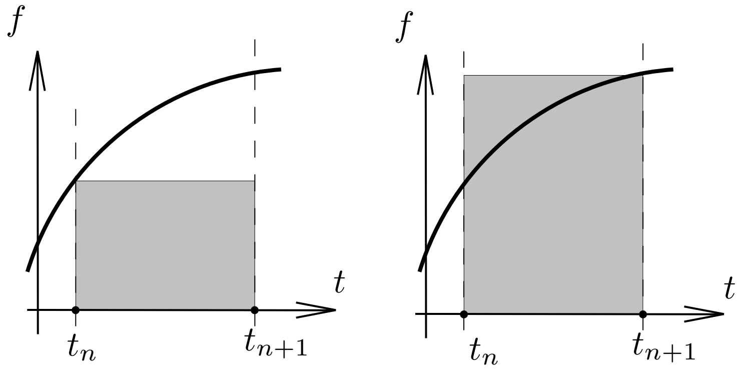


Figure 3.5: Forward and backward Euler approximation of time integral

Secondly, one could use the integrand at the new point t^{n+1} to estimate the integral

$$\phi^{n+1} = \phi^n + f(t_{n+1}, \phi^{n+1})\Delta t. \tag{3.34}$$

Method presented by previous equation (3.34) is called *implicit* or *backward Euler*.

Third method uses midpoint between t_n and t_{n+1} and is named accordingly - the *midpoint rule*. It is a semi-implicit method, as it must be solved iteratively, although not at time instant t_{n+1}, but at $t_{n+\frac{1}{2}}$.

$$\phi^{n+1} = \phi^n + f(t_{n+\frac{1}{2}}, \phi^{n+\frac{1}{2}})\Delta t. \tag{3.35}$$

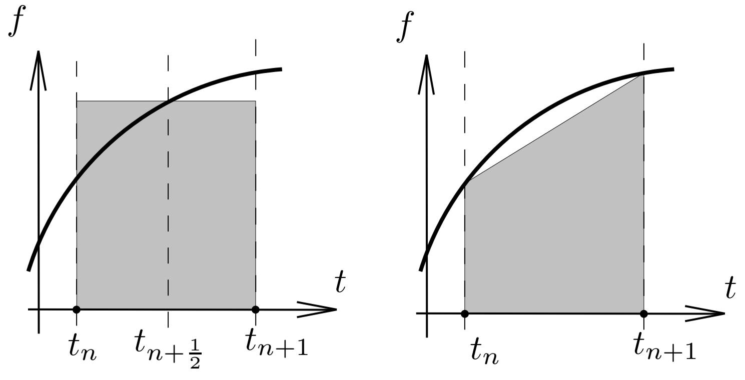


Figure 3.6: Midpoint and trapezoidal rule for approximating time integral

For the last method both, the start and the end point, is used for approximating the integral. This implicit method is called the *trapezoidal rule* and is given as,

$$\phi^{n+1} = \phi^n + \frac{1}{2}[f(t_n, \phi^n) + f(t_{n+1}, \phi^{n+1})]\Delta t. \tag{3.36}$$

All four methods, listed previously, are known as two-level methods. Both of the Euler schemes are first order accurate, while the midpoint and trapezoidal rule are second order. Although the stability depends largely on the time-step size, the backward Euler is considered the most stable and forward Euler the least.

Besides these schemes, there are many other, which are derived either by combining previous ones (*predictor-corrector methods*) or adding more points were the function is evaluated (*multipoint methods*)

As most of these schemes can be viewed as extension of the previous four methods, the stability largely depends on the contribution from implicit Euler with respect to contribution from other methods.

By combining temporal disretisation with spatial, solution methods for unsteady transport equations are derived. Although many modification and naming conventions can be found in the literature [8] [17] , the most common methods are shown in Table 3.2.

Temporal	Spatial		CFD scheme
forward Euler	+ upwind differencing	=	forward time centred space or explicit Euler method
midpoint rule	+ central differencing	=	leapfrog method
backward Euler	+ central differencing	=	implicit Euler method (one of Crank-Nicolson methods)
trapezoidal rule	+ central differencing	=	Crank-Nicolson Method

Table 3.2: Schemes of finite volume method CFD codes

3.4 Solution to Navier-Stokes Equations

3.4.1 Derivation of pressure equation

The differential form of Navier-Stokes equations were stated in Section 3.1, which for incompressible ($\rho = const$) flows becomes

Mass conservation:

$$\nabla \cdot \mathbf{v} = 0 \tag{3.37}$$

Momentum conservation:

$$\frac{\partial \mathbf{v}}{\partial t} + \nabla \cdot (\mathbf{v}\mathbf{v}) - \nabla \cdot (\nu \nabla \mathbf{v}) = -\nabla p. \tag{3.38}$$

In general Navier-Stokes follow the same discretisation rules as the generic transport equation, with exceptions that Navier-Stokes equations are vector equations, non-linear and there is a pressure term, which does not have analog from the generic equations.

Firstly, the non-linear term $\nabla \cdot (\mathbf{v}\mathbf{v})$ or in integral form, $\int_S \mathbf{v}\mathbf{v} \cdot \mathbf{n} \, \mathrm{d}S$, is under investigation. Using Equations (3.14) and (3.13) this convective term can be discretised as,

$$\begin{aligned}\nabla \cdot (\mathbf{v}\mathbf{v}) &= \sum_k \mathbf{S}_k \cdot (\mathbf{v})_k (\mathbf{v})_k \\ &= \sum_k F_k (\mathbf{v})_k \\ &= a_P \mathbf{v}_P + \sum_k a_k \mathbf{v}_k,\end{aligned} \tag{3.39}$$

where N represents the central cell and k any neighbouring cell. Moreover, F, a_P and a_k are function of $\mathbf{v}$.

In order to find solution to Equation (3.39), either a non-linear equation solver must be used or the equation has to be linearised beforehand. Taking into account computational cost of the non-linear solver, linearisation is preferred. In terms of Equation (3.39), linearisation means that coefficient a_P and a_k are calculated using the velocity field from the previous iteration, which did satisfy the continuity Equation (3.37).

As one can see, pressure term is present only in the momentum equation, but this equation is used for the calculating the velocity field $\mathbf{v}$. In fact, the mass conservation equation can be viewed as a constraint on the velocity field, rather than an individual equation for a specific variable. Thus a scheme to calculate pressure field, which satisfies the continuity equation must be derived. There are two approaches for that, which are basically the same, but involve different explanation.

One way is to find the derivative of the momentum equation and then cancel out all the terms that are carrying $\nabla \cdot \mathbf{v}$, because the continuity equation still holds. The outcome of this procedure will be a Poisson equation for the pressure. This approach is presented in textbook by Ferziger and Perić [8].

Another way to describe the same procedure is to evaluate from the momentum the velocity at a face and substitute it into the continuity equation. Hrovje Jasak has used this approach in his PhD thesis [10] and will also be presented in this study.

The starting point to derive velocity-pressure coupling, is semi-discretised momentum equation

$$a_P \mathbf{v}_P = \mathbf{H}(\mathbf{v}) - \nabla p, \tag{3.40}$$

where pressure gradient is left undiscretised and $\mathbf{H}(\mathbf{v})$ holds the rest of the terms of momentum equation that are not give in Equation (3.40). Term $\mathbf{H}(\mathbf{v})$ can be expressed as

$$\mathbf{H}(\mathbf{v}) = -\sum_k a_k \mathbf{v}_k + \frac{\partial \mathbf{v}}{\partial t} - \nabla \cdot (\nu \nabla \mathbf{v}). \tag{3.41}$$

Using Equation (3.5) for incompressible fluid and the discretisation method from Equation (3.13), the continuity equation can be given as

$$\nabla \cdot \mathbf{v} = \sum_k \mathbf{S}_k \cdot \mathbf{v}_f = 0 \tag{3.42}$$

To substitute momentum equation into mass conservation equation, one needs to evaluate $\mathbf{v}_P$ from Equation (3.40)

$$\mathbf{v}_P = \frac{\mathbf{H}(\mathbf{v})}{a_P} - \frac{1}{a_P} \nabla p. \tag{3.43}$$

Velocities at faces can be evaluated through any interpolation scheme, prescribed in Section 3.2.3 and is expressed for face k as,

$$\mathbf{v}_k = \left(\frac{\mathbf{H}(\mathbf{v})}{a_P} \right)_k - \left(\frac{1}{a_P} \right)_k (\nabla p)_k . \tag{3.44}$$

By substituting previous expression into Equation (3.42) gives

$$\begin{aligned} \nabla \cdot \left(\frac{1}{a_P} \nabla p \right) &= \nabla \cdot \left(\frac{\mathbf{H}(\mathbf{v})}{a_P} \right) \\ &= \sum_k \mathbf{S}_k \cdot \left(\frac{\mathbf{H}(\mathbf{v})}{a_P} \right)_k . \end{aligned} \tag{3.45}$$

The final form of the discretised Navier-Stokes equations for incompressible flow are

The conservation of momentum:

$$a_P \mathbf{v}_P = \mathbf{H}(\mathbf{v}) - \sum_k \mathbf{S}_k (p)_k \tag{3.46}$$

and the Poisson pressure (continuity eq.) equation:

$$\sum_k \mathbf{S}_k \cdot \left[\left(\frac{1}{a_P} \right)_k (\nabla p)_k \right] = \sum_k \mathbf{S}_k \cdot \left(\frac{\mathbf{H}(\mathbf{v})}{a_P} \right)_k . \tag{3.47}$$

3.4.2 Solution to pressure-velocity coupling

In this section the solution strategies for Equations (3.46) and (3.47) are investigated.

In general, the solution methods can be divided into *simultaneous* and into *segregated* algorithms.

For the *simultaneous* algorithm, the resulting matrix is significantly larger than the number of computational points in the domain, due to the inter-equation coupling. This scheme is barely ever used because of its computational cost and memory requirements.

Segregated algorithms solve the Equations (3.46) and (3.47) in sequence, with special treatment for the inter-equation coupling. The most used algorithms are *PISO* (Pressure Implicit with Splitting of Operators), *SIMPLE* (Semi-Implicit Mehod for Pressure-Linked Equations) and derivations of these algorithms. Despite the fact that these schemes are implicit, the matrix to be solved is much smaller compared to *simultaneous* algorithms. Furthermore, in real computer programs, the alternating direction implicit (ADI) method is used to split the momentum equations into series of one dimensional problems, each of which is block tridiagonal. This approach is also used in OpenFOAM.

Originally PISO scheme is developed for transient problems with assumption of small time-steps. The solution procedure is described by following steps:

1. The momentum Equation (3.46) is solved for $\mathbf{v}$ using pressure field from previous time instant, n. In general, the resulting velocity field, $\mathbf{v}^*$, does not satisfy the continuity equation and thus is denoted with an asterisk, "$*$" and called a prediction.

2. Using the predicted velocities, the variable $\mathbf{H}(\mathbf{v})$ in Equation (3.47) can be evaluated and new pressure field calculated.

3. Having the new pressure field, the velocity field can be corrected explicitly using Equation (3.43). Observing the Equation (3.43), it can be seen that the correction of velocity field depends on the change in pressure gradient ($\frac{1}{a_P}\nabla p$ term) and on the influence of the velocity correction in neighbouring cells

($\frac{\mathbf{H}(\mathbf{v})}{a_P}$ term). For this case, "explicit" means that term, $\mathbf{H}(\mathbf{v})$ in Equation (3.43) is calculated using the predicted velocity field, $\mathbf{v}$, and the velocity correction is neglected. Obviously, such cancellation is inadmissible and the steps 2 and 3 must be repeated until required tolerance is reached. Thus, despite some steps in correction procedure are explicit, the scheme in total is considered as implicit.

In short, PISO algorithm consists of single implicit momentum predictor followed by series of pressure evaluations and explicit velocity corrections. In OpenFOAM the number of correction loops is specified by defining `nCorrectors` in file `fvSolution`.

SIMPLE algorithm is originally developed for steady state flows, where the initial and final flow fields diverge significantly. The steps of the algorithm are:

1. Using under-relaxation for velocity, the momentum Equation (3.46) is solved for $\mathbf{v}$. Previous time-step values are used to find the pressure gradient.

2. The pressure Poisson Equation (3.47) is solved to find current pressure field.

3. Similarly to PISO algorithm, the pressure equation contains term $\mathbf{H}(\mathbf{v})$, where the correction of the velocity field is not taken into account. Instead of looping over steps 2 and 3, pressure is solved only once and the corrected velocity field is taken into account by using under-relaxation factor.

$$p^{n+1} = p^{n*} + \alpha_p(p^{n*} - p^n), \tag{3.48}$$

where p^{n*} is the pressure found in step 2. As SIMPLE algorithm is mainly used for steady state flows, $n-1$, n and $n+1$ may denote not the time-step, but iteration iteration step.

In literature, it is proven that the optimum value for under-relaxation factor for pressure is given as,

$$\alpha_p = 1 - \alpha_v, \tag{3.49}$$

where α_v is the relaxation factor for momentum equation. Moreover, $\alpha_p = 0.2$ and $\alpha_p = 0.8$ are the recommended values.

3.5 Calculation of traction forces

An essential part of a FSI solver is the calculation of traction forces, generated by the fluid. It consist of computing pressure and viscous forces.

In finite volume method, the variables (pressure and velocity) are calculated, and saved for the next time instant, at cell centres and at the face centres for the boundary. Thus the pressure field on any boundary is available after Navier-Stokes equations are solved and the traction force due to pressure can be found by multiplying face area vector by the pressure value at the centre of that face.

The calculation of viscous forces is slightly more complex. Assuming incompressible flow the Equation (3.8) renders,

$$\mathbf{T} = -p\mathbf{I} + 2\mu\mathbf{D}. \tag{3.50}$$

As the pressure is already calculated, the viscous stresses tensor is given as,

$$\boldsymbol{\tau} = 2\mu\frac{1}{2}\left[\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right] \ . \tag{3.51}$$

Similarly to pressure, the velocity field of the domain is available. The gradient of the velocity field can be found using Equation (3.25) and viscous stress tensor calculated using Equation (3.51). Finally, the same procedure as calculating the traction for pressure follows - viscous stress tensor at the face centre is multiplied by the area vector.

In order to minimise the number of operations, the pressure and viscous force vectors are added and only one traction vector is passed to solid (see Section 5.3 instruction 7).

4 Computational Fluid-Structure Interaction

In coupled multiphysics problems, such as FSI, any change in one subsystem causes a response in other subsystems.

In most fields of engineering the analysis types can be divided into three main categories:

- analytical
- computational
- experimental.

The same applies to fluid-structure interaction.

An analytical solution can be found to very a limited set of problems. However, these are substantial not just in industrial applications, but also in order to understand the problem and for the validation of numerical solvers.

Experimental analysis is the most trustful and can be used for the validation of numerical analysis, but the limiting factors are the cost and in some cases the constraints on scaling, such as, for example, satisfying Reynolds and Froude number simultaneously in a scaled towing test.

The computer simulation of fluid-structure interaction requires the discretisation of the solid and fluid domains. The resulting algebraic systems must be solved simultaneously or in turns.

To ensure the clarity, modularity and the flexibility of a FSI solver it is common to divide it into three parts: *fluid*, *solid* and *interface* [4]. This approach collects all additional calculations, such as the transfer of traction forces and kinematic data. Graphical representation of this modular approach is shown in Figure 4.1.

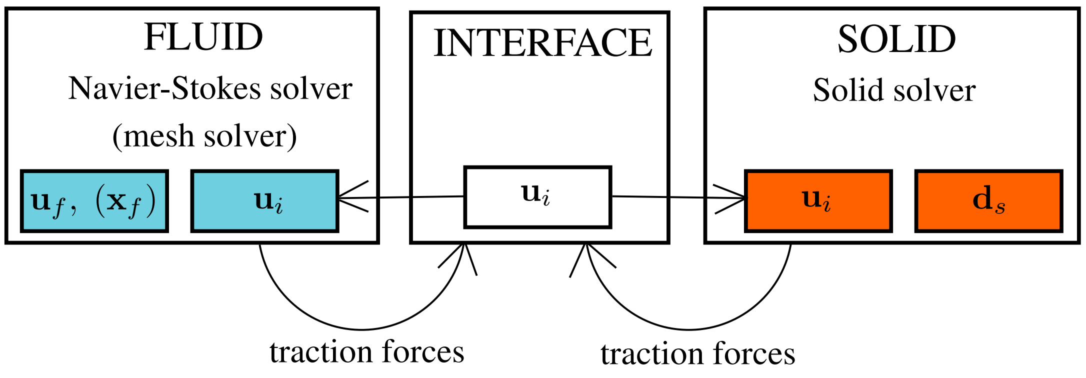


Figure 4.1: Structure of a FSI solver [4]

In Figure 4.1 the symbols denote following degrees of freedom:

$\mathbf{u}_i$ –	kinematic DOF of the interface
$\mathbf{u}_f$ –	all DOF of the fluid (velocity and pressure), except kinematic DOF on the interface,
$\mathbf{x}_f$ –	degrees of freedom of the fluid mesh
$\mathbf{d}_s$ –	degrees of freedom of the solid (nodal displacements)

Moreover, the introduction of *interface* domain simplifies data handling of non-matching meshes and allows for independent remeshing of sub-domains.

A system of equations that corresponds to previous FSI solver structure, is given as,

$$
\begin{aligned}
\mathbf{r}_f(\mathbf{u}_f, \mathbf{u}_i) &= \mathbf{0} && \text{(fluid);} \\
\mathbf{g}_f(\mathbf{u}_f, \mathbf{u}_i) + \mathbf{g}_s(\mathbf{d}_s, \mathbf{u}_i) &= \mathbf{0} && \text{(interface);} \\
\mathbf{r}_s(\mathbf{d}_s, \mathbf{u}_i) &= \mathbf{0} && \text{(solid).}
\end{aligned}
\tag{4.1}
$$

These equations are solved at each time-step for $\mathbf{u}_i$, $\mathbf{u}_f$ and $\mathbf{d}_s$. The vectors $\mathbf{r}_f$, $\mathbf{r}_s$ denote the nodal residual forces of the fluid and solid sub-domains, whereas $\mathbf{g}_f$, $\mathbf{g}_s$ represent the traction forces exerted on the interface by the fluid and the solid, respectively.

This system of equations corresponds to Newton-Raphson schemes and is solved as a whole (*monolithic*) or in turns (*partitioned*). These schemes provide generally good robustness and accuracy, but are computationally expensive and difficult to implement.

Besides Newton-Raphson schemes, Gauß-Seidel and staggered schemes have been developed [4].

These four types of computational FSI solvers are described in following Section, 4.1, and the staggered scheme, which is the focus of current project, is described in Section 4.2.

4.1 Classification of Fluid-Structure Interaction Solvers

Before categorising FSI solvers according to the strategy, possible interface formulations are reviewed. Two main types of defining the interface between fluid and solid are:

- interface capturing
- interface tracking

In the first case, the solid and fluid mesh are moving independently, *i.e.*, they slide over each other. It is mainly used with Eulerian fluid mesh. This method does not set any constraints on the scale of deformations (or rotations), but is less accurate in terms of satisfying the conservation equations.

In interface tracking, the fluid and structure mesh are moving together - there occurs neither overlapping nor gaps between the sub-domains. It is considered more accurate, but unsuitable for large deformations (remeshing needed). In order to preserve the quality of fluid mesh, additional computation is required to solve for the nodal coordinates.

In current project only the interface tracking method is used.

As a next step, the solution scheme must be chosen. A graphical representation of various FSI solvers, based on the strategy, is presented in Figure 4.2.

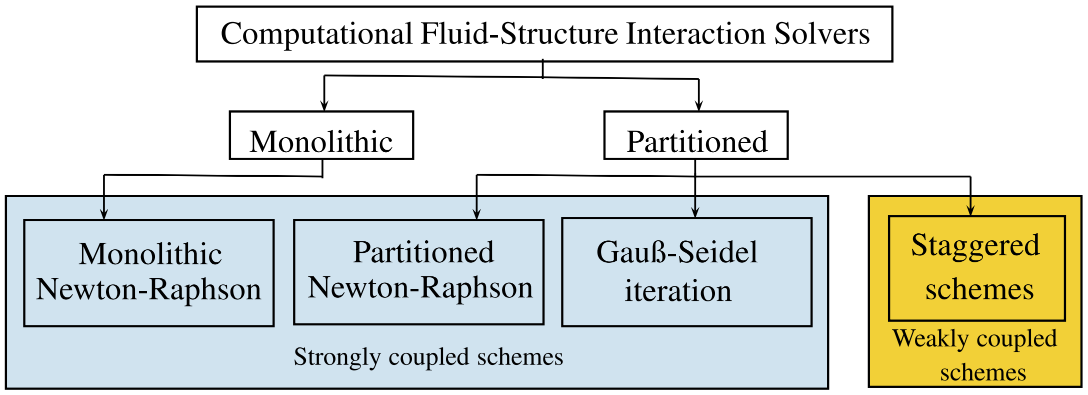


Figure 4.2: Classification of FSI solvers

Strongly coupled solvers satisfy the equilibrium of traction forces exactly. They comprise monolithic and partitioned Newton-Raphson as well as Gauß-Seidel strategies. For monolithic schemes, the DOF from all the sub-systems (fluid, solid, interface and mesh) are accumulated into single matrix equation, which is highly non-linear and involves all cross-derivatives. Besides the computational cost of solving the system of equations, monolithic Newton-Raphson strategy is difficult to implement. Consistent partitioned Newton-Raphson procedures tend to offer maximum robustness, but the implementation is even more tedious.

In Gauß-Seidel and staggered schemes the sub-domains are solved in turns - in separate systems of equations. The vector of traction forces $\mathbf{t}$ must be introduced to define the interaction between the sub-domains. As a result, the system of Equations (4.1) changes to

$$\left.\begin{aligned} \mathbf{r}_f(\mathbf{u}_f, \mathbf{u}_i) &= \mathbf{0} \\ \mathbf{g}_f(\mathbf{u}_f, \mathbf{u}_i) &= \mathbf{t} \end{aligned}\right\}\text{(fluid)}$$

$$\left.\begin{aligned} \mathbf{r}_s(\mathbf{d}_s, \mathbf{u}_i) &= \mathbf{0} \\ \mathbf{g}_s(\mathbf{d}_s, \mathbf{u}_i) &= -\mathbf{t} \end{aligned}\right\}\text{(solid)}, \tag{4.2}$$

which are solved at each time-step for $\mathbf{u}_i$, $\mathbf{t}$, $\mathbf{u}_f$ and $\mathbf{d}_s$.

Gauß-Seidel iteration scheme is named in literature also as Dirichlet-Neumann iteration or as fixed point iteration scheme. The algorithm starts from the fluid domain, which is solved for the fluid degrees of freedom as well as for the interface traction forces. The latter are then applied on the solid as an external load. Solid sub-solver calculates a new approximation of the kinematical interface degrees of freedom, which should be closer to the solution. The results from both domains are compared and if difference is below user defined tolerance, the algorithm moves to next time-step. Otherwise, the procedure is repeated.

In general the convergence of a Gauß-Seidel algorithm can be improved significantly if fixed or dynamic (e.g Aitken's or steepest descent) relaxation is used. For more information on Dirichlet-Neumann iteration scheme, one should refer to literature, [4] and [14].

The solution strategy and specific properties of staggered schemes will be discussed in the next Section.

4.2 Staggered Scheme for Fluid-Structure Interaction

Staggered schemes form a group of weakly-coupled solvers (see Figure 4.2). These solvers perform a fixed sequence of sub-solvers, traction force interpolations, data exchanges or any other constant number of calculations in each time-step, no matter whether the Equations (4.2) are satisfied or not. This causes a violation of the traction forces of the sub-domains, but if the time-step size and other algorithm parameters are carefully chosen, staggered schemes can be accurate and computationally very efficient. The solution of weakly coupled scheme coincides with those from Newton-Raphson and Gauß-Seidel schemes only as the time-step size tends to zero.

The solution procedure of staggered scheme, used in this project, is exclusively based on the article [5] by Dettmer and Perić.

Time variable t is introduced to describe the solution procedure of this scheme. The subscripts $0, 1, 2, ..., n, n+1, ...N$ denote the discrete instants of the simulation. Assuming that the solution for time instants t_{n-1} and t_n, is known, the approximation of the solution for time-step t_{n+1} is given in a Box, 4.1.

It can be seen that in step 2, the solid is loaded by the traction force predictor $\mathbf{t}^{\mathrm{P}}_{n+1}$, but the fluid responds in step 3 by a traction force $\mathbf{t}^{*}_{n+1}$. The difference $\mathbf{t}^{\mathrm{P}}_{n+1} - \mathbf{t}^{*}_{n+1}$ is the violation of the traction forces of the system of Equations (4.2). In order to spread the violation between the solid and the fluid, an average of the traction forces is calculated in step 4.

The coefficient β controls the contribution from the fluid domain, which at the same time can be viewed as the relaxation factor, parallel to Gauß-Seidel scheme. A natural choice of β is 0.5, but for large fluid-solid density ratios (large added mass effect), to ensure stability, smaller values of the coefficient are used. For

1. calculate traction force predictor
$$\mathbf{t}^{\mathrm{P}}_{n+1} = 2\,\mathbf{t}_n - \mathbf{t}_{n-1} \tag{4.3}$$
2. load solid with predicted traction force and solve for interface displacement (and for the displacement of solid internal nodes)
$$\left.\begin{aligned} \mathbf{r}_s(\mathbf{d}_{s_{n+1}}, \mathbf{u}_{i_{n+1}}) &= \mathbf{0} \\ \mathbf{g}_s(\mathbf{d}_{s_{n+1}}, \mathbf{u}_{i_{n+1}}) &= -\mathbf{t}^{\mathrm{P}}_{n+1} \end{aligned}\right\}\text{solve for } \mathbf{d}_{s_{n+1}} \text{ and } \mathbf{u}_{i_{n+1}} \tag{4.4}$$
3. move fluid non-slip boundary and update fluid mesh, then solve fluid for traction force
$$\left.\begin{aligned} \mathbf{r}_f(\mathbf{u}_{f_{n+1}}, \mathbf{u}_{i_{n+1}}) &= \mathbf{0} \\ \mathbf{g}_f(\mathbf{u}_{f_{n+1}}, \mathbf{u}_{i_{n+1}}) &= \mathbf{t}^{*}_{n+1} \end{aligned}\right\}\text{solve for } \mathbf{u}_{f_{n+1}} \text{ and } \mathbf{t}^{*}_{n+1} \tag{4.5}$$
4. compute the corrected traction force
$$\mathbf{t}_{n+1} = \beta\mathbf{t}^{*}_{n+1} + (1-\beta)\mathbf{t}^{P}_{n+1} \tag{4.6}$$
5. go to next time-step

Box 4.1 Procedures of staggered scheme in single time-step

more information, see Section 4.3 and an article [5].

Although in current project only constant size time-steps are used, in case of adaptive time stepping is used, the expression for predictor becomes

$$\mathbf{t}^{\mathrm{P}}_{n+1} = (1+\kappa_{n+1})\mathbf{t}_{i_n} - \kappa_{n+1}\mathbf{t}_{n-1}, \tag{4.7}$$

where $\kappa_{n+1} = \Delta t_{n+1}/\Delta t_n$ is the ratio of the current and the previous time-step.

One of the goals of this project is to maintain the second order accuracy for both sub-systems and for the whole FSI solver. As different time integration schemes are used for the fluid (in fact, some of the fluid time integrations schemes are not second order accurate) and for the solid, the time integration of the coupling needs extra care. The importance of time integration for coupled problems is investigated in an article [11] by Joosten and Dettmer, and the algorithm for particular staggered scheme, based on the generalised-α method, is presented in article [5].

Independently from the temporal discretisation (and its order) of the sub-systems, for the highest possible accuracy of the coupling, the traction of previous and current time-step must be interpolated. Basically, the interpolation, which is described in the following, is required if the fluid and solid traction forces are associated with different time instants within the time-step.

Assuming generalised-α method discretisation for both sub-domains the interpolation algorithm becomes

$$\mathbf{t}^{\mathrm{s}}_{n+\alpha^{\mathrm{s}}_f} = \alpha^{\mathrm{s}}_f\;\mathbf{t}^{\mathrm{s}^{\mathrm{P}}}_{n+1} + (1-\alpha^{\mathrm{s}}_f)\;\mathbf{t}^{\mathrm{s}}_n \tag{4.8}$$

$$\mathbf{t}^{\mathrm{f}}_{n+\alpha^{\mathrm{f}}_f} = \alpha^{\mathrm{f}}_f \; \mathbf{t}^{\mathrm{f}^*}_{n+1} + (1-\alpha^{\mathrm{f}}_f)\; \mathbf{t}^{\mathrm{f}}_n, \tag{4.9}$$

where

$\mathbf{t}^{\mathrm{s}}_{n+\alpha^{\mathrm{s}}_f}$	– traction forces on solid at time instant $t_{n+\alpha^{\mathrm{s}}_f}$
$\mathbf{t}^{\mathrm{f}}_{n+\alpha^{\mathrm{f}}_f}$	– traction forces by fluid at time instant $t_{n+\alpha^{\mathrm{f}}_f}$
$\mathbf{t}^{\mathrm{s}^{\mathrm{P}}}_{n+1}$	– predicted traction forces (load on solid at time instant t_{n+1})
$\mathbf{t}^{\mathrm{f}^*}_{n+1}$	– traction forces from the fluid domain at time instant t_{n+1}
$\mathbf{t}^{\mathrm{s}}_n = \mathbf{t}^{\mathrm{f}}_n$	– averaged traction forces of previous time-step, t_n
$\alpha^{\mathrm{s}}_f,\ \alpha^{\mathrm{f}}_f$	– time integration coefficients, which can be calculated as,

$$\alpha^{\mathrm{s}}_f = \frac{1}{\rho^{\mathrm{s}}_\infty + 1}, \tag{4.10}$$

$$\alpha^{\mathrm{f}}_f = \frac{1}{\rho^{\mathrm{f}}_\infty + 1}, \tag{4.11}$$

where ρ^{s}_∞, ρ^{f}_∞ are the spectral radius associated with an infinite time-step size of solid and fluid domain, respectively.

For clarity, a graphical interpretation of Equations (4.8) and (4.9) is presented in Figure 4.3.

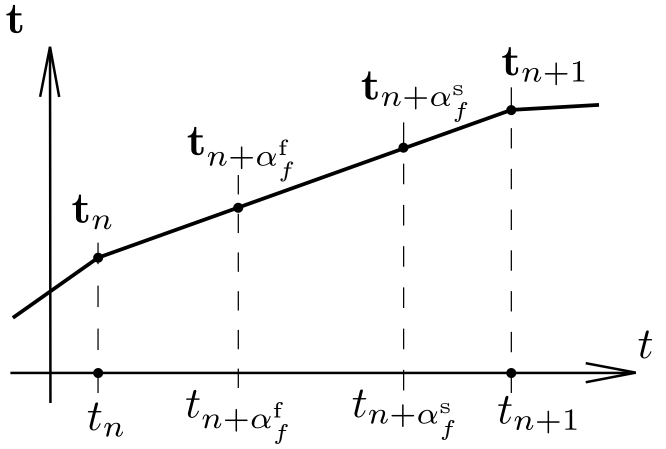

$\mathbf{t}$
$\mathbf{t}_{n+1}$
$\mathbf{t}_{n+\alpha^{\mathrm{s}}_f}$
$\mathbf{t}_{n+\alpha^{\mathrm{f}}_f}$
$\mathbf{t}_n$
t
t_n
$t_{n+\alpha^{\mathrm{f}}_f}$
$t_{n+\alpha^{\mathrm{s}}_f}$
t_{n+1}

Figure 4.3: Interpolation of traction forces

In order employ the traction forces interpolation on the scheme presented in Box 4.1, two additional calculations must be performed.

Firstly, between step1 and 2 in Box 4.1, Equation (4.8) must be used, to load the structure with forces associated to time instance $t_{n+\alpha^{\mathrm{s}}_f}$.

Secondly, between steps 3 and 4 in Box 4.1, Equation (4.9) must be employed to transfer (interpolate) the traction forces from time instant $t_{n+\alpha^{\mathrm{f}}_f}$ to t_{n+1}. More precisely, Equation (4.9) in from

$$\mathbf{t}^{\mathrm{f}^*}_{n+1} = \frac{\mathbf{t}^{\mathrm{f}}_{n+\alpha^{\mathrm{f}}_f} - \left(1 - \alpha^{\mathrm{f}}_f\right)\, \mathbf{t}^{\mathrm{f}}_n}{\alpha^{\mathrm{f}}_f} \tag{4.12}$$

is used.

In current project the generalised-α method is used for the solid domain throughout all the simulations.

For fluid, OpenFOAM (finite volume method) is used, and various schemes are available for time-integration of unsteady flows [18]. The most used of these are the first order implicit *backward Euler* (denoted as “Euler” in OpenFOAM) and the Crank-Nicolson (trapezoidal rule), with the possibility of blending it with the former (for time algorithms, see Section 3.3).

Comparing blended Crank-Nicolson method with the generalised midpoint rule, it can be seen that the blending factor (in OpenFOAM denoted as ψ) corresponds to ρ^{h}_∞ in the general midpoint rule. In both schemes, the smaller the named parameter (ψ or ρ^{h}_∞), the more numerical damping is introduced by increasing contribution from backward Euler scheme.

Thus, the parameter ρ^{f}_∞ in Equation (4.11) can be replaced by the blending factor, ψ, from the OpenFOAM (blended Crank-Nicolson).

There is not any efficient method to determine the optimal value for ψ (to ensure good accuracy and at the same time, maintain the boundedness) for unstructured grids. In this project, the optimal value was found by running many cases with various values of ψ (see Section 4.3).

4.3 Stability of Staggered Scheme

One of the main drawbacks of weakly coupled schemes has been their instability. The reason for it is the added mass effect with incompressible fluid and moderate solid over fluid mass ratio. The inertia of the fluid mass entrained by the solid structure is not considered in the structural dynamics solver. Thus, the deformation of the structure, when loaded with the fluid traction forces, is generally too inaccurate to allow for the design of a stable staggered algorithm.

Using the algorithm proposed in paper [5], a staggered scheme, with incompressible fluid and second order accurate implicit sub-solvers, is proven to be unconditionally stable even at relatively low ratios of solid over fluid. This is ensured by a simple control parameter β (see Section 4.2), similar to relaxation factor in block Gauß-Seidel schemes. Basically, β defines the contribution of traction forces from the sub-domains and averages them. To maintain the stability of the overall scheme, smaller values of β must be used with smaller ratios of solid over fluid. On the other hand, smaller values of β increase the inaccuracy of the coupling as the contribution of fluid traction forces is not accounted. Although, the second order accuracy of the coupling is maintained independently of the value of β.

In the article [5] by Dettmer and Perić, it is proven on a linear model problem, that high frequency damping of the overall system is reduced if the condition

$$\sqrt{1-\beta} \; > \; \max(\rho_\infty^\mathrm{f}, \rho_\infty^\mathrm{s}). \tag{4.13}$$

is fulfilled. Thus, instabilities are likely to occur if the Equation 4.13 is not true.

Although, this condition can not be applied directly to a complex numerical mode, it gives an indication for which combination of values β, ρ_∞^s and ρ_∞^f the computation might fail.

In limit case, where solid over fluid ratio is approaching to zero, for overall stability, the averaging factor β tends also to zero. Thus, the traction forces by fluid are ignored and high frequency oscillations may occur (if the damping by sub-solvers do not compensate it). Thus, prescribed strategy cannot be used with zero density or infinitely thin solid structures. Same criteria applies also to Gauß-Seidel strategies.

5 Implementation of the Weakly Coupled Scheme

In this chapter the two software, for fluid and solid domain, are described in the first two sections. The main focus is on the structure of the program rather than on problems arising from running a particular simulation.

The last section focuses on the merging of the two software.

5.1 MPAP2

Multi-Physics Analysis Program - MPAP2 is a finite element method software, developed in Swansea University, Civil and Computational Engineering Centre by Dr. Wulf G. Dettmer. MPAP2 has the capabilities of modelling fluid flow and non-linear solid mechanics problems. Moreover, it can solve coupled problems with interacting domains. For instance, contact dynamics and fluid-structure interaction. Theoretically, the number of sub-domains is not limited.

The current version of MPAP2 does not have the capabilities for turbulence modelling.

The core structure of MPAP2 is written in C++, while most of the material and element models have been implemented in Fortran. Object-oriented programming language, C++, enables high modularity and flexibility of the code, which is also employed in MPAP2.

One of the base classes in MPAP2 is `DomainTree` that holds all the domain types. Each domain type may contain one or more domains, which can interact with each other or with domains of another type. Each finite element method based domain has child classes: `Geometry`, `Mesh` and many other containing procedures for solving finite element method problems.

Classes related to graphical user interface (GUI) , LSE solvers etc, compose rest of the modules in the software.

The program is controlled by macros, which are executed automatically according to input file or run manually by using GUI. The flexibility of MPAP2 also allows for the straight forward implemenation of user defined macros.

5.2 OpenFOAM

OpenFOAM stands for Open Source Field Operation and Manipulation, which is a C++ library to create various executables - applications. These are divided into *solvers* and *utilities*. Most of the utilities are either *pre-processing tools* or *post-processing interfaces/converters*.

OpenFOAM is used by many universities as it is open source, flexible and modular, which makes it easy to add new solvers and boundary conditions. On the other hand, more options and larger flexibility requires wider knowledge from the user, which extend the period of study significantly.

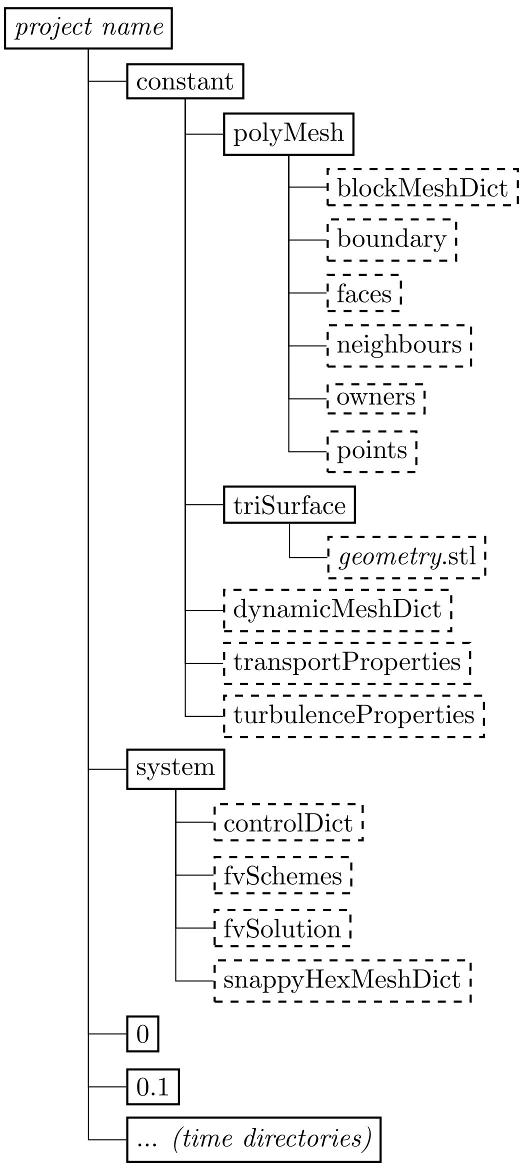
Finite volume method with collocated arrangement is the discretisation scheme for solving Navier-Stokes equations in OpenFOAM. Almost any other scheme or property of the simulation is user defined: interpolation, differentiation, integration schemes, preconditioners, LSE solvers and solution tolerances.

The program supports boundary deformations and has various built in mesh motion solvers. The boundary motion can either defined in terms of velocity or displacements. To ensure the quality of the grid, either laplace, stress-strain or interpolation based mesh solvers can be chosen (in various versions of OpenFOAM other mesh solvers are available). In the current project `displacementLaplacian` solver from `dynamicMotionSolverFvMesh` class with `quadratic inverseDistance` diffusivity was chosen as this strategy ensured the least distorted grid close to moving boundaries - the boundary layer mesh remained in good quality).

OpenFOAM version 2.1.1 contains large variety of standard solvers from potential flow solvers to combustion solver with chemical reactions and turbulence modelling and from Monte Carlo simulation to financial calculations, in total 78 standard solvers. The main categories of these solvers are: “Basic” CFD, incompressible flow, compressible flow, multiphase flow, direct numerical simulation, combustion, heat transfer and buoyancy-driven flows, particle tracking flows, molecular dynamics, direct Monte Carlo simulations, electromagnetics, stress analysis of solids and finally financial calculations.

Making all the standard solvers available in the assembled software would firstly be time consuming and more importantly - unnecessary. After the examination of all standard solver in OpenFOAM, it appeared that all required features exists in `pimpleDyMFoam` solver. `PimpleDyMFoam` is a transient solver for incompressible, flow of Newtonian fluids on a moving mesh using the PIMPLE (merged PISO and SIMPLE pressure coupling) algorithm [18].

A typical OpenFOAM `pimpleDyMFoam` solver project directory is shown in Figure 5.1



(continuous line rectangles denote directories and dashed denote files)

Figure 5.1: Structure of typical pimpleDyMFoam case directory

Geometry and constant physical properties are stored in the `constant` directory while files related to control parameters, numerical schemes and options for LSE solvers, are stored in the `system` directory. Boundary and initial conditions are defined time directory that is the starting time of the simulation (usually "0").

Most of data in OpenFOAM related files is saved in form, known as the dictionary. A dictionary is an entry, which contains data entries that can be retrieved by the input/output by means of *keywords.* A dictionary consists of dictionary name followed by curly brackets, in which one or more keywords with one or more entries are defined (please refer to [18] page U-102 for more information).

The results are saved in time directories that follow to initial conditions directory. Result directories can have the same sequence as the time instants in the solution process or once in every fixed number of time-steps. For example, one can save the velocity, pressure or any other physical quantity field after every 10 time-steps.

5.3 Assembly of Sub-solvers into Single Computer Program

The aim of this project was to construct a FSI solver, which is able to simulate (incompressible) fluid-structure interaction over a long period of time. Such solver can only be efficient if the sub-solvers share the same memory. Another option would have been to transfer data through files, which is time consuming, and increases even more in cases, where the ratio of interface DOF to sub-domains internal DOF, is larger.

The carrying idea of the assembly was to keep the OpenFOAM totally unchanged, which means that OpenFOAM libraries were attached to MPAP2 existing libraries. There are many reasons behind it:

- MPAP2 has already got the structure for interacting domains
- OpenFOAM is designed without any global variables
- compilation process of OpenFOAM is significantly more time consuming with that of MPAP2

In following paragraphs a detailed description of the assembly process of MPAP2 and OpenFOAM is given.

1. *Installation of OpenFOAM.*

   OpenFOAM source code can be download from *www.openfoam.org/download*. According to the Linux operating system (OS) distribution, relevant source pack with the compiling instructions, must be chosen. It is recommended to download and install the third-party software (e.g ParaView for post-processing) at once, in case it is not already included in the pack for particular Linux distribution.

   In case of using older version of Linux, it might be required to update the GNU Compiler Collection (GCC) for the installation of the latest release of OpenFOAM.

   On a regular personal computer (e.g CPU: Intel Core 2 Duo P8400/2.26 GHz, RAM: 4.0 GB) the compilation process takes about two hours.

2. *Installation of MPAP2*

   Having the source code of MPAP2, a `makefile`, needed for installation, is usually included in the pack. Besides *g++* compiler for the `*.cpp` files, a Fortran compiler is required due to material and element models, written in Fortran (see Section 5.1). For example, *gfortran*, which is part of the GCC package for Linux.

3. *Testing the functionalities of MPAP2 and OpenFOAM individually*

   After the installation of both software, it is recommended to test their functionality by running some of the tutorial cases which come with the software. Besides becoming familiar with the programs, it helps to discover any mistakes caused by incompatibility of computer architecture, compiler version and Linux distribution in the early stage.

4. *Removing ambiguous names*

   There are a few common function, class and variable names in MPAP2 and OpenFOAM. To avoid ambiguous names in the coupled software, some changes are needed in MPAP2. As the scope of MPAP2 classes is global with respect to OpenFOAM, thus any ambiguous class name can be avoided by adding the global scope symbol "::" at the front of each conflicting class in MPAP2.

   Careful positioning of calls to include header files in a `*.cpp` file may reduce the need for using the global scope notation. In general, files containing variables of wider scope - header files belonging to MPAP2 - should be placed before the header files of OpenFOAM.

   For some ambiguous function names the easiest solution, to knowledge of author, is changing the function name in MPAP2. In current project following changes were made: `time` ← `mpapTime`, `debug` ← `mpapdebug` and `solve` ← `mpapSolve`.

   The replacement in MPAP2 files is easily accomplished using Linux terminal command:

   ```
   $find .../fem/mpap2 -type f -exec sed -i 's/solve/mpapSolve/g'
   ```

5. *Compiling MPAP2 together with OpenFOAM libraries*

   After ensuring that both programs are working without any complications, the libraries originally belonging to OpenFOAM must be made available also for MPAP2. As this paper focuses only solving fluid domain using *pimpleDyMFoam* solver, the necessary libraries and paths for compilation can

be found in corresponding directory:
`/opt/openfoam211/applications/solvers/incompressible/pimpleFoam/pimpleDyMFoam/Make`. In case more solvers or utilities are necessary in the FSI solver, relevant libraries from OpenFOAM must be added to MPAP2 `makefile`.

To ensure that the solver is functioning, a macro that calls *pimpleDyMFoam* could be generated and any tutorial case of OpenFOAM run through MPAP2.

Besides added libraries, OpenFOAM requires several definitions of macros used by the preprocessor. These must be also added to the MPAP2 `makefile` as variables in command line option `-D`. More information about `wmake` (`makefile` used in OpenFOAM) can be found by studying files in directory: `/opt/openfoam211/wmake`.

6. *Creating a new domain type*, `OpenFOAM`, *in MPAP2*

   To use as much as possible the existing data structures for FSI in MPAP2, the most convenient method to program the interaction between pimpleDyMFoam and solid domain, is by defining OpenFOAM as another domain type. Changes that are needed in MPAP2 `Definitions.h` file are:

   - Adding `OpenFOAM` into `DOMAIN_TYPE_NAMES` list
   - Adding `OPENFOAM` into `DOMAIN_KEY` list
   - Adding `OPENFOAM` into `DOMAIN_TYPE_ENUM` list

   and changes in `prgReadFile.cpp` file are:

   - Adding to the end of header files list:
     ```
      #ifdef INCLUDE_OPENFOAM
        #include "OpenFOAM.h"
     #endif
     ```
   - Adding the OPENFOAM case into the list of domains (e.g after `case INTERFACEWCAM` block):
     ```
     case OPENFOAM:
     #ifdef INCLUDE_OPENFOAM
        domain.newDom(new OpenFOAM);
        n = domain[OPENFOAM].dom.n;
        cout << `` loading OPENFOAM '' << n << `` ...  \n\n'';
        domain(OPENFOAM,n-1).readFile(*Ifile);
     #else
        prgError(1,``prgReadFile'',``OpenFOAM not linked in!'');
     ```

```
  #endif
     break;
```

It is recommended to add

```
#ifdef INCLUDE_OPENFOAM
   ....
#endif
```

around any piece of code, related to OpenFOAM. By following this recommendation, in case OpenFOAM is not needed or there are compatibility errors, OpenFOAM functionalities can be easily revoked by not defining `INCLUDE_OPENFOAM` during the compilation.

Having MPAP2 ready to accept domain type "OpenFOAM", the header and `*.cpp` file with the declarations and constructors must be created.

It is unpractical to print and explain every line of class `OpenFOAM` in this paper, but the most important elements of the code will be presented.

In order to benefit from the shared memory and ensure fast data transfer and efficient memory use, pointers must be used to access data either from MPAP2 or OpenFOAM. This might not be immediate as the class "OpenFOAM" is created before any velocity or pressure field exist. Thus a `NULL` pointer must be created for each and every OpenFOAM data structure that is later needed.

Some of the most important data that is needed for the calculation of traction forces are: `runTime` (current time instant), `p` (pressure field) and `U` (velocity field), but also `nu` (value of kinematic viscosity). In current project the pointers for data from OpenFOAM is denoted with its original name + `MpapPtr`. For instance, `runTime` becomes `runTimeMpapPtr`.

Although `OpenFOAM` class constructor does not have any argument, the most important data read from the input file is the path to the particular OpenFOAM case directory, number defining the dimension, solver name (e.g `pimpleDyMFoam`) and name of patches that form the interface with solid.

In current version of the coupled software a possibility to define a individual motion of every patch has been programmed. The functionality can be employed in simulations, where the boundary motion is described by a time function. For an example, please see Section 6.4.

In OpenFOAM class the most important functions are:

- `OpenFOAM::createFieldsPimpleDyMFoam` - creates velocity and pressure field, by reading the boundary and initial conditions from files. Moreover, reads fluid properties ν, ρ, etc, from corresponding file.

- `OpenFOAM::runPimpleDyMFoam` - runs pimpleDyMFoam solver for single time instant. May contain many velocity-pressure coupling or non-orthogonality corrections loops. Furthermore, calls functions to deform interacting patch and to move internal mesh.
- `OpenFOAM::calcTraction` - based on the current velocity and pressure field, calculates the traction forces on the boundary. This is done by adding effective viscous stress and hydrostatic pressure.

7. *Creating a new class*, `InterfaceOFWC`

InterfaceOFWC (referring to keywords: Interface, OpenFOAM, Weakly Coupled) is child class of `Domain` and holds data structures and functions related to transferring traction forces and boundary motion. The most important member functions of `InterfaceOFWC` class are:

- `InterfaceOFWC::prepareInteractions` - function that is called once in the beginning of the simulation to calculate the traction and kinematic interpolation coefficients.

  The basis of this function is sticky interface, which assures that the mesh of sub-domains do not slide at the interface. Thus, the transfer coefficients are calculated once in the beginning of the simulation and remain constant.

  Calculation of transfer coefficients for traction forces in 2D:

  1. In case of non-matching meshes, a fluid cell is divided at the location of each solid node. The traction force of a cell is divided into smaller units (drawn as streaked in Figure 5.2).

  2. At centre of each traction force unit (denoted by triangle) two closes solid nodes are found and corresponding traction transfer coefficients are found (depending on the distance between unit centre and solid node).

$$p_1 = \frac{a}{a+b}; \qquad p_2 = \frac{b}{a+b}, \tag{5.1}$$

  where

  a, b – distance from force unit centre to solid node on left and right, respectively.

  p_1, p_2 – force coefficients for solid node on left and right, respectively,

  where $p_1 + p_2 = 1$.

  Calculation of transfer coefficients for velocity in 2D:

  1. In case of non-matching meshes, at each solid node two closest cell-vertexes are found and corresponding velocity transfer coefficients for both vertexes are found. Similarly to Equation (5.1).

A graphical interpretation of interface data transfer is shown in Figure 5.2.

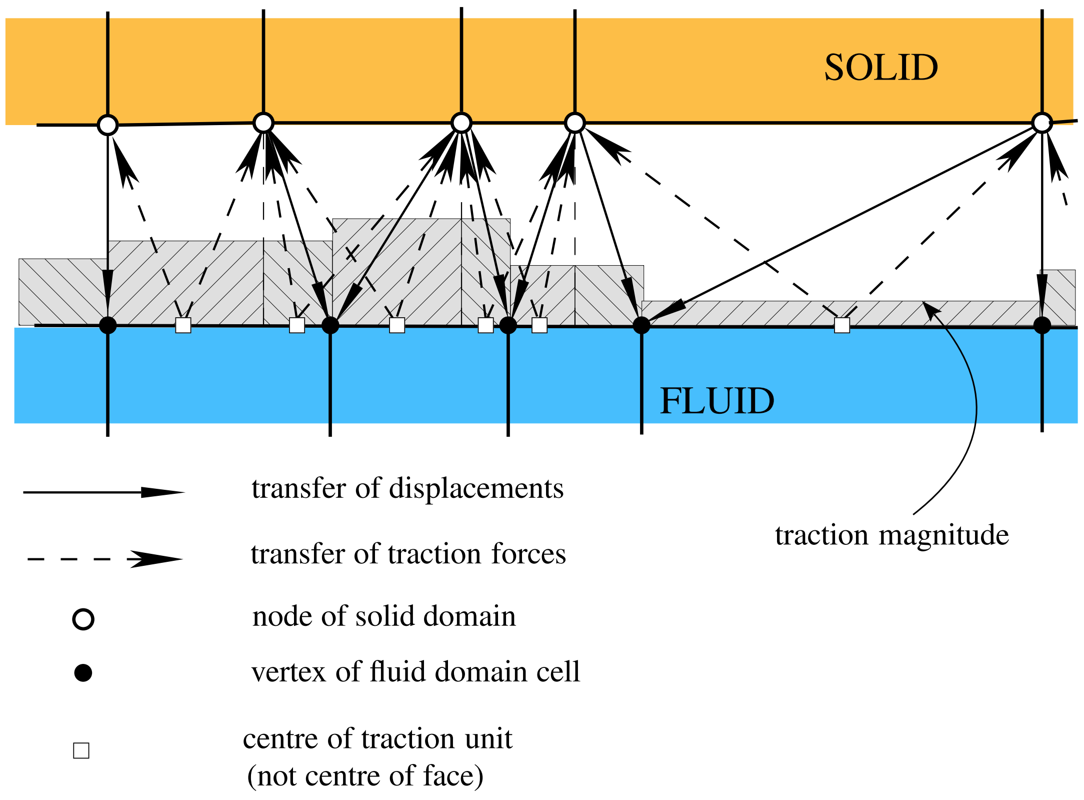


Figure 5.2: Interface data transfer

- `InterfaceOFWC::deformInterfacePatch` - employs the transfer coefficients, found from the previous function, and calculates new locations for the points at interface. Furthermore, calls mesh solver and finds new locations for internal points of fluid mesh according to boundary displacements and the solution scheme.

  Finally calculates the fluid velocity at face centres according to solid node velocities and the interpolation coefficients. The "sticky" boundary condition ensures that the velocity of fluid particle at the interface is the same as the velocity of solid boundary.

- `InterfaceOFWC::loadSolidInterface` - employs the coefficients from function `InterfaceOFWC::prepareInteractions` to find the load from traction force unit centre for particular node.

6 Numerical Examples

6.1 Fluid Dynamics - Channel Flow (study on grids)

In this section numerical results of flow in channel (2D Poiseuille flow) are presented. The main focus of this study is to investigate the affect of different finite volume grids on the results.

The domain with dimensions and boundary conditions is shown in Figure 6.1. Pressure is fixed at the outlet, a parabolic velocity profile with maximum value of 1.5, described at the inlet and non-slip boundary conditions are applied at the walls. Fluid density, $\rho = 1.0$ and the dynamic viscosity, $\mu = 0.00197$ ($Re = 507$). Internal velocity field was set to $v_x = 1.0$ as the initial condition.

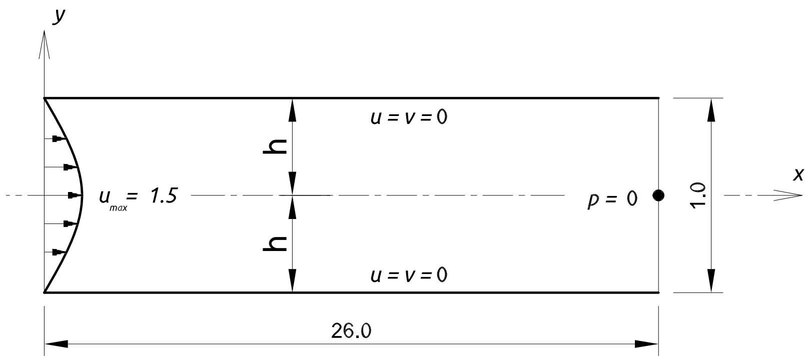


Figure 6.1: Geometry of channel flow problem

Being a simple domain, there exists a analytical solution for the velocity profile cross the channel as well as for the pressure distribution along the channel. The velocity profile is given,

$$u = -\frac{1}{2\mu}\frac{\mathrm{d}p}{\mathrm{d}x}(h^2 - y^2), \tag{6.1}$$

while the pressure distribution along the pipe gives:

$$p = -\frac{3\mu u_m}{h^2}x + p_0, \tag{6.2}$$

where u_m is the mean velocity at the inlet. For parabolic velocity profile with maximum value, $u_{max} = 1.5$, the mean value becomes $u_m = 1.0$.

Using Equation (6.2), one can find the pressure at the inlet

$$p = -\frac{3 \cdot 0.00197 \cdot 1.0}{0.5^2}(-26.0) + 0.0 = 0.61464\ . \tag{6.3}$$

In total six different grids were used, which properties are presented in Table 6.1. For the first five grids FEM oriented mesh generator was used and for the last OpenFOAM own mesher, `blockMesh` was used.

No.	Grid type	No. of	Non-orthogonality		Skewness	Pressure error
		cells	max.	avg.	max.	at inlet
1	unstructured triangular	22,356	20.16	4.78	0.497	23.1 %
2	unstructured quadrilateral	15,110	19.00	2.73	0.566	-1.91 %
3	structured triangular	6,656	36.87	21.28	0.333	29.1 %
4	structured quadrilateral	14,400	1.10	0.04	0.021	-0.25 %
5	structured quadrilateral	3,200	4.66	0.39	0.174	-0.40 %
6	struc. quad. (blockMesh)	3,000	0.00	0.00	0.000	-0.43 %

Table 6.1: Grid properties of channel flow model

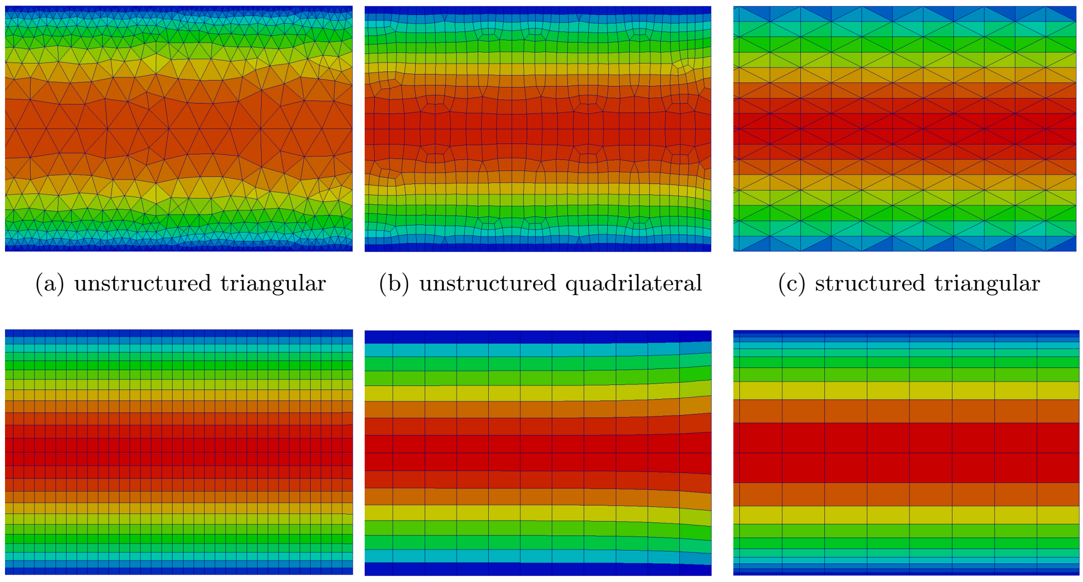
A grid samples of the outlet for each case is presented in the Figure 6.3.

All the grids, presented in the Table 6.1, pass the OpenFOAM mesh quality evaluation utility, `checkMesh` (tool to evaluate mesh quality for FV simulation).

PimpleFoam incompressible flow solver is used for the simulation. Time-step size is chosen such that $Co < 1.0$ throughout the simulation. For first five grids non-orthogonal correctors are used by setting `nNonOrthogonalCorrectors` $= 2$ (see Section 3.4.2). In all cases the number inner and outer loops are fixed to two (`nCorrectors` $= 2,$ `nOuterCorrectors` $= 2$). Turbulence model is set to `laminar`.

Pressure distribution is saved after the simulation has reached steady state phase (for various grids steady state is reach after different number of time-steps).

The pressure variation along the channel is presented in Figure 6.2.


(a) unstructured triangular (b) unstructured quadrilateral (c) structured triangular

(d) structural quad. (fine) (e) structured quad. (coarse) (f) struc. quad. (blockMesh)

Figure 6.3: Velocity field at outlet

In the Figure 6.2, it can be easily seen that triangular meshes without specific boundary layer mesh are performing very poorly compared to quadrilateral

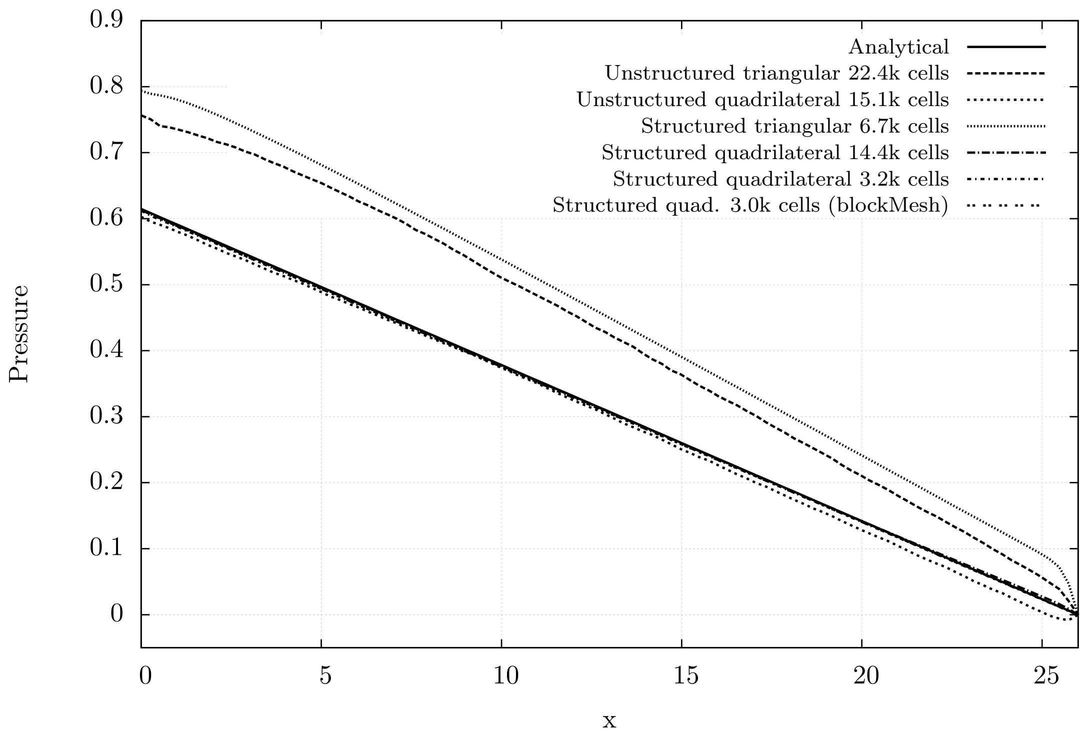


Figure 6.2: Pressure distribution of 2D channel flow for various grids

meshes. Grid 1 and 3 have also the largest non-orthogonality, but compared to grid 2, the ratio of the pressure error at the inlet and the non-orthogonality, is significantly larger for triangular meshes.

It can be argued that the number of non-orthogonal correctors or outer loops was not sufficient for particular grids, but even increasing the `nNonOrthogonalCorrectors` parameter did not improve the results significantly.

Secondly, it was observed that structured and more orthogonal grids provide better results compared to unstructured and largely non-orthogonal meshes.

Despite the fact that listed grids behaved in previously presented way just for particular domain and for trivial flow pattern, these outcomes can be used as guideline for domain discretisation for later problems.

6.2 Solid Dynamics - Cantilever Beam

In this section a simple structural dynamics problem is solved using finite element software MPAP2 and the results are compared to analytical solution. The main aim of this example problem is to investigate generalised-α method scheme for time integration in structural dynamics.

The model consist of a circular pipe, which is clamped at one end and loaded by body forces. The body forces are applied suddenly at time $t = 0$, causing the beam to oscillate. The material is described by following properties: density

$\rho = 7.85{\cdot}10^{-9}$; bulk modulus $K = 160,000$ and shear modulus $G = 80,000$, which corresponds to mild steel. Acceleration of $g = -9,800$ along y-axis is applied to create the loading.

The geometry with dimensions and boundary conditions is shown in Figure 6.4.

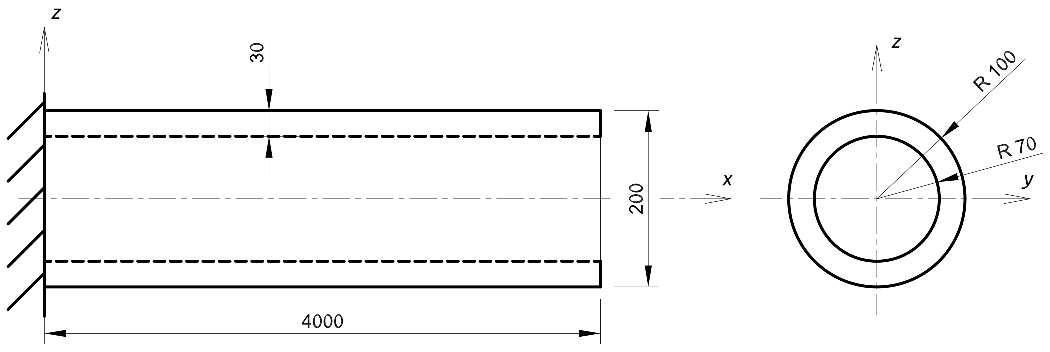


Figure 6.4: Geometry of clamped pipe

The finite element model, which can be seen in Figure 6.5, is constructed of 6,000 linear hexahedral elements using nearly incompressible Neo-Hook material model.

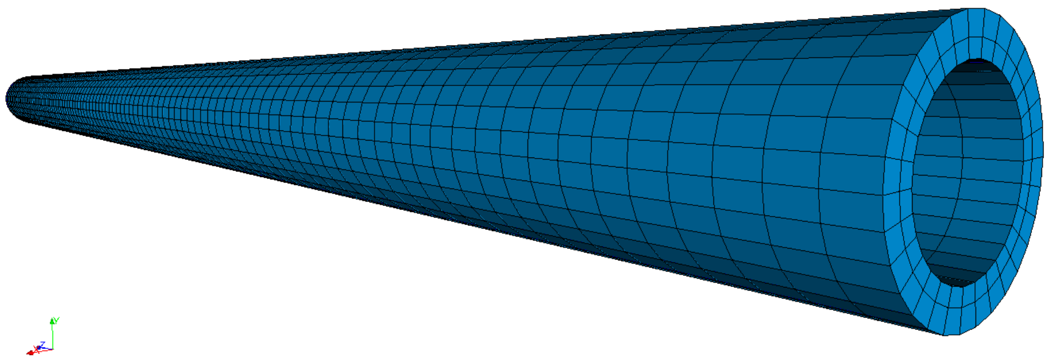


Figure 6.5: Finite element model of the pipe

As the three-dimensional (3D) domain is symmetric and loaded just in one direction, it can be viewed as 2D cantilever beam. Thus, an analytical solution can be found from the literature for the natural frequencies of this structure.

Only the main equations of the derivation of analytical solution are presented in this paper. One should refer to any textbook on vibrations for more information (e.g Dukkipati and Srinivas [7]).

Assuming a continuous cantilever beam with distributed mass and subjected to free vibration, the equation of motion can be written as (Meirovitch, 1967),

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}\left[EI(x)\frac{\mathrm{d}^2 w(x)}{\mathrm{d}x^2}\right] = -\lambda_m \omega^2 w(x), \tag{6.4}$$

where

E	– Young's modulus,
I	– moment of inertia,
w	- displacement along z-axis,
λ_m	- mass density of the beam (ρA),
ω	- angular frequency.

The boundary condition for this system are given as,

$$\text{At} \quad x = 0; \qquad w(x) = 0, \qquad \frac{\mathrm{d}w(x)}{\mathrm{d}x} = 0$$

$$\text{and at} \quad x = L; \qquad \frac{\mathrm{d}^2w(x)}{\mathrm{d}x^2} = 0, \qquad \frac{\mathrm{d}^3w(x)}{\mathrm{d}x^3} = 0\ .$$

Solving Equation (6.4) (employing separation of variables) gives

$$\frac{\mathrm{d}^4w(x)}{\mathrm{d}x^4} - \beta_n^4 w(x) = 0, \tag{6.5}$$

where

$$\beta_n^4 = \frac{\omega^2\lambda_m}{EI}\ . \tag{6.6}$$

After integrating and using the boundary conditions, the frequency equation is given as,

$$\cos(\beta_n L)\cosh(\beta_n L) = -1, \tag{6.7}$$

Corresponding to the eigenvalues of β_n, the mode shapes for the beam are given as,

$$w_n(x) = A_n[(\sin\beta_n L - \sinh\beta_n L)(\sin\beta_n x - \sinh\beta_n x) \tag{6.8}$$
$$+(\cos\beta_n L - \cosh\beta_n L)(\cos\beta_n x - \cosh\beta_n x)], \tag{6.9}$$

where $n = 1, 2, 3...\infty$.

Denoting $\alpha_n = \beta_n L$, the circular natural frequency can be expressed as,

$$\omega = \alpha_n\sqrt{\frac{EI}{\lambda_m L^4}}\ . \tag{6.10}$$

The Equation (6.7) is satisfied by specific values of α_n, which correspond to natural frequencies. For the first mode $\alpha_n = 1.875$ and natural frequency becomes

$$\omega_1 = 1.875^2\sqrt{\frac{EI}{\lambda_m L^4}} = 1.875^2\sqrt{\frac{205,714\cdot 59,682,000}{7.85\cdot 10^{-9}\cdot 16,022\cdot 4,000^4}} = 68.65\ . \tag{6.11}$$

This angular frequency corresponds to following frequency and period,

$$f_1 = \frac{\omega_1}{2\pi} = \frac{68.65}{2\pi} = 10.93$$

$$T_1 = \frac{1}{f_1} = \frac{1}{10.93} = 0.915\ .$$

As the objective of this example problem is to investigate generalised-α method scheme, the main focus stays on the effect of integration parameter ρ_∞^h on the results. Six different values of ρ_∞^h are used: 0.1, 0.3, 0.5, 0.7, 0.9 and 0.99. Time-step size $\Delta t = 0.01$ is used, which corresponds approximately to 90 time-steps per single oscillation period.

The results of the numerical simulation can be seen in Figures 6.6 and 6.7. Figure 6.6 shows the free end displacement over couple of cycles with the time integration parameter set to $\rho_\infty^h = 0.5$. The results for various values of ρ_∞^h are so close to each other that Figure 6.7 was generated to show the difference in the behaviour of the pipe in various simulations. The displacement curve for $\rho_\infty^h = 0.99$ is removed in Figure 6.7, as it is visually impossible to distinguish it from curve of $\rho_\infty^h = 0.9$

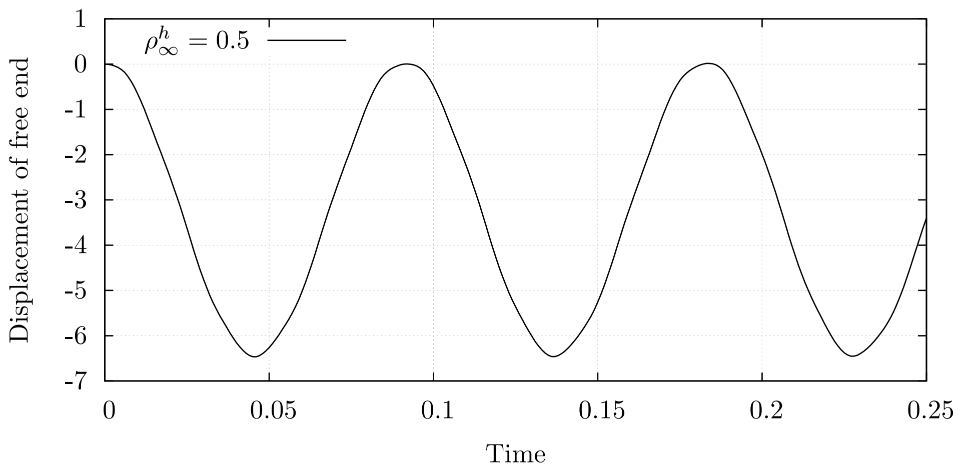


Figure 6.6: Oscillation of free end of pipe

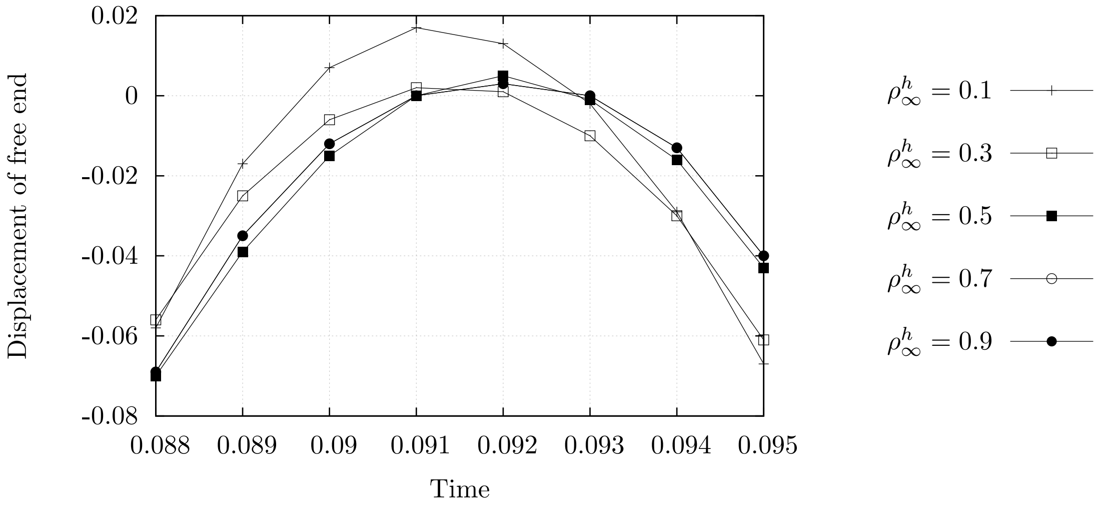


Figure 6.7: Pipe end displacement at the end of the first period

Moreover, Figure 6.7 provides the possibility to compare the analytical and numerical results visually. In terms of frequency, for all values of ρ^h_∞ the oscillation period coincides with the analytical results (difference less than 1%). Smaller time-step size and larger length to diameter ratio should be used to analyse the natural frequencies further, but in the scope of this project, the accuracy achieved, is absolutely sufficient.

As there is no damping nor active external loading involved in this simple cantilever beam problem, it is assumed that the amplitude of the oscillation remains constant. By analysing Figure 6.7, it can be seen that the integration parameter, $\rho^h_\infty = 0.5$, is performing best in terms of amplitude conservation.

Considering the outcomes of this simple beam problem, in further simulations integration parameter, $\rho^h_\infty = 0.5$, was used for the solids.

6.3 Fluid Dynamics - Flow around cylinder

The aim of this classical fluid flow problem is to investigating methods to access traction forces in OpenFOAM. The results, in terms of drag and lift coefficients, are compared with results from MPAP2 and the literature.

The geometry and boundary conditions of the model are shown in Figure 6.8. The fluid properties are: fluid density, $\rho = 1.0$ and the dynamic viscosity, $\mu = 0.005$ ($Re = 200$). The inlet velocity is set to constant over the whole section, $u_\infty = 1.0$ and same value for the internal velocity field is set as the initial condition.

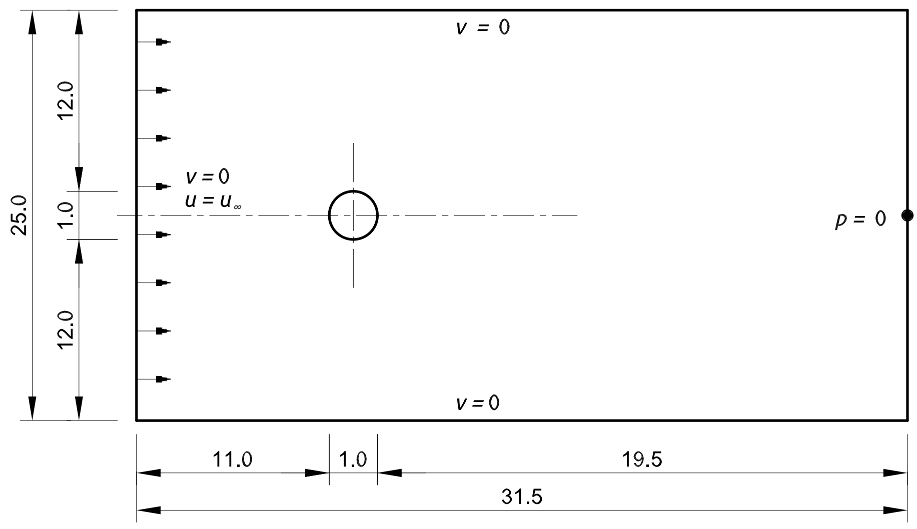


Figure 6.8: Geometry and boundary conditions of flow around cylinder problem

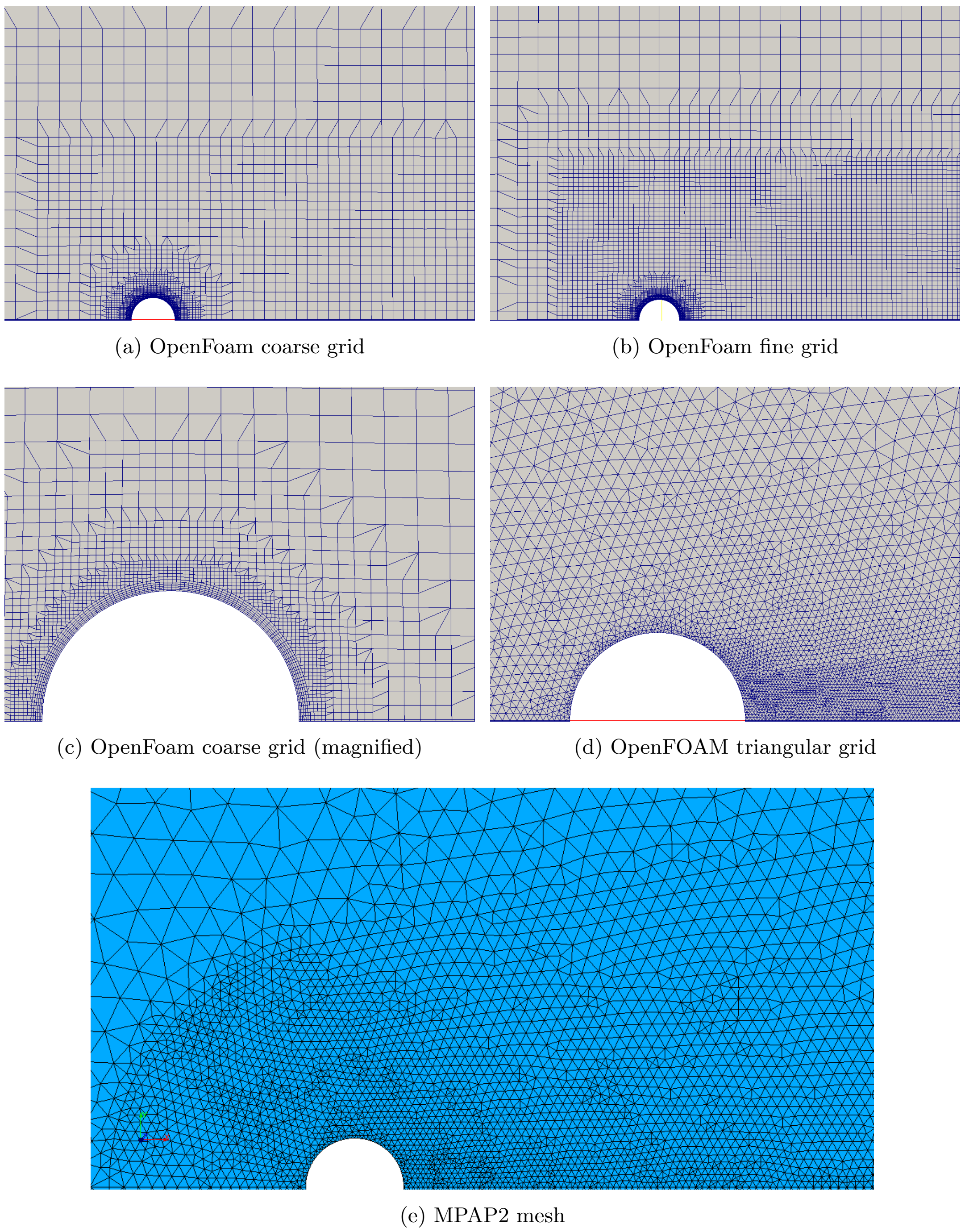
Three different meshes, coarse, fine and triangular (without structured boundary layer mesh), are generated for OpenFOAM and just single mesh for MPAP2. The properties of grids can be seen in Table 6.2.

Grid	Solver	No. of DOFs	Non-orthogonality max.	Non-orthogonality avg.	Skewness max.	Δt
OpenFoam Coarse	pimpleFoam	~ 25,728	25.97	5.42	0.369	0.005
OpenFoam Fine	pimpleFoam	~ 56,595	25.83	3.93	0.366	0.005
OpenFoam Triangular	pimpleFoam	~ 127,680	27.27	2.96	0.603	0.0025
MPAP2	MPAP2	~ 36,144	-	-	-	0.01

Table 6.2: Simulation properties for flow around cylinder problem

The coarse and fine grids for pimpleFoam solver are generated using `blockMesh` and `snappyHexMesh` utilities of OpenFOAM. The number of cells around cylinder for both grids is almost the same, while for the rest of the mesh, fine mesh is two times denser in both directions compared to coarse. The reason for this is to keep time-step size and thus maximum Courant number constant for both of the grids. Time-step sizes for each simulation are presented in last column of Table 6.2.

The mesh for MPAP2 and triangular grid for OpenFOAM is built in FEM oriented mesh generator and consists of 2D three-noded elements. For OpenFoam triangular grid no specific boundary layer mesh was generated close to the cylinder, but just very fine triangular grid. Fragments of meshes can be seen in Figure 6.9.


(a) OpenFoam coarse grid (b) OpenFoam fine grid

(c) OpenFoam coarse grid (magnified) (d) OpenFOAM triangular grid

(e) MPAP2 mesh

Figure 6.9: Grid fragments of flow around cylinder problem

Each simulation is ran until ($t > 100$) the amplitude of the drag and lift reaches constant value. For all OpenFOAM simulations the time-step size (shown in Table 6.2) is chosen small enough that Courant number stays below unity and the time integration scheme is set to Crank-Nicolson with blending factor $\psi = 0.5$. For FEM, generalised-α method is used with time integration parameter $\rho_\infty^{\mathrm{f}} = 0.5$.

In each case, the traction forces are saved and drag/lift coefficients calculated. Moreover, the period of the vortex shedding is found and Strouhal number calculated.

$$St = \frac{D}{Tu_\infty}$$

In OpenFOAM the functions for calculating the traction forces are compiled into individual library `libforces.so`, which must be included manually by defining it in `controlDict` file. Although, the same library cannot be used directly in the FSI solver, many functions from that library are employed by copying into `OpenFOAM` class in MPAP2.

Results of the simulations are presented in Table 6.3 and in Figures 6.10, 6.11.

Grid	max. C_D	max. C_L	St	compared to MPAP2 max. C_D	compared to MPAP2 max.C_L	compared to MPAP2 St
OpenFoam Coarse	1.39	0.63	0.1980	99.8%	94.2%	99.6%
OpenFoam Fine	1.35	0.60	0.1942	97.4%	90.3%	97.7%
OpenFoam Triangular	1.42	0.70	0.1984	102.1%	105.7%	99.8%
MPAP2	1.39	0.66	0.1988	100.0%	100.0%	100.0%

Table 6.3: Comparison of lift and drag coefficients

In Table 6.3, the MPAP2 results are chosen as reference value without any particular reason. The only aim is to provide easier comparison between the results.

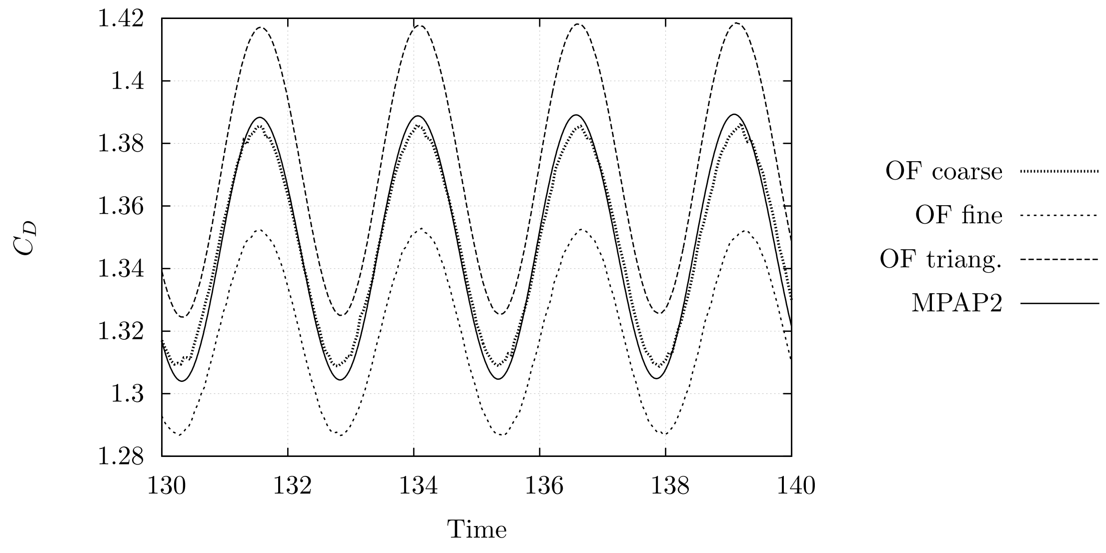


Figure 6.10: Comparison of drag coefficients

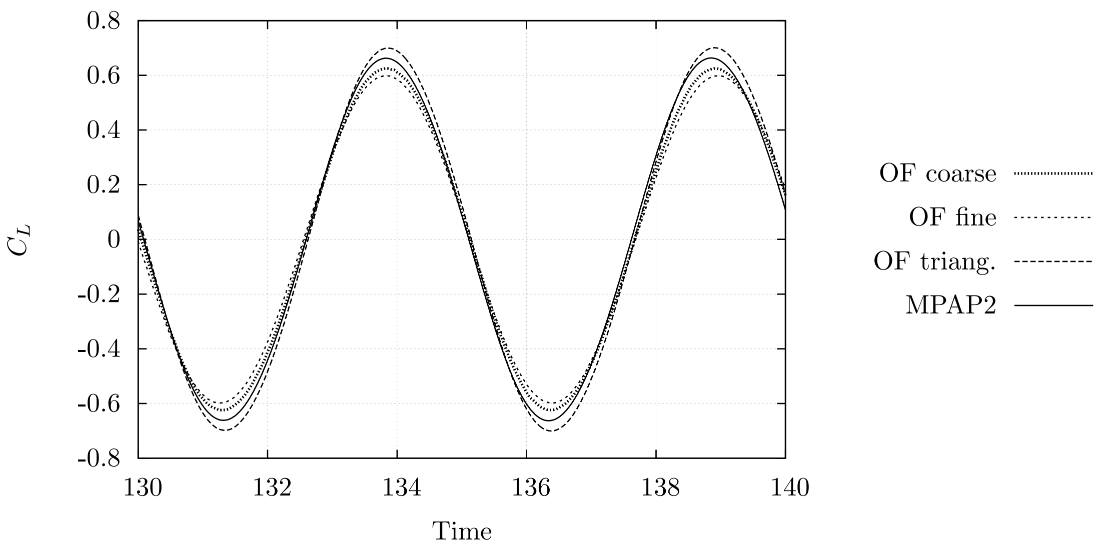


Figure 6.11: Comparison of lift coefficients

From the results it can be seen that the triangular mesh behaves “stiffer” compared to quadrilateral or to finite element results. This outcome coincides with results from the Section 6.1. Ignoring results of the triangular grid, mesh with large DOFs creates less drag and lift which is in correspondence with the theory.

Furthermore, FEM and FVM provide similar results at the same discretisation level - having similar number of degrees of freedom. Although, it is observed that FEM is less mesh dependent compared to FVM.

The pressure field of a fully developed wake region can be seen in Figure 6.12.

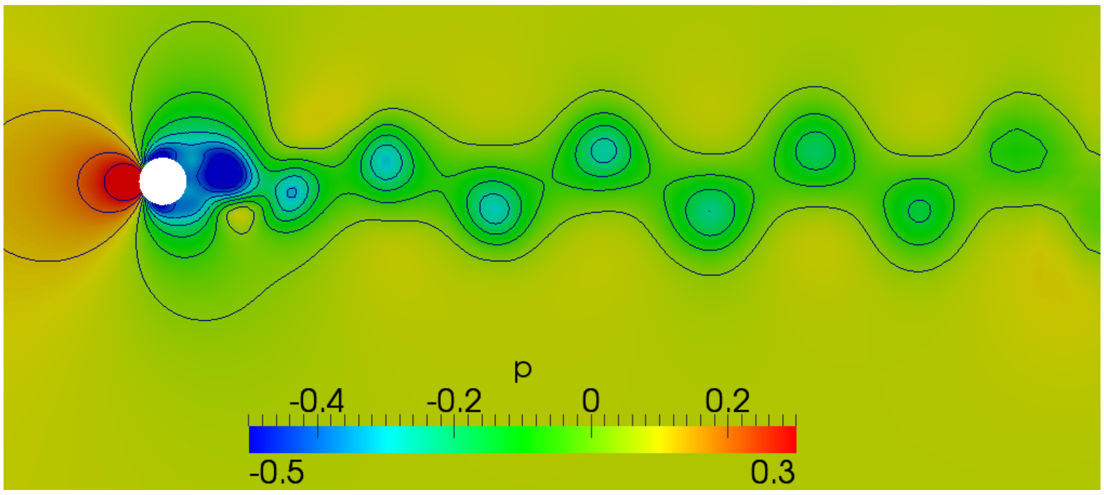


Figure 6.12: Pressure field of fully developed wake region (OpenFOAM fine grid)

Compared to some of the results in the literature (e.g Bergmann and Cordier [16]), the results vary approximately in the same scale as in present study. Most of the results of previous studies lie between the MPAP2 and OpenFOAM fine grid values, but even very close results to the triangular mesh can be found in some papers.

Thus, it can be said that the drag and lift coefficients computed using OpenFOAM are satisfactory. However, extra care is needed in order to guarantee high grid quality, especially in the region of boundary layer.

6.4 Fluid Dynamics - Flow in a Channel with Prescribed Wall Motion

The problem has been used previously presented by many researches (e.g Wall [22] and Dettmer [3]). The initial motivation of the problem was to have a simplified model of blood flow in a vessel. The aim of this example in current project is to study defining complex boundary conditions in OpenFOAM. This includes defining the motion of the boundary, specifying the velocity of fluid on that boundary and solving the internal mesh. The geometry and boundary conditions can be seen in Figure 6.13.

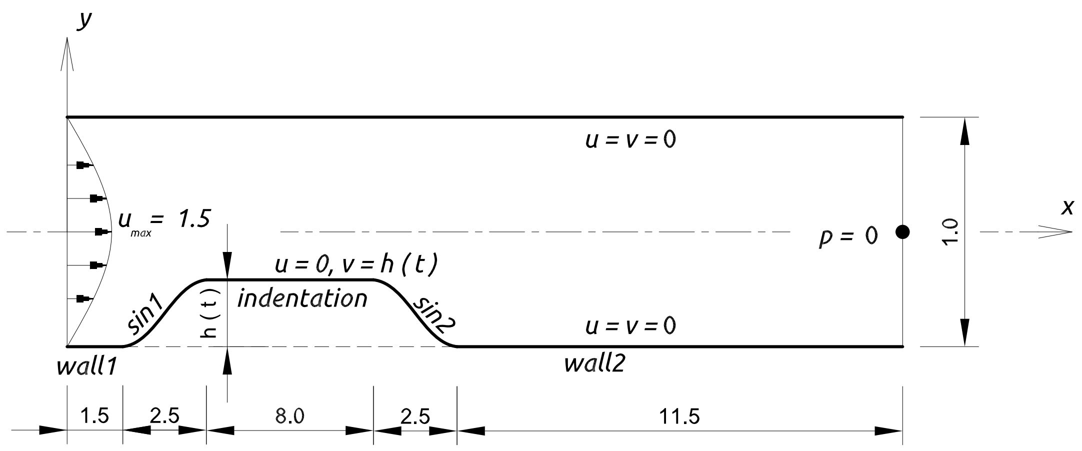


Figure 6.13: Geometry and boundary conditions of channel with wall indentation

Fluid flow parameters as well as the time function for moving the indentation are taken from Dettmer [3] and are given as, inflow velocity $\bar{u}_\infty = 1.0$, dynamic viscosity $\mu = 0.00197$, density $\rho = 1.0$ and the boundary motion

$$h(t) = \frac{1}{2}\epsilon\left[1 - \cos\left(2\pi\frac{t}{T}\right)\right], \tag{6.12}$$

where amplitude and period are, $\epsilon = 0.38$ and $T = 27.027$, respectively.

These parameters correspond to $Re = u_\infty b\rho/\mu = 507$. Initial channel width, $b = 1.0$, is taken as the characteristic length of the flow.

The mesh is generated using `blockMesh` and `mirrorMesh` utilities of OpenFoam. Grid is built of 6656 quadrilateral cells and a fragment of it is shown in Figure 6.14.

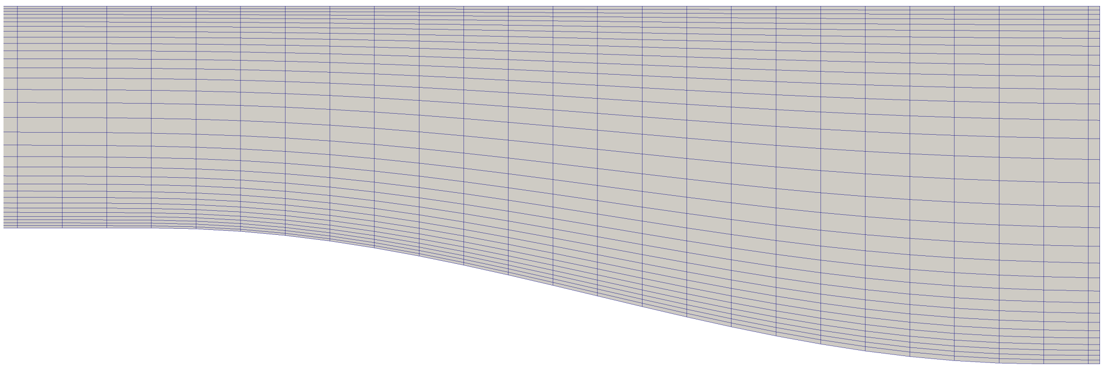


Figure 6.14: Fragment of the grid at $t = 13.5 \approx T/2$

Three new boundary condition were defined and compiled into individual libraries in OpenFOAM. They were used for describing the boundary movement of the wall fragment (denoted as "sin1", "indentation" and "sin2" in Figure 6.13). These new boundary conditions are similar to `oscillatingDisplacement` boundary condition, which is available in standard OpenFOAM (version 2.0). The main difference consists in finding the end points of the patches and calculating the displacement for every point on that patch according to the desired shape.

Two main criterion are examined in this simulation:

1. Wake region must form in downstream region of the indentation at the beginning of expansion phase and recover the uniform parabolic flow profile at the beginning of contraction phase (observations by Wall [22] and Dettmer [3]).
2. The fluid velocity at the boundary must coincide with velocity of the boundary in order to ensure correct wall conditions.

In Figure 6.15 it can be seen that the first criteria is fulfilled. Wakes are forming just after the indentation in the expansion phase and disappear by the beginning of next contraction phase.

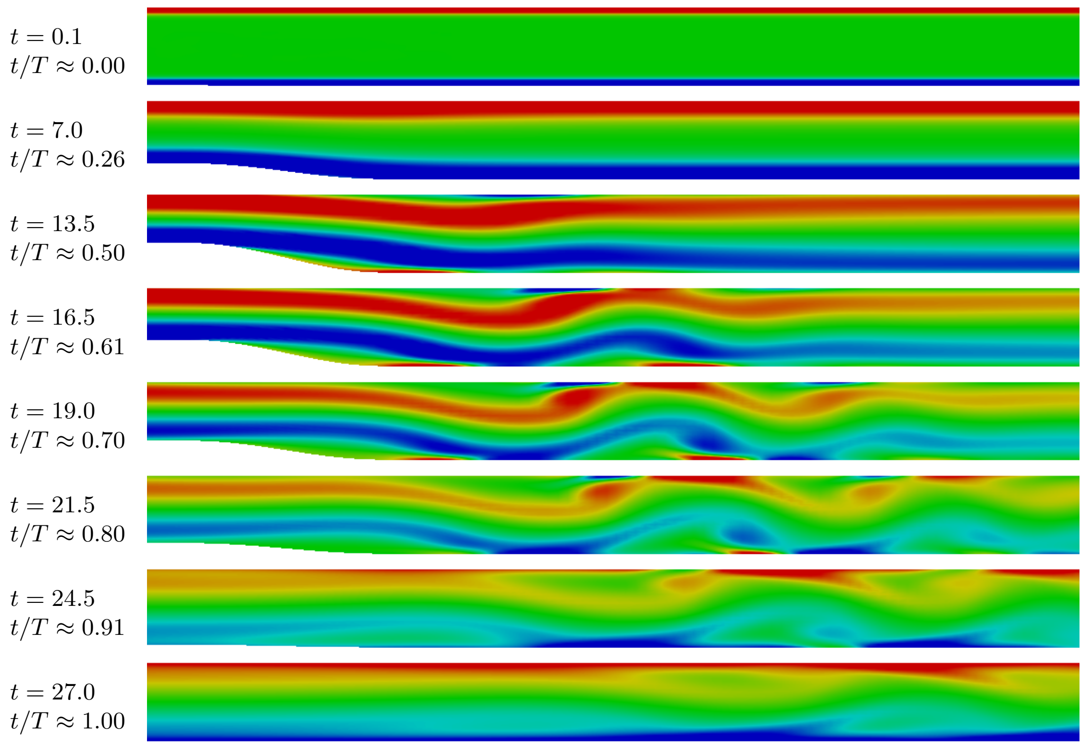


Figure 6.15: Vorticity (colour scale: $-6 \ldots +6$)

In order to verify that the second criteria is satisfied, one must find the velocity of the indentation. Thus the time derivative of displacement (Equation (6.12)) is needed,

$$v(t) = \frac{\partial h(t)}{\partial t} = \frac{\epsilon\pi}{T} \sin\left(2\pi\frac{t}{T}\right) . \tag{6.13}$$

The velocities are compared at time instant $t = 7.0$, when the wall motion is fastest

$$v(t = 7.0) = \frac{0.38\pi}{27.027} \sin\left(2\pi\frac{7.0}{27.027}\right) = 0.0441002 \; .$$

The velocity field at the same time instant, close to the moving boundary, was observed in ParaView. The vertical component of velocity field was the same as calculated using Equation (6.13). Thus, also the second criteria is satisfied.

6.5 FSI - Flow Induced Oscillations of a Flexible Beam

This model problem has been used by many researches (e.g A. Wall [22] and W.G Dettmer [3]) to test their FSI solution strategies. In current project the geometry as well as the material properties are the same as in previously mentioned papers.

The model consists of bluff body and a flexible beam behind it, in the region of wakes as shown in Figure 6.16.

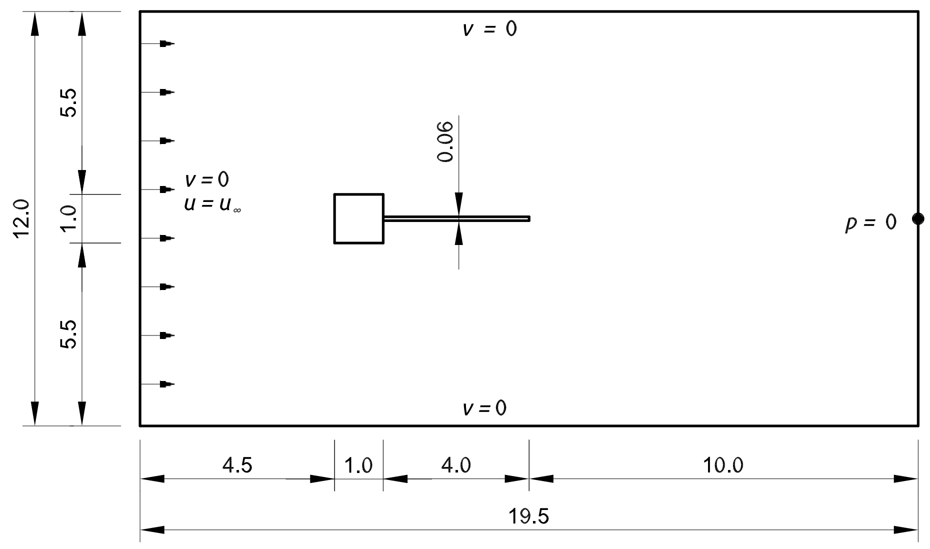


Figure 6.16: Geometry and boundary conditions of oscillating beam

The fluid domain is bounded by constant velocity inlet, pressure defined outlet and slip condition walls. No-slip boundary conditions are applied on the interfaces of square body and the beam. The fluid flow parameters are following:

inflow velocity $u_\infty = 51.3$,

dynamic viscosity $\mu_\mathrm{f} = 1.82 \cdot 10^{-4}$,

density $\rho_\mathrm{f} = 1.18 \cdot 10^{-3}$,

which correspond to Reynolds number

$$Re = \frac{\rho_\mathrm{f}\ D\ u_\infty}{\mu_\mathrm{f}} = \frac{1.18 \cdot 10^{-3} \cdot 1.0 \cdot 51.3}{1.82 \cdot 10^{-4}} = 332.6 \ .$$

Two fluid grids, a coarse and a fine, are generated, which parameters are presented in Table 6.4. The mesh quality parameters presented in this table are valid for initial configuration and will change during the simulation as the mesh deforms.

Grid	No. of	Non-orthogonality		Skewness
	cells	max.	avg.	max.
Coarse	20,962	49.67	4.30	2.883
Fine	70,453	46.42	3.16	0.839

Table 6.4: Fluid grid properties of oscillating beam problem

Both of the meshes are generated using OpenFOAM utilities `blockMesh`, `snappyHexMesh` and `extrudeMesh`. Although, fine grids is twice (in both directions) as dense as the coarse grid, the thickness of cells close to the interface with solid remains similar. An overview of the dense mesh and a closer snapshot of the coarse mesh are shown in Figures 6.17 and 6.18, respectively.

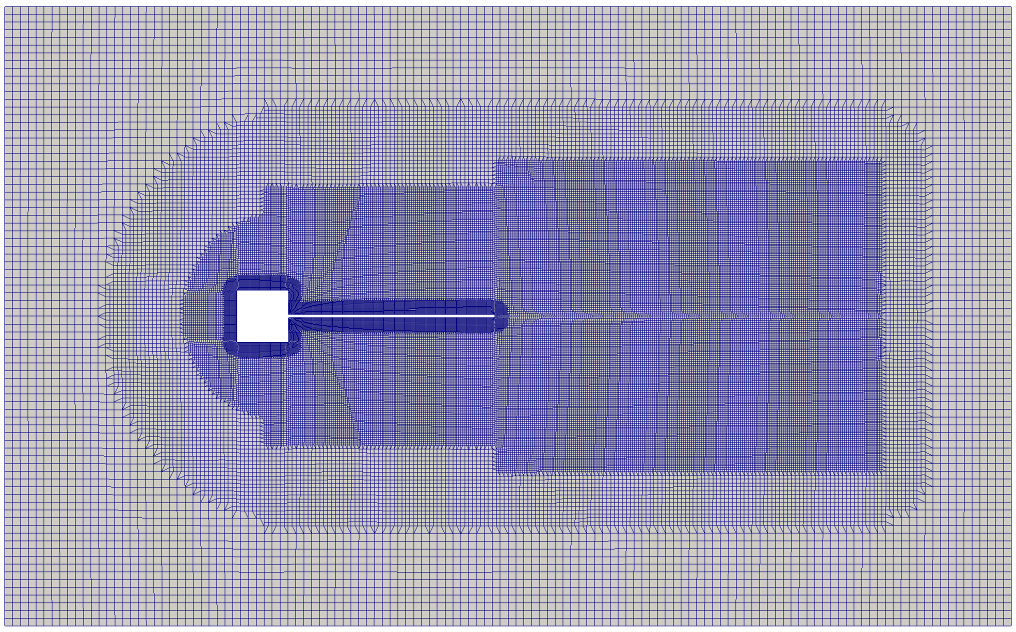


Figure 6.17: Overview of the fine mesh

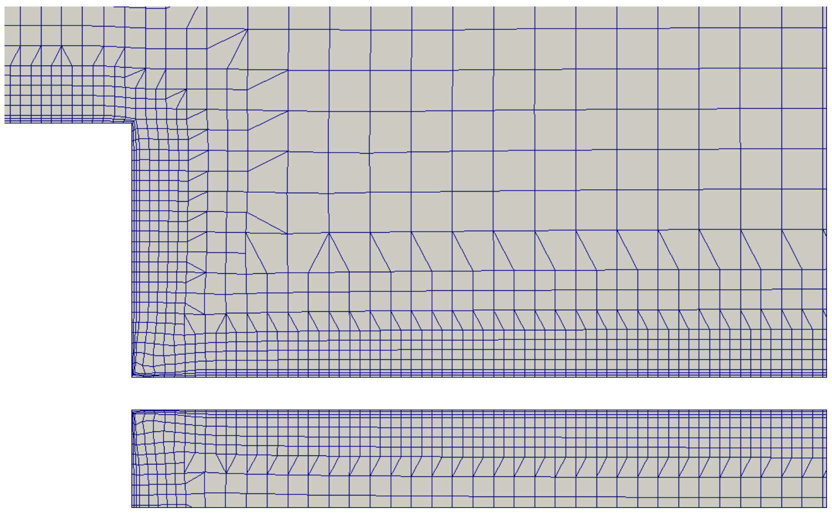


Figure 6.18: Coarse mesh at the connection of the bluff body and the beam

Employing PIMPLE algorithm in all the simulations, Courant number can be larger than one, but still one can not choose any time-step size. As the determinative Courant number of this problem arises from the motion of beam (time-step size with respect to thickness of the cell on interface), it is complicated to find correct prior running the simulation.

The bluff body is rigid and fixed in space whereas the beam material properties are

shear modulus $G_{\text{s}} = 9.2593 \cdot 10^5$,

bulk modulus $K_{\text{s}} = 2.78 \cdot 10^6$,

Young's modulus ($E_s = 2.5 \cdot 10^6$),

density $\rho_\mathrm{s} = 0.1$,
Poisson ratio $\nu_\mathrm{s} = 0.35$.

For the beam, plane stress conditions are assumed and only one mesh is generated, which consists of 20 nine-noded quadratic fully integrated rectangular 2D elements.

In most of the FSI problems the critical domain, in terms of the stability as well as computational cost, is the fluid. Thus, it is assumed, that the solid does not cause any instabilities and the main focus is on the fluid sub-domain. For the solid sub-domain, single time integration parameter is used, $\rho_\infty^\mathrm{s} = 0.5$.

To investigate the effect of traction force averaging parameter β (see Box 4.1 in Section 4.2 and the Section 4.3) and the time integration parameter for fluid ρ_∞^f (known as blending factor ψ in OpenFOAM), 16 simulations for both of the meshes are run.

In order to analyse just the stability of particular FSI scheme, the largest blending factor, which ensures the stability of fluid flow was determined beforehand. This was done by exploiting the same fluid model with fixed rigid beam at various blending factors. Although, many other parameters (e.g time-step and grid size) can affect the stability, it was found that $\psi > 0.8$ leads in most of the cases to unstable results.

The simulation parameters and results for coarse mesh and fine mesh are summarised in Tables 6.5 and 6.6, respectively.

β	$\rho^{\mathrm{f}}_{\infty}$	Stability		max. disp.		avg. frequency	
		Δt 0.0001	Δt 0.0005	Δt 0.0001	Δt 0.0005	Δt 0.0001	Δt 0.0005
0.05	0.05	Stable	Stable	1.196	1.206	3.247	3.247
0.05	0.3	Stable	Stable	1.148	1.173	3.195	3.247
0.05	0.5	Stable	Stable	1.171	1.172	3.279	3.247
0.05	0.8	Unstable	Unstable	-	-	-	-
0.3	0.05	Stable	Stable	1.176	1.182	3.236	3.247
0.3	0.3	Stable	Unstable	1.178	-	3.247	-
0.3	0.5	Stable	Unstable	1.144	-	3.215	-
0.3	0.8	Unstable	Unstable	-	-	-	-
0.5	0.05	Stable	Stable	1.178	1.162	3.268	3.247
0.5	0.3	Stable	Stable	1.152	1.246	3.279	3.247
0.5	0.5	Unstable	Unstable	-	-	-	-
0.5	0.8	Unstable	Unstable	-	-	-	-
0.8	0.05	Unstable	Stable	-	1.153	-	3.247
0.8	0.3	Unstable	Unstable	-	-	-	-
0.8	0.5	Unstable	Unstable	-	-	-	-
0.8	0.8	Unstable	Unstable	-	-	-	-

Table 6.5: Results of the coarse grid simulation

β	$\rho^{\mathrm{f}}_{\infty}$	Stability	max. displacement	avg. frequency
0.05	0.05	Stable	1.324	3.185
0.05	0.3	Stable	1.228	3.226
0.05	0.5	Stable	1.262	3.300
0.05	0.8	Unstable	-	-
0.3	0.05	Stable	1.316	3.205
0.3	0.3	Stable	1.255	3.226
0.3	0.5	Stable	1.238	3.165
0.3	0.8	Unstable	-	-
0.5	0.05	Stable	1.281	3.175
0.5	0.3	Stable	1.209	3.155
0.5	0.5	Unstable	-	-
0.5	0.8	Unstable	-	-
0.8	0.05	Stable	1.213	3.165
0.8	0.3	Unstable	-	-
0.8	0.5	Unstable	-	-
0.8	0.8	Unstable	-	-

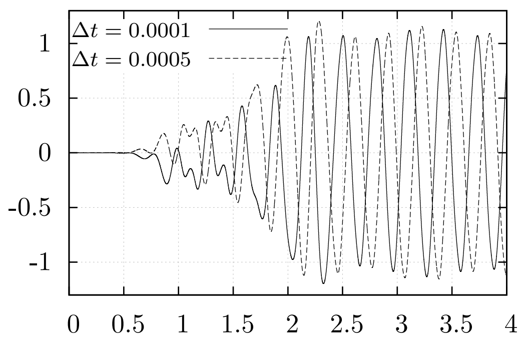
Table 6.6: Results of the fine grid simulation ($\Delta t = 0.00025$)

Stability. According to the theory in article [5], the implemented scheme becomes unstable for large values of β, which for the problem, under consideration, is approximately $\beta > 0.5$. At low values of β (high frequency damping of the coupling is reduced) the algorithm fails only if the fluid sub-solver possesses very little or no numerical damping (corresponds to large value of ρ_∞^{f}), which was observed also in current example.

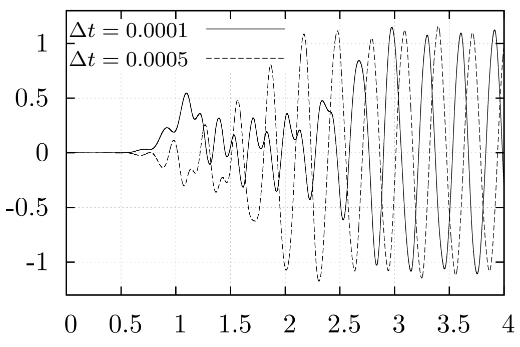
Amplitude and frequency. For all discretisation models considered, maximum the beam tip displacement lie between 1.14 and 1.32, the average frequency between 3.165 and 3.279.

Although, the maximum amplitudes of fine mesh are about 10% larger compared to coarse mesh, the average amplitude is approximately in the same range. See Figures 6.19 to 6.22 and 6.24 to 6.27 (axis notation is removed for better readability - vertical axis is for displacement and horisontal for time). The observation that finer spatial as well as temporal discretisation, firstly, increases the amplitude and, secondly, increases the number of cycles before the stable response of the beam is developed, was noticed in current project similarly to work [3] by Dettmer.

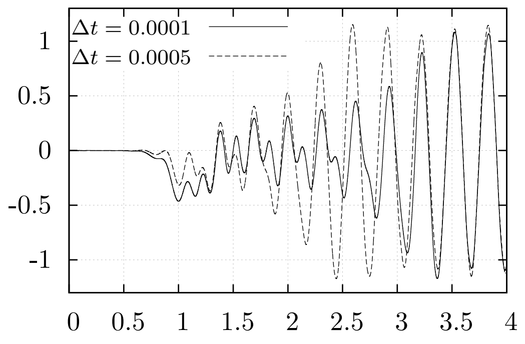
Comparing the values in Table 6.5 to results by Dettmer, the frequencies agree well, but the maximum amplitude in current work is approximately 8% smaller. Taking into account that the difference for the drag and lift coefficient in flow around cylinder example (see Section 6.3) was in the same range (5 to 8%), it may be concluded that the variation in amplitude is not due to resolution of interaction, but due to differences in fluid domain. For further analysis, more comparative simulations with fixed body should be performed.



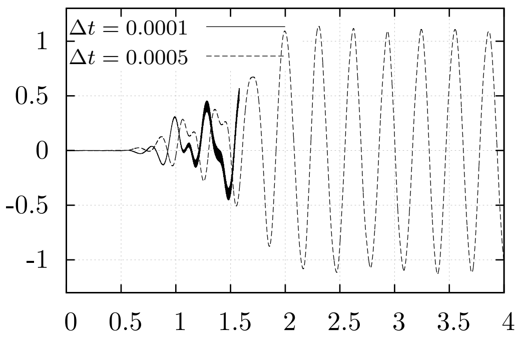
(a) $\beta = 0.05,\ \rho^{\mathrm{f}}_{\infty} = 0.05$



(b) $\beta = 0.05,\ \rho^{\mathrm{f}}_{\infty} = 0.3$

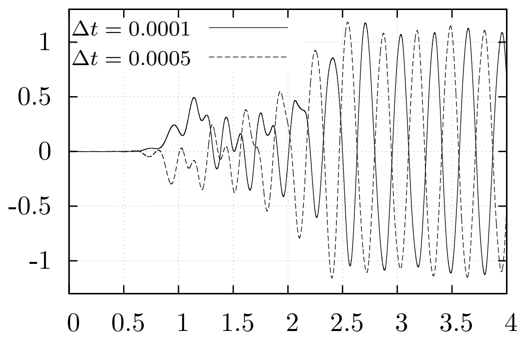


(c) $\beta = 0.05,\ \rho^{\mathrm{f}}_{\infty} = 0.5$

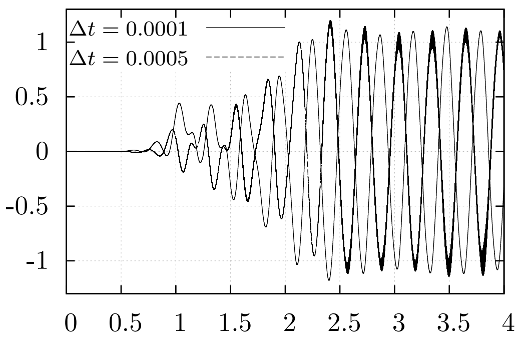


(d) $\beta = 0.05,\ \rho^{\mathrm{f}}_{\infty} = 0.8$

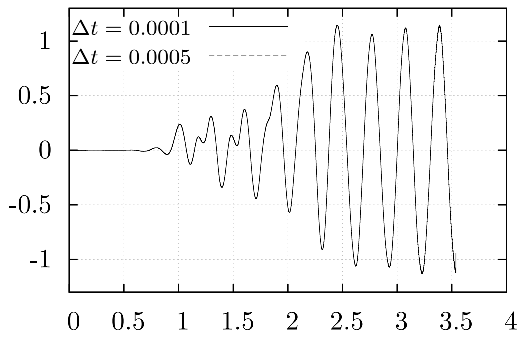
Figure 6.19: Oscillation of the beam tip (coarse mesh, $\beta = 0.05$)



(a) $\beta = 0.3,\ \rho^{\mathrm{f}}_{\infty} = 0.05$

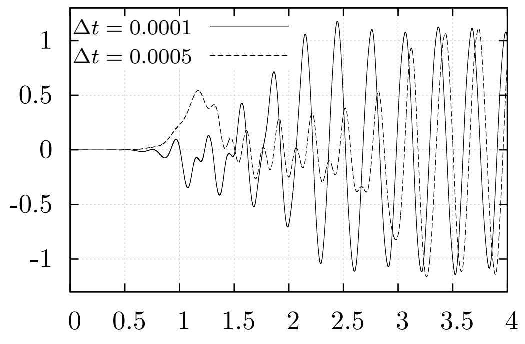


(b) $\beta = 0.3,\ \rho^{\mathrm{f}}_{\infty} = 0.3$

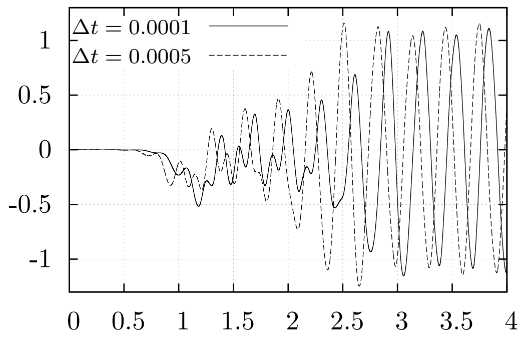


(c) $\beta = 0.3,\ \rho^{\mathrm{f}}_{\infty} = 0.5$

Figure 6.20: Oscillation of the beam tip (coarse mesh, $\beta = 0.3$)

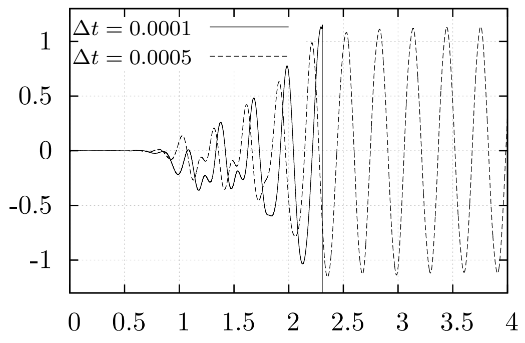


(a) $\beta = 0.5,\ \rho^{\mathrm{f}}_{\infty} = 0.05$



(b) $\beta = 0.5,\ \rho^{\mathrm{f}}_{\infty} = 0.3$

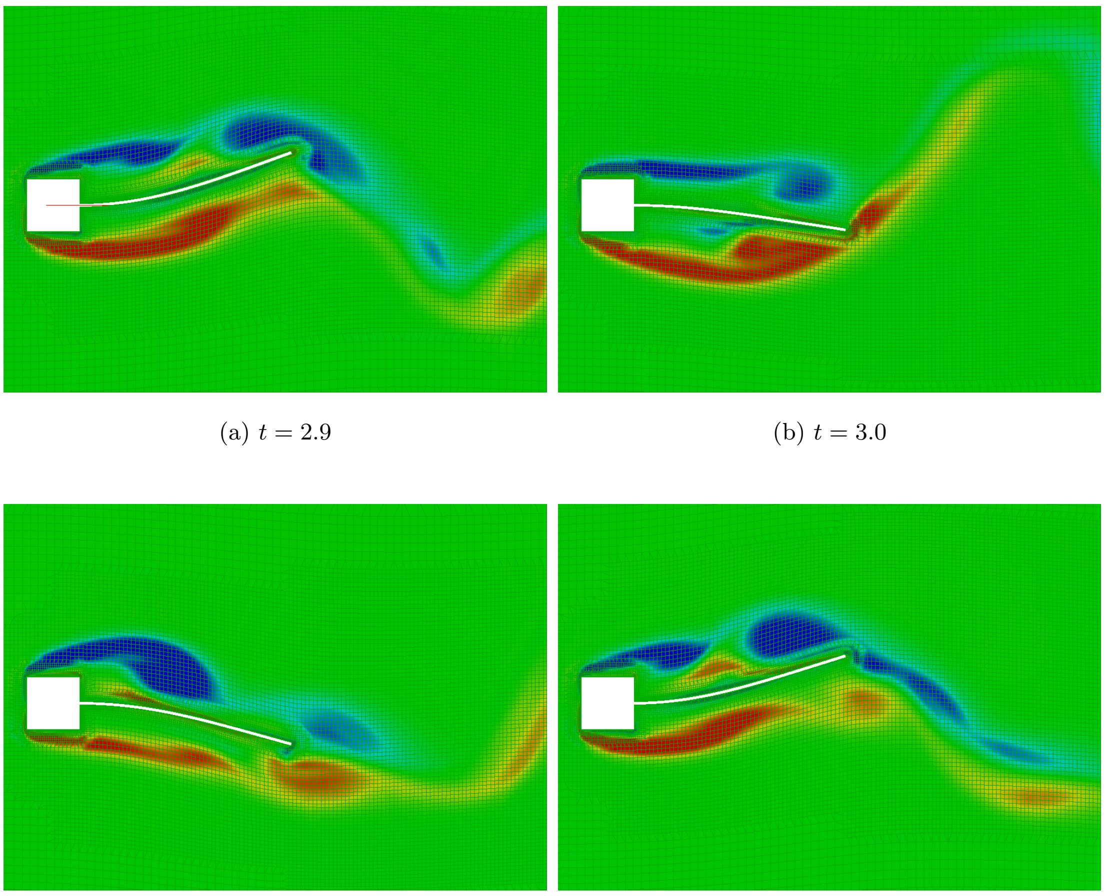
Figure 6.21: Oscillation of the beam tip (coarse mesh, $\beta = 0.5$)



(a) $\beta = 0.8,\ \rho^{\mathrm{f}}_{\infty} = 0.05$

Figure 6.22: Oscillation of the beam tip (coarse mesh, $\beta = 0.8$)

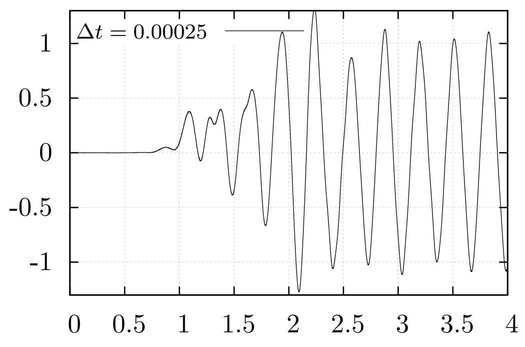
A typical oscillation cycle of coarse mesh with parameters of $\Delta t = 0.0001$, $\beta = 0.5$ and $\rho^{\mathrm{f}}_{\infty}$ is shown in Figure 6.23. The vorticity contour plots agree well with those presented in [3].



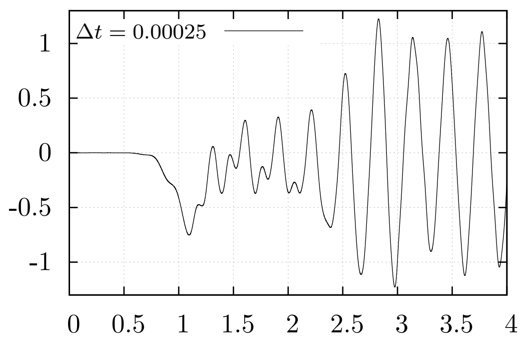
(a) $t = 2.9$ (b) $t = 3.0$

(c) $t = 3.1$ (d) $t = 3.2$

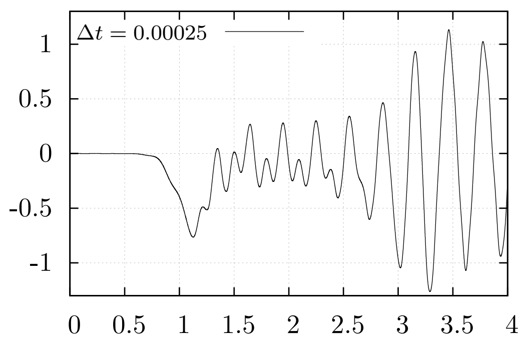
Figure 6.23: Flow induced oscillations (vorticity colour scale -150...+150)



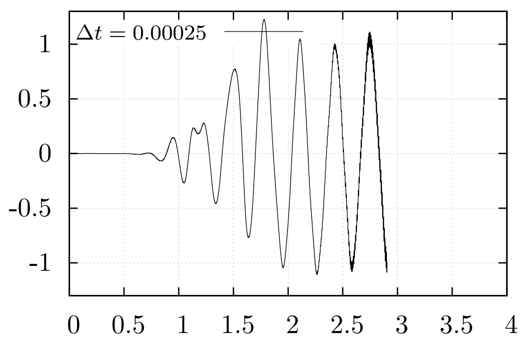
(a) $\beta = 0.05,\ \rho^{\mathrm{f}}_{\infty} = 0.05$



(b) $\beta = 0.05,\ \rho^{\mathrm{f}}_{\infty} = 0.3$

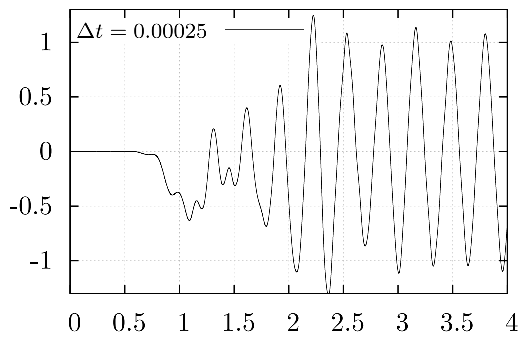


(c) $\beta = 0.05,\ \rho^{\mathrm{f}}_{\infty} = 0.5$

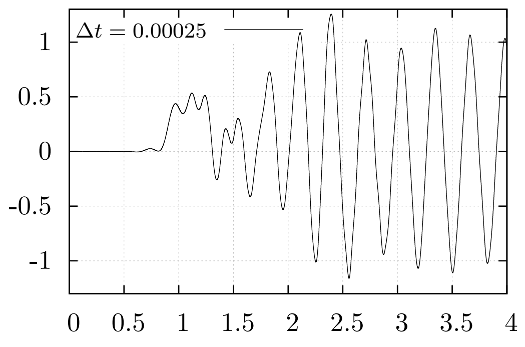


(d) $\beta = 0.05,\ \rho^{\mathrm{f}}_{\infty} = 0.8$

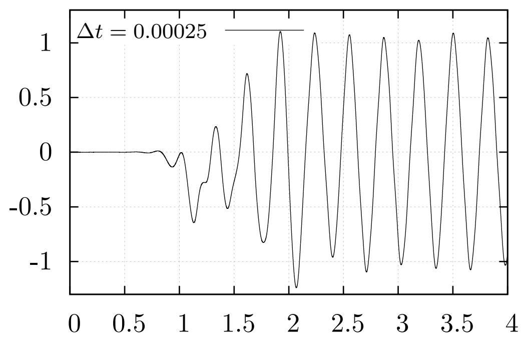
Figure 6.24: Oscillation of the beam tip (fine mesh, $\beta = 0.05$)



(a) $\beta = 0.3,\ \rho^{\mathrm{f}}_{\infty} = 0.05$

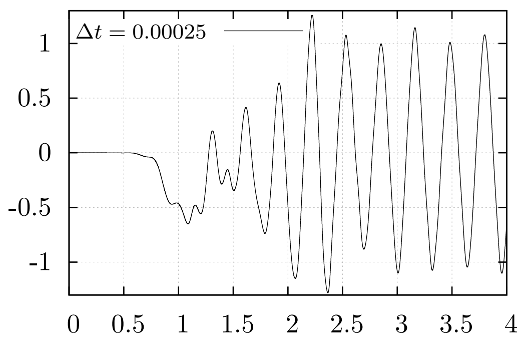


(b) $\beta = 0.3,\ \rho^{\mathrm{f}}_{\infty} = 0.3$

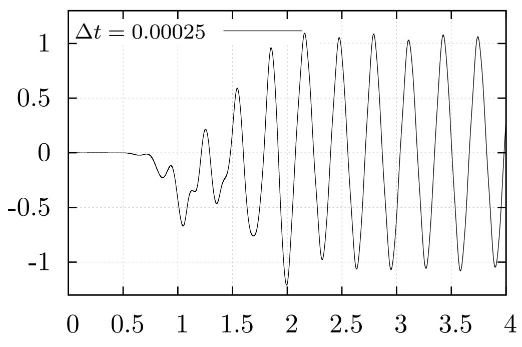


(c) $\beta = 0.3,\ \rho^{\mathrm{f}}_{\infty} = 0.5$

Figure 6.25: Oscillation of the beam tip (fine mesh, $\beta = 0.3$)

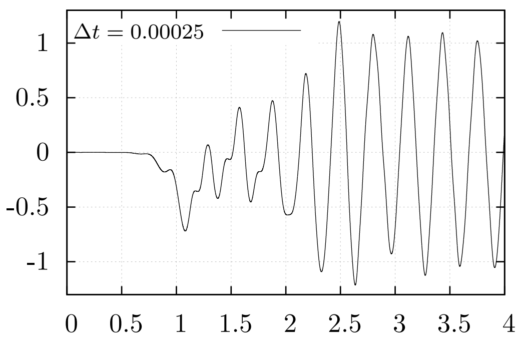


(a) $\beta = 0.5,\ \rho^{\mathrm{f}}_{\infty} = 0.05$



(b) $\beta = 0.5,\ \rho^{\mathrm{f}}_{\infty} = 0.3$

Figure 6.26: Oscillation of the beam tip (fine mesh, $\beta = 0.5$)



(a) $\beta = 0.8,\ \rho^{\mathrm{f}}_{\infty} = 0.05$

Figure 6.27: Oscillation of the beam tip (fine mesh, $\beta = 0.8$)

7 Conclusions

The main objective of this work, the linking of two existing sub-solver software into a combined fluid-structure interaction solver, has been achieved. A software framework to access data or call functions of OpenFOAM via MPAP2 was developed. The staggered scheme for combining implicit fluid and solid solvers, developed by Dettmer and Perić, was implemented and a tested on a benchmark problem. In the scope of this work only two-dimensional laminar flow problems were viewed.

A more detailed description of the achievements and suggestions for further work are presented in the following.

Firstly, the theory of basic finite element analysis was presented on the example of heat conduction and linear elasticity. The steps from governing equations to weak form and finally, to system of algebraic equations were shown. A brief description of the solution procedure of a typical finite element software was given.

In order to illustrate the theory, and study the main features of MPAP2, a simple structural dynamics problem was solved and compared to analytical solution (see Section 6.2). Results from numerical simulation were basically identical to the analytical solution.

Secondly, an introduction of finite volume analysis was given on the basis of open source software - OpenFOAM. The most employed discretsation schemes for each term in Navier-Stokes equations were presented. The steps of SIMPLE and PISO algorithms were given, as these are the main strategies for solving pressure-velocity coupling of incompressible fluid dynamics in OpenFOAM. The aspects of non-orthogonality in the finite volume grid were described and illustrated by a numerical example of two-dimensional Poiseuille flow (see Section 6.1).

Thirdly, the theory of computational fluid-structure interaction was given. A brief overview of various strategies was presented with the main focus on the weakly coupled scheme. The implementation of specific solution strategy was presented and short instructions for merging OpenFOAM and MPAP2 were given.

Finally a numerical example of FSI benchmark problem was provided. Two different mesh densities were considered and for each mesh 16 different combinations of fluid time integration parameters and traction forces averaging coefficients were run.

In this example, it was observed that the stability of the scheme can be ensured by careful choice of traction forces averaging parameter β. Small values of β lead to instabilities only if no or very little numerical damping is introduced by the fluid sub-solver, whereas large values of β cause the scheme to fail in all the cases, except if large time-step and backward Euler scheme is used for the fluid.

The oscillation frequencies agreed well with works by other researchers, but approximately 8% difference was observed in the beam tip amplitude. Similar disagreement was observed in drag and lift coefficient for cylinder in the flow (see Section 6.3).

Current project included one benchmark problem with various settings (meshes, integration parameters). In order to get better understanding of the differences with respect to work by Dettmer, more comparative fluid flow simulations should be performed. Moreover, deeper study of OpenFOAM would be beneficial to ensure that the meshing is good quality and all optimal solution parameters are chosen.

After a good agreement of lift and drag is achieved with well known CFD software or experimental data, studies should be extended to large Reynolds number and turbulent flows.

To widen the range of applicability, the coupling algorithm should be modified to solve 3D problems. Moreover, for large problems, a possibility of employing multiple processors should be programmed.

References

[1] J. Bonet and R.D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analyis*. Cambridge University Press, 2rd edition, 2008.

[2] R.D. Cook, D.S. Malkus, M.E. Plesha, and R.J. Witt. *Concept and Applications of Finite Element Analysis*. John Wiley & Sons Inc, 4th edition, 2001.

[3] W.G. Dettmer. *Finite Element Modelling of Fluid Flow with Moving Free Surfaces and Interfaces Including Fluid-Solid Interaction*. Phd thesis, University of Wales Swansea, September 2004.

[4] W.G. Dettmer. Fluid-structure interaction. Lecture notes, 2012. MSc in Computational Mechanics, Swansea University.

[5] W.G. Dettmer and D. Perić. A new staggered scheme for fluid-structure interaction. *International Journal for Numerical Methods in Engineering*, 93(1):1–22, 2013.

[6] P. Díez. The finite element method. Lecture notes, 2011. MSc in Computational Mechanics, UPC Barcelona.

[7] R.V. Dukkipati and J. Srinivas. *Textbook of Mechanical Vibrations*. PHI Learning Private Ltd, 2rd edition, February 2012.

[8] J.H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer-Verlag, 3rd edition, 2002.

[9] G.A. Holzapfel. *Nonlinear Solid Mechanics - A Continuum Approach for Engineering*. John Wiley and Sons Ltd, 2000.

[10] H. Jasak. *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. Phd thesis, Imperial College of Science, Technology and Medicine, June 1996.

[11] M.M. Joosten, W.G. Dettmer, and D. Perić. On the temporal stability and accuracy of coupled problems with reference to fluid-structure interaction. *International Journal for Numerical Methods in Fluids*, 64:1363–1378, 2010.

[12] C. Kassiotis, A. Ibrahimbegovic, and H.G. Matthies R. Niekamp. Nonlinear fluid-structure interaction problem. part i: implicit partitioned algorithm, nonlinear stability proof and validation examples. *Computational Mechanics*, 47(3):305–323, 2011.

[13] C. Kassiotis, A. Ibrahimbegovic, and H.G. Matthies R. Niekamp. Nonlinear fluid-structure interaction problem. part ii: space discretization, implementation aspects, nested parallelization and application examples. *Computational Mechanics*, 47(3):335–357, 2011.

[14] U. Küttler and W.A. Wall. Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43(1):61–72, 2008.

[15] J. Lorentzon. Fluid-structure (fsi) case study of a cantilever using openfoam and deal.ii with application to viv. Msc thesis, Lunds Institute of Technology, June 2009.

[16] M.Bergmann and L.Cordier. Control of the circular cylinder wake by trust-region methods and pod reduced order models. Technical report, Institut National De Recherche en Informatique et en Automatique, June 2008.

[17] P. Nithiarasu. Computational fluid dynamics. Lecture notes, 2012. MSc in Computational Mechanics, Swansea University.

[18] OpenFOAM Foundation. *OpenFOAM User Guide*, version 2.1.1 edition, May 2012.

[19] J.N. Reddy. *An Introduction to the Finite Element Method.* McGraw-Hill, 3rd edition, 2006.

[20] G. Romanelli, E. Serioli, and P. Mantegazza. An accurate invicit compressible solver for aerodynamic applications. In G. D'Errico, editor, *3rd OpenFOAM Workshop*, 2008.

[21] M. von Scheven and E. Ramm. Strong coupling schemes for interaction of thin-walled structures and incompressible flows. *International Journal of Numerical Methods in Engineering*, 87(1-5):214–231, 2010.

[22] W.A. Wall and E. Ramm. Fluid-structure interaction based upon a stabilized (ale) finite element method. In S.R Idelsohn, E.Oñate, and E.N Dvorkin, editors, *Computational Mechanics - New Trends and Applications*, 1998.