Immersed boundary method for computational fluid dynamics

A review and verification in OpenFOAM

Ville Väisänen

School of Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 22.12.2019

Supervisor

Prof. Ville Vuorinen

Advisor

D. Sc. Heikki Kahila

Aalto University
School of Engineering

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Ville Väisänen

Title Immersed boundary method for computational fluid dynamics
A review and verification in OpenFOAM

Degree programme Master's Programme in Mechanical Engineering

Major Mechanical Engineering **Code of major** ENG25

Supervisor Prof. Ville Vuorinen

Advisor D. Sc. Heikki Kahila

Date 22.12.2019 **Number of pages** 62+2 **Language** English

Abstract

Fluid-structure interaction (FSI) is a common phenomenon encountered in various computational fluid dynamics (CFD) problems. Conventional methods rely on the generation of a so-called body- or boundary-fitted (BF) grid, which conforms to the structure geometry. Typically, these grids consist of either structured or unstructured cells, of which the former are more efficient. Although, generating a good quality structured BF grid might be straightforward for simple geometries, it may become tedious and take even months to do using the best available software when complex geometries are present. Furthermore, the BF grid needs to be updated at each time step when structure motion is present.

In the present work, we concentrate on an immersed boundary (IB) approach implemented into the FOAM-Extend CFD library, which is based on the OpenFOAM software platform. In the IB approach, the grid does not conform to the structure, but instead, the position of the structure is tracked separately and the equations–either continuous or discretized depending on the method in question–are modified to account for the boundary. A literature review of different IB methods is conducted and the implementation is validated by running simulations on various flow cases consisting of stationary (channel flow, pipe flow and flow over a cylinder) and moving (flow over an oscillating cylinder) structures. These results are compared to ones retrieved by using a BF grid in OpenFOAM CFD software and those found in literature. In addition, a user guide for the implemented IB method is presented.

Initialization of a flow simulation with the IB method is very quick. However, simulations show that this is acquired with the cost of efficiency and accuracy. Simulation with the BF grids was much faster and more accurate than with the IB grids. Furthermore, resolving values at the surface of a moving boundary was difficult for the IB grid. Even though the IB method does show promising results for the cases with stationary structures, the fast initialization is outweighed by the required high grid resolution and slow simulation times. This is further emphasized by problems with parallel computing, rendering the current IB implementation impractical for computationally demanding CFD simulations.

Keywords computational fluid dynamics, fluid-structure interaction, immersed boundary method, OpenFOAM

Aalto University
School of Engineering

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Ville Väisänen

Työn nimi IB-menetelmä laskennallisessa virtausmekaniikassa
katsaus ja validointi OpenFOAM-ympäristössä

Koulutusohjelma Master's Programme in Mechanical Engineering

Pääaine Konetekniikka **Pääaineen koodi** ENG25

Työn valvoja Prof. Ville Vuorinen

Työn ohjaaja TkT Heikki Kahila

Päivämäärä 22.12.2019 **Sivumäärä** 62+2 **Kieli** Englanti

Tiivistelmä

Neste–rakenne-vuorovaikutus on yleinen ilmiö, johon törmätään monissa laskennallisen virtausdynamiikan (CFD) ongelmissa. Tavanomaisessa laskentametodissa luodaan rakenteen muotoja mukaileva (eng. body-fitted – BF) hila. Tyypillisesti, nämä hilat koostuvat joko rakenteellisista tai rakenteettomista laskentatilavuuksista, joista rakenteelliset ovat tehokkaampia. Rakenteellisen BF-hilan tekeminen yksinkertaisille geometrioille on varsin suoraviivaista, mutta prosessi saattaa muuttua vaativaksi ja kestää jopa kuukausia, kun hilaa rakennettaan monimutkaisille muodoille, vaikka käytettäisiin parhaita olemassa olevia ohjelmistoja. Lisäksi BF-hila joudutaan uudelleengeneroimaan aika-askelten välissä, jos rakenne liikkuu.

Tässä diplomityössä keskitytään niin sanottuun *immersed boundary* (IB) -menetelmään, joka on saatavilla OpenFOAM-ohjelmistoon pohjautuvassa FOAM-Extend CFD-kirjastossa. IB-menetelmässä hila ei mukaile rakenteen muotoja, vaan rakenteen vaikutus otetaan huomioon virtausyhtälöissä. Työssä suoritetaan kirjallisuuskatsaus olemassa oleviin IB-menetelmiin sekä validoidaan FOAM-Extendissä oleva IB-sovellus simuloimalla erilaisia virtaustapauksia, jotka koostuvat paikallaan pysyvistä (kanavavirtaus, putkivirtaus ja virtaus sylinterin yli) ja liikkuvista (virtaus oskilloivan sylinterin yli) rakenteista. Tuloksia verrataan OpenFOAM CFD-ohjelmistosta BF-hilalla saatuihin ja kirjallisuudesta löydettyihin tuloksiin. Lisäksi, työssä laaditaan IB-sovelluksen käyttöohjeet.

Virtaustapauksen alustus IB-menetelmällä on hyvin nopeaa. Simulaatiot kuitenkin osoittavat, että nopeus on saatu tehokkuuden ja tarkkuuden kustannuksella. BF-hilat olivat huomattavasti nopeampia ja tarkempia simuloinneissa kuin IB-hilat. Lisäksi liikkuvan rakenteen pinnalla olevien arvojen määrittämisessä oli haastetta IB-hilaa käytettäessä. Vaikka IB-metodilla saadaan lupaavia tuloksia simuloidessa paikallaan pysyviä rakenteita, alustuksessa saatu nopeushyöty kumoutuu IB-menetelmän vaatiessa tiheämpää hilaa sekä hitaan simuloinnin vuoksi. Tämä korostuu entisestään rinnakkaislaskennan ongelmien vuoksi, mistä syystä FOAM-Extendin IB-menetelmää ei voi suoranaisesti suositella vaativien ja suuren kokoluokan CFD-ongelmien simuloinnissa.

Avainsanat IB-menetelmä, laskennallinen virtausmekaniikka,
neste–rakenne-vuorovaikutus, OpenFOAM

Contents

Symbols and acronyms

Latin	
A	amplitude, cross-sectional area
a	diagonal solution matrix contribution factor, polynomial coefficient
(ave)	average
b	solid
C	Courant number
C_d	drag coefficient
C_f	friction coefficient
C_l	lift coefficient
c	polynomial coefficient
D	diameter
D_H	hydraulic diameter
d	discrete Dirac delta function, displacement
$\mathbf{F}$	forcing term, deformation stress
F_d	drag force
$\mathbf{f}$	forcing function
f	face, fluid, forcing term, frequency
G	ghost-cell
$\mathbf{g}$	external force
$\mathbf{H}$	transport part of the Navier–Stokes equations
h	channel height
K	permeability of porous medium
I	image node
L	circulation bubble length, characteristic length scale, entrance length
M	mass
$\mathbf{n}$	component of the unit normal vector
n	normal coordinate, time step
$\mathbf{P}$	pressure field
P	perimeter
p	kinematic pressure
r	radial coordinate, radial distance
Re	Reynolds number
(rms)	root mean square
$\mathbf{S}$	surface normal vector
S	surface area
St	Strouhal number
t	tangential coordinate, time
$\mathbf{U}$	velocity vector field
U_∞	freestream velocity
U_τ	friction velocity
U_x	velocity component in the x-coordinate direction

$\mathbf{u}$	discrete velocity vector
u_{tan}	tangential velocity component
$\mathbf{v}$	velocity vector of boundary, velocity vector at the IB face
$v_{n_{ib}}$	velocity of the interpolated IB cell
v_p	fluid cell velocity
W	channel width
$\mathbf{X}_k$, $\mathbf{x}_k$	position vector of fibre k
$\mathbf{x}_k^{eq}$	position vector of the equilibrium point for the elastic fibre k
x'	coordinate normal to the boundary
x_i	distance of the internal boundary node from the boundary surface
x_o	distance of the external boundary node from the boundary surface
y'	coordinate tangential to the boundary
y^+	dimensionless wall distance

Greek

α, β	free variable in the feedback forcing approach
Γ, Γ_b	boundary of solid b
Δ	change
Δy_1	first cell height
δ	Dirac delta, cell size
Θ_O	slope of the surface normal at boundary node O
κ	spring coefficient
λ_s	shedding vortex period
μ	dynamic viscosity
ν	kinematic viscosity
ξ	weighing factor
ρ	density
τ_w	wall shear stress
ϕ	fluid flow variable
ψ_b	cell volume weighing coefficient
Ω	domain
ω_i	weighing factor

Acronyms

2D	two-dimensional
3D	three-dimensional
BF	body-fitted, boundary-fitted
BC	boundary condition
CFD	computational fluid dynamics
CFL	Courant–Friedrichs–Lewy condition
CPU	central processing unit

DNS	direct numerical simulation
FFT	fast Fourier transform
FSI	fluid–structure interaction
FTR	OpenFOAM triangulated format
FV	finite-volume
GDI	gasoline direct injection
IB	immersed boundary
IBS	immersed boundary surface
LES	large-eddy simulation
STL	stereolithography

1 Introduction

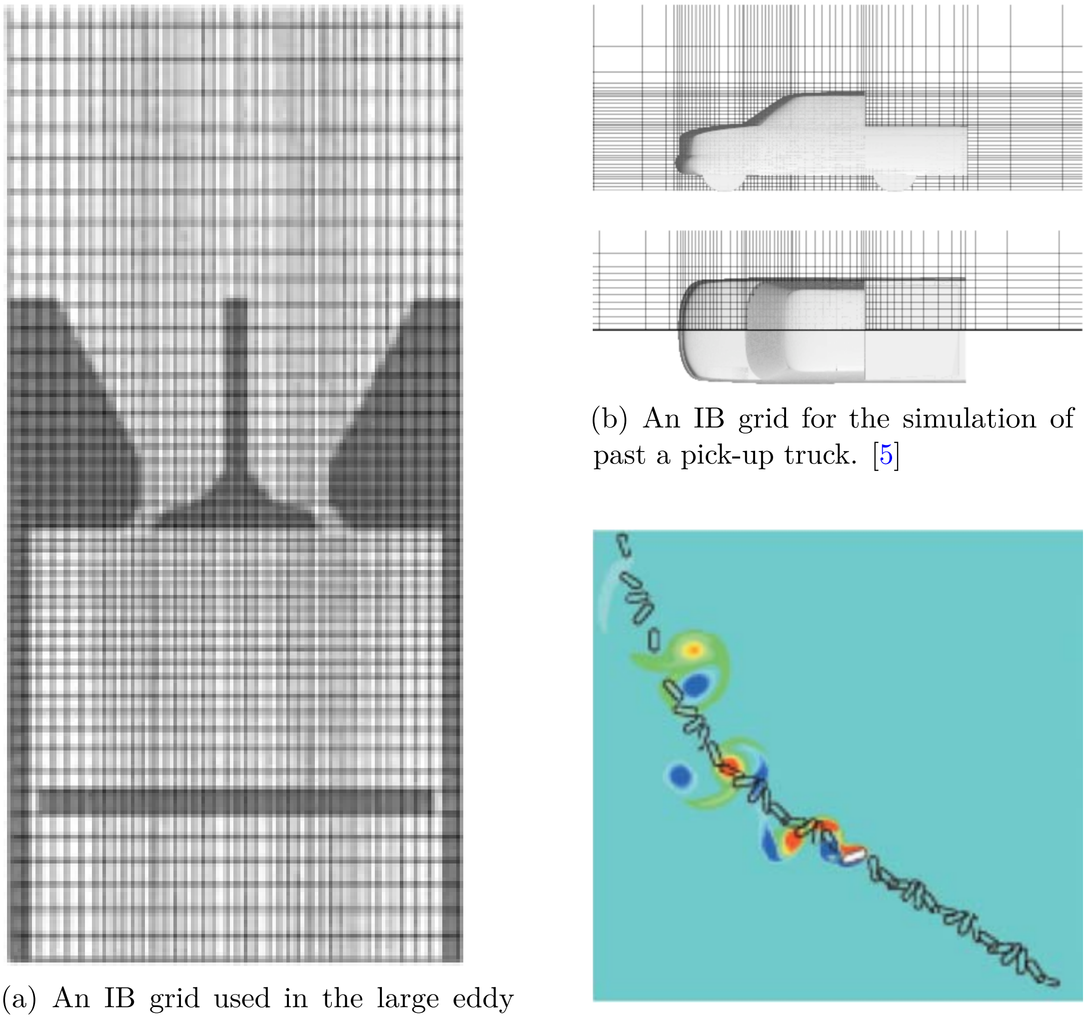
Fluid dynamics around solid surfaces have been of interest in various technical appliances throughout the history of the branch. The interactions can vary from a fairly simple case of wind blowing across a chimney or flow of liquid inside a pipe; to something more complex, such as the exhaust gas flow over a turbine blade in a jet engine or the combustion process inside of a reciprocating engine. Phenomena encountered in these flow cases can be modelled to a necessary degree of accuracy with the aid of computational fluid dynamics (CFD) to e.g. calculate lift and drag forces of a turbine blade or to optimize the spray formation of the fuel-air mixture injected into the cylinder.

These types of flows are often addressed as fluid-structure interaction (FSI) problems. [1] Typically, such processes are driven by forces following known conservation laws, characterized by continuous differential equations. In order to make predictions with modern computers, these continuous equations must be discretized into a discrete format where floating-point operations can be executed. In such a process, the spatial domain of interest is often divided into three-dimensional (3D) cells, forming a computational grid or mesh. Conventionally, the problems are solved with a computational grid that conforms to the solid body. The generation of a suitable grid for a very complex 3D geometry may prove to be very difficult. However, a different approach called the immersed boundary (IB) method may be used instead to overcome the problems associated with complex geometries.

Solving CFD problems requires a dedicated software. The OpenFOAM toolbox was developed to satisfy the need of an open source CFD solver, that can handle problems in academic research and industry. [2] OpenFOAM has branched into three distinct packages, of which one, the FOAM-Extend Project, [3] has the capability of solving fluid problems with the IB method.

1.1 Immersed boundary methods

One may encounter problems with complex structure geometries when using conventional body-fitted (BF) grid generation methods discussed in chapter 2. An interesting alternative is the immersed boundary (IB) method, in which the boundary conditions are imposed mathematically at the separately tracked boundary area. In contrast to the BF method, where the grid extremities are located at the structure boundaries, the IB grid does not conform to the structure. One example of an IB grid with complex geometry can be seen in figure 1.1a, with the dark grey areas indicating solid structures. The aforementioned grid was used to run large-eddy simulation (LES) of in-cylinder flow with a moving piston and valve. [4] Furthermore, figure 1.1b shows the grid used to simulate air flow past a pick-up truck [5], and figure 1.1c is a visualization of results obtained using an IB technique in the simulation of a free falling plate [6].



(a) An IB grid used in the large eddy simulation of in-cylinder flow. [4]

(b) An IB grid for the simulation of flow past a pick-up truck. [5]

(c) Free-falling plate. [6]

Figure 1.1: Some examples of simulations using IB techniques.

The IB method enables the use of structured grids on complex geometries. As an extreme example, a grid consisting of only Cartesian hexahedra (cubes) can be used in combination with an IB technique, to represent a complex structure. In principle, the generation of the grid in question would be very straightforward and quick [1] and the grid would be very versatile while still possessing the accuracy advantages of a structured grid. [7] Moreover, there would be no skewness or distortion in the grid cells. Using a Cartesian grid eliminates the need for curvilinear coordinate transformation, possibly reducing the amount of required operations depending on the implementation. [6] However, since the grid points do not have to, and will not lie on the boundaries, an accurate method to resolve the boundary inside the cells is required. [8] Additionally, adaptive mesh refinement will have to be done near the boundary areas to keep the number of calculation cells reasonable. [7]

Overall the IB method could be an attractive method for product design purposes, as it is able to accommodate for the ever-changing product geometry. The immersed boundary method will be discussed in detail later in chapters 4 and 5.

1.2 Objectives of the thesis

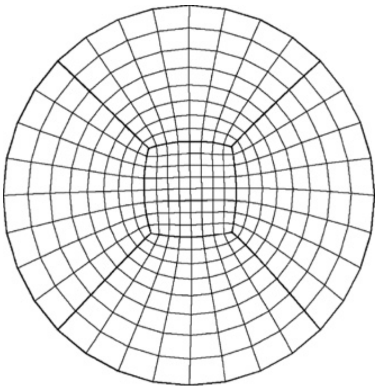
This thesis can be seen to have four distinct goals. The primary aim of this thesis is to validate the IB method implemented by Jasak et al. [9] in the FOAM-Extend Project [3] and research the possibility of using that method in various LES flow cases with complex geometries. The validation process will start with stationary flow cases consisting of laminar channel flow (sections 6.1 and 6.2) and flow around a cylinder (section 6.3). Then, the process is continued by introducing motion into the simulations via a case of an oscillating cylinder in cross flow (section 6.4). The secondary goal of the thesis is to present the operation of the immersed boundary method (section 5.2) and lay out the theory behind it (section 5.1). The third objective is to serve as a guide to immersed boundaries in FOAM-Extend Project for the future user (section 5.3). Finally, the last objective is to conduct a literature review of past immersed boundary implementations (chapter 4).

The thesis is organized as follows. We start by looking into conventional methods of grid generation in chapter 2. Then, chapter 3 introduces the governing equations and finite-volume methods associated with incompressible viscous flow. In chapter 4, the immersed boundary method is discussed in detail. Next, the method behind the IB adaption to FOAM-Extend Project is laid out in chapter 5. The validation tests are defined in chapter 6 and their results are presented in chapter 7. An analysis of the obtained results is done in chapter 8. Finally, the conclusions are presented in chapter 9.

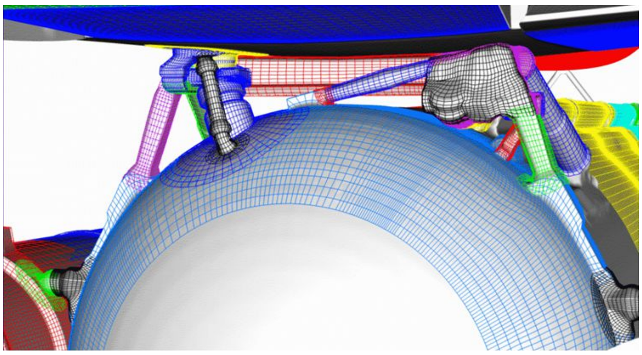
2 The computational grid

A feasible computational method must be accurate, efficient and be able to handle complex geometries. [10] Clearly, the modelling of the fluid-structure interaction plays a vital role in the outcome of the simulation. Excluding the use of meshless methods, which will not be discussed here, the first step in setting up a computational framework is to generate a computational grid. The quality of the grid has an important influence on the quality of the final calculations, [11] which will be further discussed in section 2.1.

When solving fluid flow problems, the conventional approach is to generate a grid, whose boundaries lie at the extremities of the structure, [6] i.e. the surface of the channel corresponds to the edges of the mesh (see figure 2.1a). The generation of a good quality grid this way is quite straightforward for simple flow geometries, such as the flow over a flat plate. However, when the geometries get more complex, so does grid generation, see figure 2.1b. This can manifest as long grid generation times, degraded solution accuracy and increased computation times. [11]



(a) The cross-section of a simple body-conformal pipe flow grid with O-H topology. [11]



(b) The body-conformal mesh for a space shuttle fastening system. [12]

Figure 2.1: Grids with different degrees of complexity.

Nowadays, there are automated unstructured grid generation tools that can be used for complex geometries, [11] but these will not be discussed here. Instead, we will look at the different types of grid techniques that are available and are also used within the automatic grid generation tools. When deciding what type of grid to generate, the first step is to choose between a structured or unstructured grid.

2.1 Spatial discretization and grid generation

2.2 Structured grids

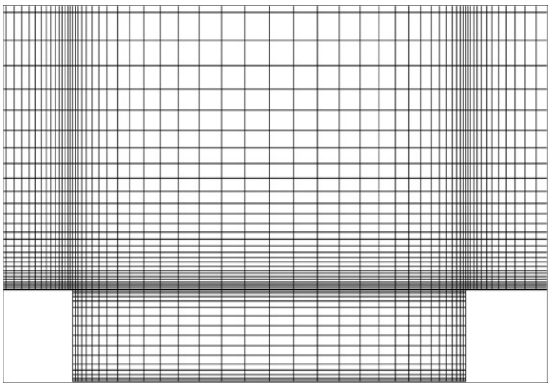
A structured grid can be identified by its regular connectivity. It is composed of quadrilateral and hexahedral elements in 2D and 3D, respectively, so that each node is intersected by one line in every dimension (see figure 2.1a). The structured grid is the most natural choice for flow simulation as one can easily imagine the fluid flow to be aligned with its gridlines. As a matter of fact, the accuracy, processing time and memory requirements are often much better with structured grids than with unstructured ones. [11] However, their weakness lies in complex geometries. It may take from weeks to months to generate a good quality structured grid for a complex geometry using the best available software. [11] Furthermore, it is tedious to generate structured grids automatically without prior knowledge of the involved topologies. [11]

The most accurate mesh follows a Cartesian distribution, where every cell is a perfect cube. However, doing this in practice for complex geometries is impossible. Thus, the grid needs to be bent to follow the structure, introducing a curvilinear coordinate system.

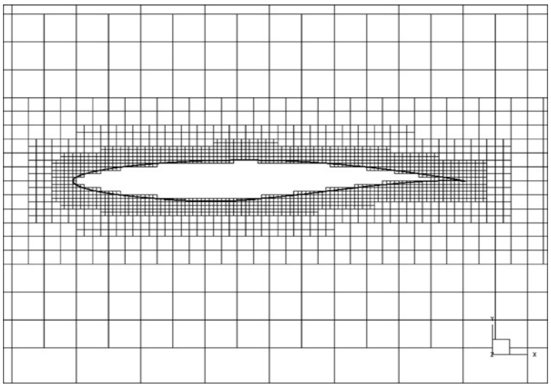
2.2.1 Non-uniform Cartesian grids

Possibly the simplest extension to grid modification is to vary the original grid spacing. In particular, the near-wall flow regime is utmost important for many internal and external flow cases. Turbulence is generated by the surface shear, hence accurate capturing of velocity gradients near the wall is important, and that is where structured grids often outperform their unstructured counterparts. [11] Non-uniform grid spacing enables the analysis of straight structures where a cluster of cells is necessary near the structure to resolve the boundary layers accurately. [11] The grid is then spaced out in the far field to maintain computational costs at a decent level. The increase in grid spacing for consecutive cells should be kept at less than 20 %. [13, p. 56] Another way to handle structures with straight edges is to locally refine the grid with extra nodes, leading to a so called non-uniform grid, which can be also considered as an unstructured mesh due to locally non-conforming grid lines. Examples of varying grid spacing, and local non-uniform refinement can be seen in figure 2.2a. Varying grid spacing and local refinement preserves the structured grid nature, although care must be taken not to degrade the grid performance with cell size discontinuities and extreme aspect ratios. [11]

Using Cartesian grids to handle curved structures may be tricky. One method is to approximate the curved boundary surface with a staircase shaped grid as can be seen in the case of an aerofoil in figure 2.2b. Two other methods also exist, one being the immersed boundary method, and the other the cut-cell method, [11] which is very much like the immersed boundary method, and thus both will be discussed later in chapter 4.



(a) A simple, non-uniform Cartesian grid around a cavity. [11]

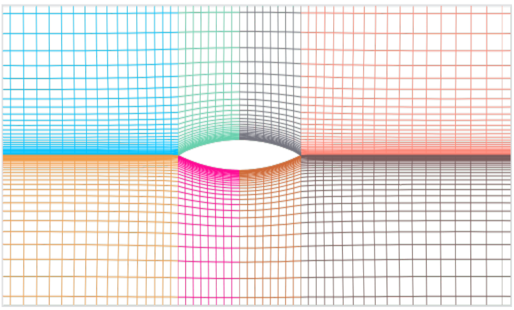


(b) The staircase boundary approximation for a curved geometry with a Cartesian grid. [11]

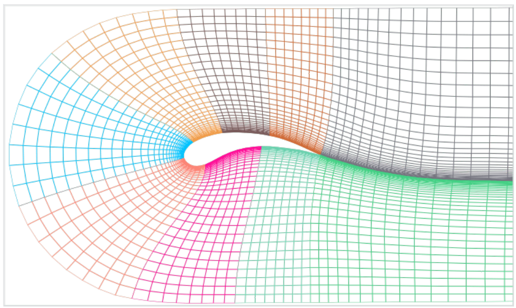
Figure 2.2: Body-conformal Cartesian grids.

2.2.2 Structured curvilinear grids

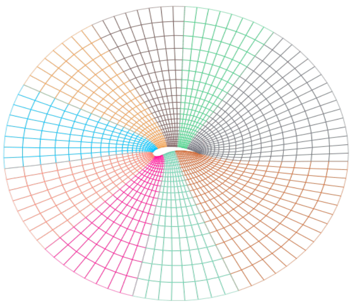
The structured grid can be made to conform to a curved body by bending it and introducing a curvilinear coordinate system. These body-fitted or boundary-fitted (BF) grids are categorized into H-, C-, O- and I-meshes depending on the orientation of grid lines. [11] Combinations of these configurations are also possible, leading to multi-block grids which will be discussed briefly in the next section. Figures 2.3a-2.3c show the difference in appearances with various types of body-fitted grids.



(a) H-topology



(b) C-topology



(c) O-topology

Figure 2.3: Body-fitted grids with different topology configurations.[14]

2.2.3 Multiblock grids

The aforementioned methods are not enough to generate a good quality mesh when working with sharp interface topology. It is possible to combine multiple grid areas, called blocks, using different meshing techniques, to achieve sufficient quality. [11] The result is a multi-block grid configuration, which have been adapted to multiple areas of engineering. [15] As stated above, a combination of BF grids can be used, e.g. C-H, O-H or H-O-H type meshes. An O-H type multiblock grid, also known as a "butterfly" type grid, [11] can be seen in figure 2.1a.

The treatment of block interfaces is important when using multi-block grids. The normal solution is to use matching mesh lines so that the grid is continuous. However, it may be attractive to use non-matching block interfaces to gain better flexibility, i.e. a traditional cell splitting approach that can be seen in figure 2.2b. As with the non-conformal meshes, this leads to a requirement for the solver to transfer information accurately from one block to another using interpolation routines. [11] Multi-block grids are also available for unstructured grids, which will be discussed later. The reason for the multi-block grid choice with unstructured grids, however, would not be the accommodation of structure topography but the parallelization of the computational process. [16]

Figure 2.4 shows an example of a multiblock grid used in the simulation of a gasoline direct injection (GDI) fuel spray. When the needle inside of the injector moves during the spraying operation, the gap where fluid flows may become as small as 5 microns. As one can imagine, the generation of a good quality multiblock grid that can accommodate such large changes in flow passage scale is a cumbersome and time-consuming process.

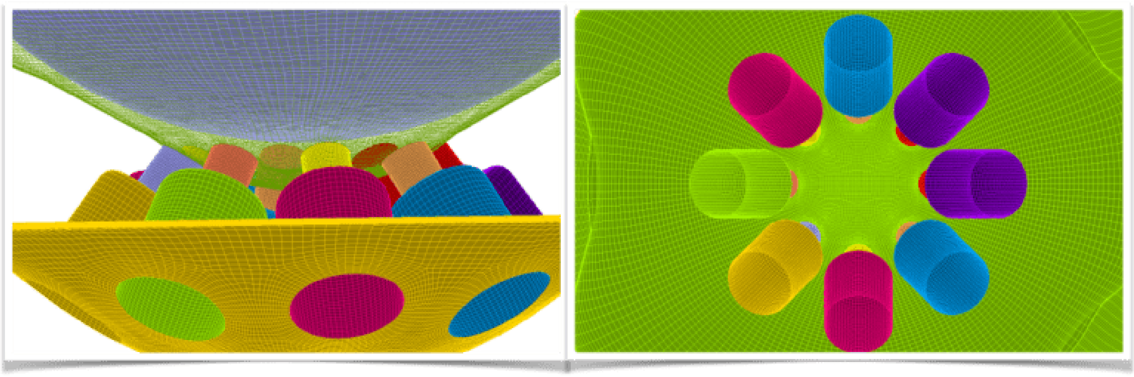


Figure 2.4: A multiblock grid used in GDI spray simulation with different colours corresponding to different blocks. [17]

2.2.4 Patched and overset grids

The patched grid method proposed by Rai [18] decomposes the background grid into subdomains around bodies of interest. These patched areas are refined to accurately resolve the geometry in question. The subdomains of the patched grid do not overlap at their boundaries.

The overset grid, or chimera grid, proposed by Steger et al. [19], has become one of the most important techniques for aerodynamic simulations. They can be seen as patched grids where subdomains overlap one another. [20] The idea is to generate independent grids around bodies of interest—be it moving or stationary—and overlap them with the fixed background grid. These overset grids can be moved around according to the position of their respective structure during simulations. Once again, accurate interpolation procedures must be provided, which can be extremely difficult for three-dimensional overset grids. [11]

2.3 Unstructured grids

Grids with an arbitrary distribution of mesh points using, but not limited to, triangles in 2D and tetrahedra, pyramids or polyhedra in 3D are called unstructured grids. Due to the nature of the elements, unstructured grids can accommodate complex geometries exceptionally well. They have progressively become the more attractive approach to industrial simulations as automatic grid generation for arbitrary geometries is readily available for unstructured grids. [11] Furthermore, local refinement of unstructured grids can be done without affecting the surrounding region. However, the downside of increased flexibility is the decreased accuracy within the interpolation routines [11] required in the solution of discretized Navier–Stokes terms, further depicted later.

2.3.1 Hybrid grids

The resolution of boundary layer flows around structures is very difficult with triangles or tetrahedra. The aspect ratio should be of order $\delta x/\delta y \sim \sqrt{Re}$ near the surface to optimize accuracy and simulation cost. [11] The dimensionless number Re is the Reynolds number that characterizes the flow. δx and δy represent cell sizes in the streamwise and normal directions, respectively. This may lead to a poor-quality unstructured mesh, as e.g. the height-to-base ratio of tetrahedra or triangles increases with increasing Reynolds number leading to distorted elements. [11] To counter this, the boundary area can be modelled with layers of quadrilaterals, hexahedra or prisms, which can withstand higher aspect ratios. [21] The grid continues with unstructured elements after the structured layer, and thus is called a hybrid grid. An example of a hybrid mesh can be seen in figure 2.5.

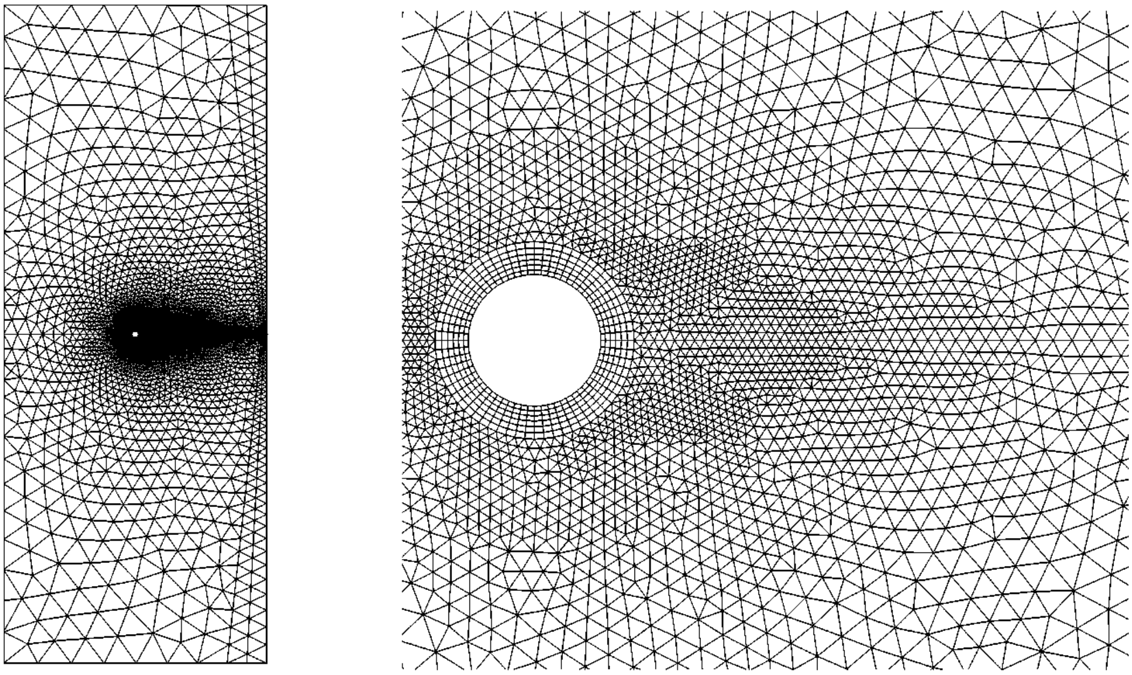


Figure 2.5: A hybrid grid with a structured O-type topology in the boundary layer of the cylinder and unstructured cells elsewhere. [22]

2.4 Moving and deforming structures

Dedicated techniques need to be applied to resolve the fluid-structure interface during body movement and/or deformation. For flow problem with solid surfaces using conventional grids, this usually means an implementation of an interface-tracking technique. [23]

When interface-tracking is used, the deformation of grid points is calculated, and the grid is updated as simulation progresses. Essentially, this requires either full or partial remeshing, [23] either by deleting the old mesh and generating a new one, or by dynamically deforming the old mesh. [24]

This whole process greatly increases computational cost due to an increased number of operations, since the old solution must be mapped onto the newly generated mesh each time step in addition to the mesh regeneration. [8] Furthermore, problems arise if cells deform extensively, leading to large cell ratios, skewness or node overlapping or crossing. [24] Due to the nature of the fluid flow simulation, it is clear that manual remeshing is not a feasible solution, thus various mesh deformation techniques have been studied, some more robust than others, while others computationally more efficient. [24] The ensemble of these techniques is numerous and will not be discussed here in further detail.

3 The governing equations

In this chapter, we introduce the governing equations for incompressible viscous flow. The flow is described by the continuity equation and the Navier–Stokes equation of motion.

3.1 Continuity equation

First, we consider the mass in a fluid system. A natural fact is that mass must be conserved and is independent of the fluid and forces acting on it. [25, p. 40] This means that a local change in mass must be accounted for by the convective transport of mass elsewhere, i.e.:

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{U}) = 0 \ , \tag{3.1}$$

where ρ is the density of the fluid and $\mathbf{U}$ is the velocity vector. Equation (3.1) is called the *continuity equation*.

As the systems in question for this thesis are incompressible, the density is independent of time and (3.1) reduces to:

$$\boldsymbol{\nabla} \cdot \mathbf{U} = 0 \tag{3.2}$$

3.2 Navier–Stokes equations of motion

The conservation of momentum in a fluid system is related by the *Navier–Stokes equation of motion*:

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{U} \otimes \mathbf{U}) = -\nabla \mathbf{P} + \mu \nabla^2 \mathbf{U} + \frac{1}{3} \mu \nabla \left(\boldsymbol{\nabla} \cdot \mathbf{U} \right) + \rho \mathbf{g} \ , \tag{3.3}$$

where $\mathbf{U}$ is the velocity vector, $\mathbf{P}$ is pressure, ρ is the density, μ is the dynamic viscosity and $\mathbf{g}$ is an external force, e.g. gravity, affecting the fluid. This equation is true for fluids with constant dynamic viscosity coefficients. The left hand side of equation (3.3) corresponds to inertial forces consisting of the variation of density ρ and velocity $\mathbf{U}$ in addition to the convection of mass. Overall, they describe the forces acting on each fluid particle. The right-hand side shows the forces which consist of the pressure effects $-\nabla \mathbf{P}$, the viscous effects and the external force $\mathbf{g}$ acting on the fluid particles.

By assuming the density stays constant and the flow is incompressible, we can divide equation (3.3) with ρ and omit the term containing $\boldsymbol{\nabla} \cdot \mathbf{U}$ as per the continuity equation it equals to 0.

$$\frac{\partial \mathbf{U}}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{U} \otimes \mathbf{U}) = -\nabla \mathbf{p} + \nu \nabla^2 \mathbf{U} + \mathbf{g} \tag{3.4}$$

In equation (3.4), $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity and $\mathbf{p} = \frac{P}{\rho}$ is the kinematic pressure.

3.3 Pressure equation

The above equations are discretized to obtain a finite set of governing equations, which can be then solved with modern computers. Equations (3.1) and (3.4) are discretized and combined to form the pressure equation. The final form of the discretized Navier–Stokes system for a single cell has been derived by Jasak [26]:

$$a_P \mathbf{U}_P = \mathbf{H}(\mathbf{U}) - \sum_f \mathbf{S}(p)_f \ , \tag{3.5}$$

$$\sum_f \mathbf{S} \cdot \left[\left(\frac{1}{a_P} \right)_f (\boldsymbol{\nabla} p)_f \right] = \sum_f \mathbf{S} \cdot \left(\frac{\mathbf{H}(\mathbf{U})}{a_P} \right)_f \tag{3.6}$$

where $\mathbf{S}$ denotes the surface normal vector of face f and $\mathbf{H}$ is the transport part of the Navier–Stokes equation. The coefficient a_P is the diagonal contribution to the solution matrix and naturally is a function of $\mathbf{U}$. The flux F through any face f can be calculated from

$$F = S \cdot \mathbf{U}_f = \mathbf{S} \cdot \left[\left(\frac{\mathbf{H}(\mathbf{U})}{a_P} \right)_f - \left(\frac{1}{a_P} \right)_f (\boldsymbol{\nabla} p)_f \right] \tag{3.7}$$

The discretization procedure will not be described in further detail, but the interested reader is referred to the seminal paper of Jasak [26].

The dependence of the velocity on the pressure and vice versa, which can be seen from the discrete forms in equations (3.5) and (3.6), requires special treatment. [26] This treatment follows a Rhie–Chow type stabilization.

4 The Immersed Boundary (IB) method

This chapter serves as the introduction to the immersed boundary (IB) method which we will use to solve flow cases in chapter 6 with the FOAM-Extend Project CFD toolbox [3] that will be introduced in chapter 5.

4.1 General

The immersed boundary method is group of numerical methods that consist of manipulating the governing equations (discretized or not) to account for the boundary in question. Over time, a large variety of methods which can be classified as immersed boundary methods have been developed. The division of these methods is ambiguous and difficult. In this thesis, they have been classified according to how the boundary conditions are imposed into the governing equations.

In general, the momentum equation, equation (3.4), is modified with a forcing function which serves as a source term. The point at which the forcing function is introduced to the equations divides the IB methods into two distinct categories: the continuous forcing approach and the discrete forcing approach.

The earliest form of IB methods is the continuous forcing approach, which will be discussed in the next section. In this approach the forcing function is modified directly into the continuous governing equations before discretization. If a solid b occupies the domain Ω_b and is surrounded by a fluid domain Ω_f so that the boundary in between them is denoted by Γ_b, the governing equations for a continuous approach IB method can be written as

$$\nabla \mathbf{U} = 0 \qquad \text{in } \Omega_{\mathrm{f}} \tag{4.1}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{U} \otimes \mathbf{U}) + \nabla \mathbf{p} - \nu \nabla^2 \mathbf{U} = \mathbf{f} \tag{4.2}$$

$$\mathbf{U} = U_\Gamma \qquad \text{on } \Gamma_{\mathrm{b}}, \tag{4.3}$$

where $\mathbf{f}$ is the forcing function. The next step is to define $\mathbf{f}$, which also further divides this category.

The second category is the discrete forcing approach detailed in section 4.3. Here, the governing equations are first discretized without considering the immersed boundary. After this, the IB is imposed by adjusting the discretization in the cells near the boundary. The partially discretized Navier–Stokes equation can be expressed as

$$\frac{u^{n+1} - u^n}{\Delta t} = -\left(\mathbf{u} \cdot \nabla\right) \mathbf{u} - \nabla \mathbf{p} + \nu \nabla^2 \mathbf{u} + \mathbf{f}_b \ , \tag{4.4}$$

where n denotes the time step. To solve the required force, we assume that the velocity at the boundary node corresponds to the velocity of the boundary, i.e. $u^{n+1} = \mathbf{u}_\Gamma$. The forcing term is then simply

$$\mathbf{f}_b = \left(\mathbf{u}^n \cdot \nabla\right) \mathbf{u}^n + \nabla \mathbf{p}^n - \nu \nabla^2 \mathbf{u}^n + \frac{\mathbf{u}_\Gamma - \mathbf{u}^n}{\Delta t} \tag{4.5}$$

at points where the boundary velocity is imposed. Elsewhere, the forcing term reduces to zero. Equation (4.5) works only as is for grids aligned with the boundary, hence interpolation procedures are required for cells that are intersected by the boundary. Different approaches in the discrete forcing method are identified by their manner of velocity reconstruction and will be discussed later.

4.2 Continuous forcing approach

Originally, the immersed boundary (IB) was conceived by C.S Peskin [27] as part of his doctoral thesis in 1972. Peskin studied the blood flow patterns around the mitral valve in the human heart. The dynamic interactions between the coronary valve structure and the blood flow was known to be essential and, thus had to be included in simulations. [27, pp. 17-18] Due to the shortcomings of a body-fitted mesh to accommodate for structure movement, and the state of contemporary computational resources, an innovative way to handle deforming, elastic structures was required. [28] Peskin's approach came in the form of a numerical tool, which allowed the study of the opening and closing heart valve to be done in a Cartesian mesh with the structure immersed inside of the mesh instead of being located at the edge. Contrary to the classical approach, where the boundaries constrain the fluid flow, the boundary areas were now regarded as specific regions inside the computational domain to which extra force terms were introduced. [27, p. 12] Peskin argued that the heart valve could be modelled as elastic fibres subject to stress which deformed via Hooke's law. These fibres were tracked in a Lagrangian fashion as a set of massless points over the background grid and would move according to the local flow velocity. For an elastic fibre k, we can write:

$$\frac{\partial \mathbf{X}_k}{\partial t} = \mathbf{u}(\mathbf{X}_k, t) \tag{4.6}$$

where $\mathbf{X}_k$ is the position vector of the fibre, t is time and $\mathbf{u}$ is the velocity vector. The effect of the fibres was transmitted to the fluid as a source term in the momentum equation at grid positions corresponding to the set of points:

$$\mathbf{f}_m(\mathbf{x}, t) = \sum_k \mathbf{F}_k(t)\delta(\mid \mathbf{x} - \mathbf{X}_k \mid) \tag{4.7}$$

where the deformation stress is denoted by $\mathbf{F}_k$ and δ is the Dirac delta function. In equation (4.7), the Dirac delta function is a sharp function, i.e. it has a value of 1 only on the boundary and 0 elsewhere. Unfortunately, the boundary gets smeared and forcing is distributed to the surrounding cells since the fibres do not necessarily align with the nodal points of the Cartesian grid as is described in figure 4.1. This in turn forces the sharp delta function into a smoother function d, which is more suitable for a discrete mesh. As can be imagined, the simulation accuracy is strongly dependent on the choice of the smooth distribution function, and thus multiple distribution functions have been developed and compared. [27, 29, 30, 31] Figure 4.2 shows some of these smooth distribution functions. The accuracy of this method was described to be of first order by the authors and worked only for elastic structures. This approach would later be known as the continuous forcing approach. [6]

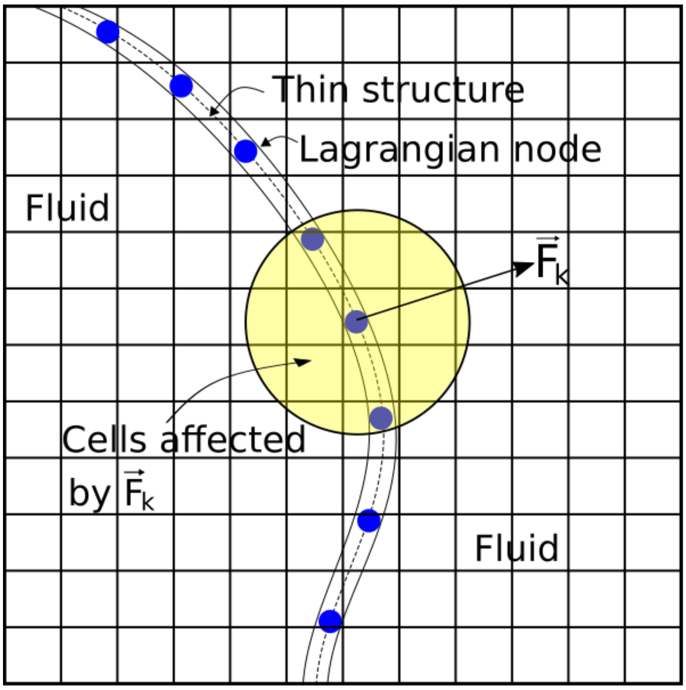


Figure 4.1: Boundary smearing around a nodal point with the forcing term affecting a band of surrounding cells. Illustration follows that of Kim et al. [22].

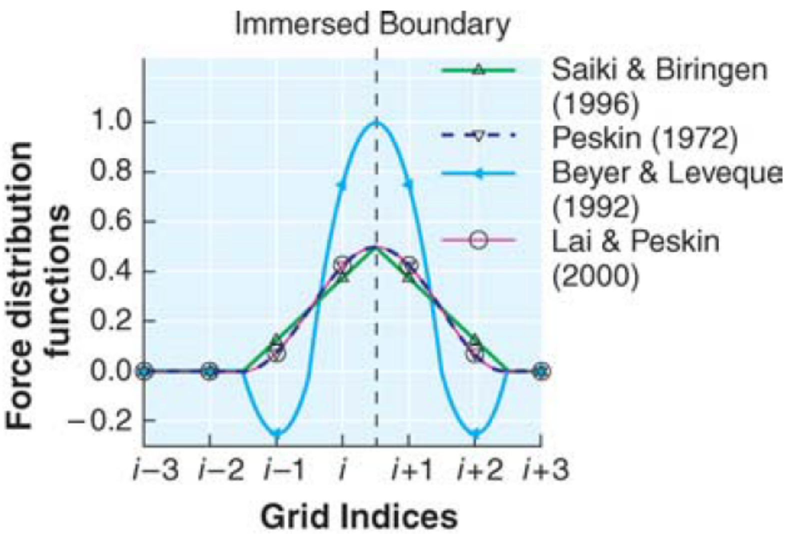


Figure 4.2: A comparison of different smooth distribution functions d used for the treatment of the near-boundary nodes. [6]

As the above formulation can handle only massless elastic boundaries, it was later developed to account for the mass of an elastic body by introducing the d'Alembert force $(-M\frac{\partial^2 x}{\partial t^2})$ to the external force in the momentum equation. [32] Although well suited for elastic boundaries, this method still could not be applied to rigid boundaries, as the constitutive laws used do not generally work at the rigid limit. [6]

The derived IB method was suitable for elastic bodies and thus was second-order accurate in space and time only for smooth solutions. However, solutions to problems with sharp interfaces, such as flow around a cylinder or any other rigid surface, are not smooth fundamentally i.e. a single point can be observed where the fluid changes to solid. When using this IB method, the fluid-structure interface gets smeared to a thickness of the order of nearest cell width and, thus does not correlate well with the actual flow. [29]

Further development was done by Goldstein et al. [33] who combined the IB with spectral methods and changed the way the boundary force was introduced into the equations of motion from momentum forcing to feedback forcing. Essentially, it was still a continuous forcing approach, with the exception that the boundary was modelled as a forcing term which exerts just enough force to halt the fluid at the surface of the virtual boundary. The force must be calculated by a feedback loop, since it is not known beforehand. The forcing term to be used in equation (4.2) would be

$$\mathbf{F}(t) = \alpha \int_0^t \mathbf{u}(\tau) d\tau + \beta \mathbf{u}(t). \tag{4.8}$$

This approach required the fine tuning of two free variables α and β which are dependent on flow frequencies. When they are large enough and negative, the system will keep $\mathbf{u}$ close to the desired value. [33] As might be expected from a feedback mechanism, equation (4.8) can be also seen as a proportional-integral controller. Unfortunately, the use of feedback forcing manifests as severe stability issues unless a very small time step is used. For context, Goldstein et al. were able to run simulations with a Courant-Friedrich-Lewy number (C) of $\mathcal{O}(10^{-3} - 10^{-2})$. [4] Furthermore, spurious flow oscillations may be present. [4] Despite the feedback forcing method being successfully used to simulate the start-up flow of an impulsively accelerated cylinder and turbulent flow inside a rectangular channel, [33] it has trouble modelling the boundary precisely. The feedback controller (equation (4.8)) may not respond quick enough to changing flow to keep the boundary constant. In addition, determining pressure and viscous forces at the surface may be difficult. Overall, the method is promising for low Reynolds number flows, but high Reynolds number flows require accurate boundary resolution, which in turn forces large values for α and β causing stability problems. [6]

The feedback forcing method was further developed by Saiki et al. [31] so that spurious oscillations were eliminated. This was done by modifying the forcing function to an area-weighted average function

$$\mathbf{F} = \alpha \int_0^t \left[\mathbf{u}(\tau) - \mathbf{v}(\tau)\right] d\tau + \beta \left[\mathbf{u}(t) - \mathbf{v}(t)\right] \tag{4.9}$$

In this improved method the velocity of the body is controlled by specifying $\mathbf{v} = (u_b, v_b)$ at the boundary points. With these modifications, the feedback forcing method was shown to be capable of handling even moving solid boundary problems. [31]

Another way of looking at the solid boundary while keeping the original elastic system idea is to consider it as a spring system. [30, 29] That is to say, each boundary point is connected to a spring at equilibrium. Now, when the fluid acts on that point, the spring will compress or extend and act back on the fluid according to a restoring force

$$\mathbf{F}_k(\mathbf{x}, t) = -\kappa \left(\mathbf{x}_k(t) - \mathbf{x}_k^{eq}\right) \tag{4.10}$$

where $\mathbf{x}_k^{eq}$ denotes the equilibrium position of the boundary point and the spring coefficient κ is chosen to be very large, i.e. $k >> 1$, for a rigid boundary. [30] This method allowed an accuracy of second order for smooth surfaces but continued to show first-order accuracy for rigid bodies. Navier–Stokes flow simulations of cylinder in cross flow agreed considerably well with experimental data. [29] In the end, this method is a special case of the feedback forcing method. By choosing $\beta = 0$ in equation (4.8) and integrating from 0 to t we get

$$\mathbf{F}(t) = \alpha \left(\mathbf{u}(t) - \mathbf{u}(0)\right) \tag{4.11}$$

In equation (4.11) $u(0)$ corresponds to the equilibrium position $\mathbf{x}^{eq}$. Physically, the PI-controller of equation (4.8) can rather be seen as a damped oscillator, and thus, the method proposed by Beyer et al. [30] and Lai et al. [29] is subject to similar limitations and shortcomings as the feedback forcing method. The reasoning behind the stability problems and a requirement for a very small time step can be, for example, considered in the following way: to model a rigid boundary we need to use a feedback forcing coefficient α or the spring coefficient k that is very large, i.e. the solid structure is a very rigid elastic structure. Thus, if the time step permits a too large deflection of the spring system, the force exerted on the fluid particles on the next time step will also be too large, leading to a large deflection to the other side of the boundary. This in turn, will later manifest as boundary oscillations.

In addition to the methods discussed above, Khadra et al. [34] developed the so-called penalty method, in which the whole domain is assumed a porous medium and governed by the Navier–Stokes–Brinkman equations. [35] These equations contain an additional force term which is of form

$$\mathbf{F} = \frac{\mu}{K}\mathbf{u} \tag{4.12}$$

where μ is the dynamic viscosity of the fluid and K is the permeability of the porous medium. The idea is to consider the fluid as extremely permeable and have the force term vanish in the fluid domain ($K \gg 1$) and force the velocity to zero inside of the solid domain ($K \ll 1$). Furthermore, a smoothing function is once again used for the variation of K near the fluid-solid interface, smearing the boundary. Curiously, if the constants α and β in the feedback forcing method (equation (4.8)) are chosen to be 0 and μ/K respectively, the penalty method can be seen as a specific case of the feedback forcing method, subject to—once again—similar stiffness problems.

Other implementations of the continuous forcing approach include the distributed Lagrange multiplier method by Glowinski et al. [36], the immersed interface method by Lee et al. [37] and the projection approach by Taira et al. [38]. However, these will not be detailed in this work.

Overall, the continuous forcing approach is successful in simulating FSI problems with elastic boundaries as it possesses a correct physical basis. Implementation of the approach into existing solvers is quite straightforward, since modifications are done directly to the governing equations, not to the discretized equations. Different methods of continuous forcing have been successfully used in a variety of problems with elastic boundaries, such as the modelling of flow around heart valves [39] and the collision of bubbles [40]. The simulation of rigid boundaries is limited to low Reynolds number flows and has been done e.g. for flow past a cylinder. [34]

4.3 Discrete forcing approach

The other main branch of the IB methods is the discrete forcing approach, which can be divided into categories depending on the method of velocity imposition. This section consists of the direct boundary condition imposition and the ghost-cell method. The latter is the closest to the method implemented into the FOAM-Extend Project toolbox [41] detailed in chapter 5.

4.3.1 Direct boundary condition imposition

As the IB method still lacked accuracy of the second order for rigid bodies, a method which implemented the virtual boundary forces straight into the discretized equations of motion was proposed by Mohd-Yusof. [42] This new method did not require user-specified parameters in the forcing term and simultaneously eliminated the associated stability constraint. [4] It would later be known as the discrete forcing approach with direct boundary condition imposition,

The new proposition was an IB method in which the velocity field is reconstructed at the IB to satisfy the no-slip condition. In this method, the forcing function is determined from the numerical solution for which a prediction can be made using the current time step (see equation (4.5)). For a stationary boundary, Mohd-Yusof [42] proposed that the external flow field (flow in the fluid domain) is tangentially opposed by the internal flow field (flow in the solid domain). The required tangential velocity inside of the boundary is retrieved using a weighted linear interpolation scheme: [42]

$$u_{tan,i} = \xi u_{tan,o} \tag{4.13}$$

where the subscripts i and o denote the internal and external boundary points respectively, and $\xi = x_i/x_o$, where x is the distance of the corresponding point from the surface. Essentially, this forces the no-slip condition at the wall (see figure 4.3). The forcing terms would then be calculated from equation (4.5) with these velocities. This method of mirroring the velocity satisfies the boundary condition to the accuracy of the interpolation method used but causes mass conservation problems in the boundary cells. [43]

An advantage of this method is that a conventional Dirichlet boundary condition is achieved without the use of user-specified parameters. A smooth velocity field is obtained without using a discrete delta function to smooth the forcing term. [42] However, it is possible that the generated internal velocities exceed the free-stream velocities when this method is applied to significantly curved three-dimensional structures. This can then manifest as a requirement for a smaller time step. [42]

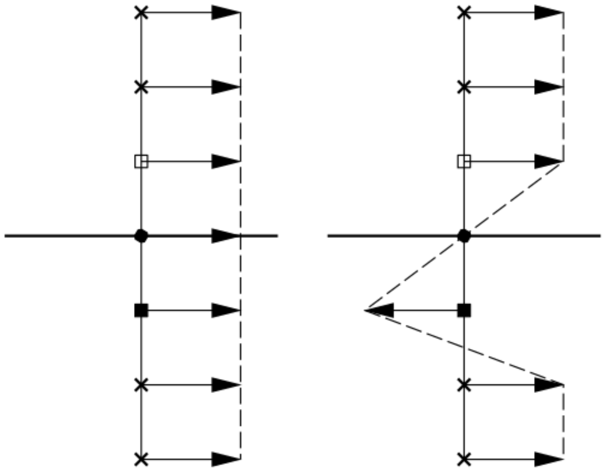
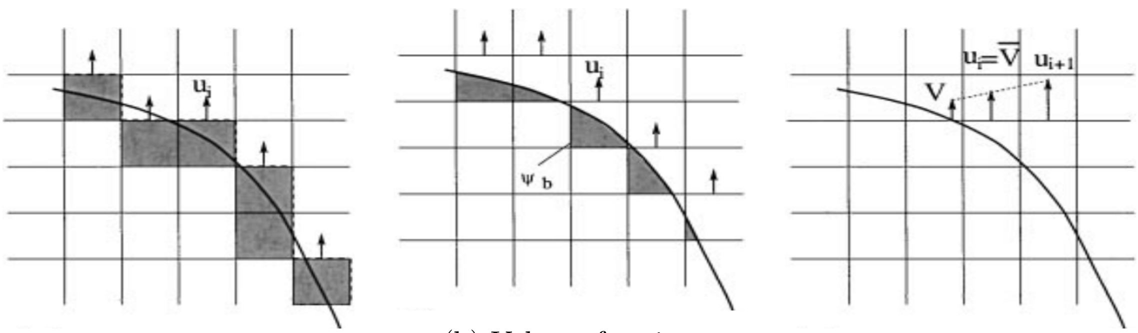


Figure 4.3: Weighted linear interpolation scheme for the reversal of the tangential velocity interior to the surface (■) using the velocity at the point exterior to the surface (□). [42]

The method of Mohd-Yusof [42] was further developed by Fadlun et al. [4] They studied three distinct methods for velocity reconstruction near the boundary (see figure 4.4). Their first method was essentially a stepwise approximation of the boundary corresponding to the grid points (figure 4.4a), which did not include interpolation. The second was a method where the forcing would be scaled by a weight coefficient ψ_b taking into account the volume of the cell being intersected by the immersed boundary, as can be seen in figure 4.4b. The third method (figure 4.4c) was a simple linear interpolation of the boundary velocity to the velocity of the first exterior node. The latter method of linear interpolation retained the second-order accuracy of the interpolation scheme and thus was superior to previous approaches. [4] However, as the velocity profile is interpolated and assumed linear, grid resolution must be fine enough near the boundary. This new approach enabled the use of a much larger time step than before and sped up the computational process by a factor of 60. [4] The combined efficiency and accuracy upgrade made the new approach suitable for complex three-dimensional flow simulations. [44]



(a) Stepwise reconstruction

(b) Volume fraction weighting

(c) Linear interpolation

Figure 4.4: Velocity reconstruction schemes used by Fadlun et al. [4]

The method was improved further by Kim et al. [43] who introduced mass source/sink terms in the cells surrounding the IB to account for the unphysical mass flux over the immersed boundary.

4.3.2 The ghost-cell finite-difference approach

Another way of identifying the boundary in the discrete forcing method is the ghost-cell approach proposed by Tseng and Ferziger. [10] The approach consists of identifying cells that have their centres located in the solid and at least one neighbouring cell located in the fluid. These so-called "ghost-cells" can be seen in figure 4.5a as cells with centres marked with a triangle. Multiple interpolation schemes are available to impose the boundary condition in the ghost-cells [45] and there is also variation in the definition of which cells are regarded as ghost-cells. [10, 6, 9] The simplest method in a two-dimensional approach would be to construct a triangular stencil with the ghost-node (G) and two nearest fluid nodes (X_1 and X_2) as the vertices. [10] Furthermore, a node on the boundary (O) should be chosen, either as the midpoint of the boundary inside the cell or a point on the boundary where the vector $\overrightarrow{GO}$ is normal to the surface. [10] A Dirichlet boundary condition could be imposed by a linear interpolation such as

$$\phi = c_0 + c_1 x + c_2 y \ , \tag{4.14}$$

where ϕ is any variable of interest and x and y are the coordinate distances to the chosen points (O, X_1, X_2). [10] The ghost-cell value would then be a weighted sum of these values per [6]

$$\sum \omega_i \phi_i = \phi_G \ , \tag{4.15}$$

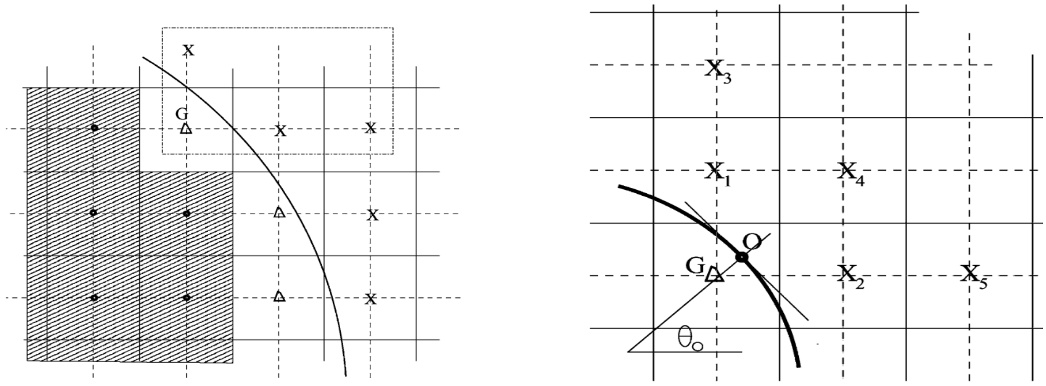
where the summation takes into account all of the points on the interpolation stencil, with at least one boundary point. The weighing coefficients ω_i are known and geometry dependent. [10, 6] The coefficients can be evaluated at each point initially and then stored for later use. [10] An improvement to the linear interpolation scheme would be the addition of a bilinear term $c_3 xy$ to equation (4.14) and with the use of node X_4 in the interpolation stencil. In 3D, this can be extended to be trilinear. [6]

Equation (4.14) can be used to reconstruct the velocity profile of a laminar flow or of high Reynolds number flow provided that the first grid point is located inside of the boundary layer. However, when the grid resolution is not enough to identify the boundary layer, a higher-order interpolation should be used. [6]

A Neumann boundary condition can be directly imposed as well. For pressure, the zero gradient condition at the boundary is described by [10]

$$\frac{\partial p}{\partial n} = \frac{\partial p}{\partial x}\mathbf{n}_x + \frac{\partial p}{\partial y}\mathbf{n}_y = 0 \ , \tag{4.16}$$

where $\mathbf{n}_x$ and $\mathbf{n}_y$ represent the components of the unit vector in the direction of the surface normal. Similar stencils as above can be used to calculate e.g. a linear or quadratic interpolation of the flow coefficient ϕ. These procedures require the determination of Θ_O, the slope of the surface normal at the boundary node O (see figure 4.5b).



(a) Determination of ghost-cells. (b) Determination of boundary point O.

Figure 4.5: Schematics of the ghost-cell IB method with solid cells (·), fluid cells (X) and ghost-cells (△). [10]

Using the ghost-cell method is problematic when dealing with boundary points that are located close to the fluid nodes used in the interpolation stencil. This manifests as large negative weighing coefficients that can lead to the overestimation of ϕ and numerical instability. The problem can be addressed by defining an image of the ghost node in the fluid domain and using this point to determine the flow variable. The value of at the ghost node is then retrieved from $\phi_G = 2\phi_O - \phi_I$, where the subscript O refers to the boundary node and I to the image node. [10] Another way to tackle the problem is to modify the boundary by a piecewise linear approximation so that the boundary point is located right at the closest fluid node. When this is necessary, the point is already located very close to the boundary, i.e. the normal distance of the boundary point to the fluid point is less 10% of the cell size, and thus the accuracy of solution is hardly affected. [10] A visualization of the two methods is presented in figure 4.6.

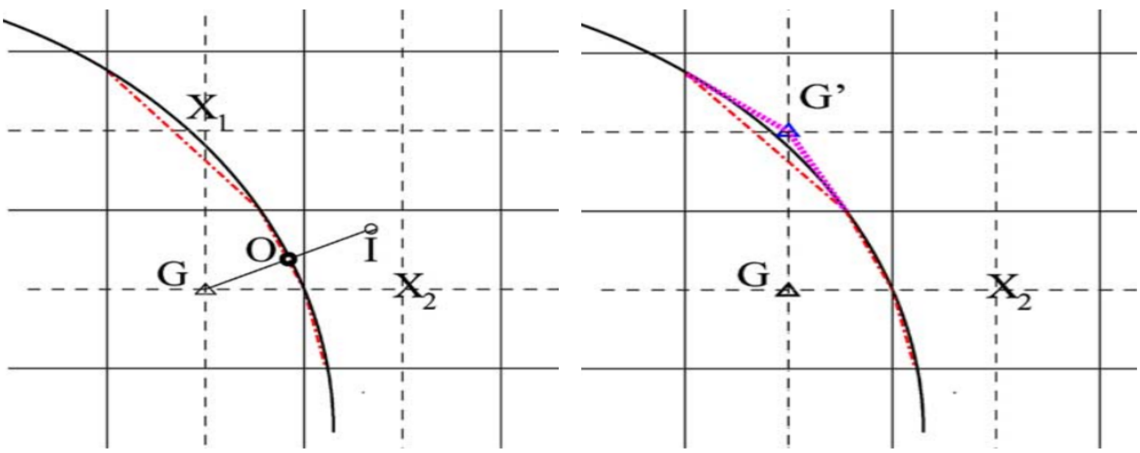


Figure 4.6: Methods for the treatment of boundary points located close to a fluid node. Left: ghost-cell imaging, where the ghost-cell is mirrored into the fluid domain as point I. Right: piecewise linear approximation of the boundary surface so that the boundary point is located at the fluid node G'. [10]

The ghost-cell method described above shows second-order accuracy for complex geometries. [10] However, due to the presence of a fictitious velocity field inside the immersed boundary, an unrealistic mass flux is generated over the IB. [46] The method was later improved by Kim et al. [43] with an addition of mass source and sink terms to correct the continuity equation at the intersected cells. Other authors improved the ghost-cell approach to account for e.g. moving bodies [47] and to preserve the local and global continuity over the IB without any mass source/sink terms. [48] The ghost-cell method has been used successfully to calculate e.g. compressible flow past an aerofoil [49] and turbulent flow past a flat plate. [50]

4.4 Cut-cell finite-volume approach

The cut-cell method was first proposed by Clarke et al. [51] for inviscid flows and later applied to viscous flow simulation by Ye et al. [52] This approach differs greatly from the earlier approaches, since it does not use momentum forcing and thus is not an IB method in the same sense as the continuous and discrete forcing methods.

The method consists of identifying cells that are intersected by the boundary similarly to the ghost-cell method. However, the cells are reshaped to accommodate the boundary instead of interpolating velocities. This method is also called the Cartesian grid method. [51] The boundary is treated as a piecewise linear surface inside of each intersected cell. The cut cells are categorized to those that have their cell centre in the fluid and those in the solid. [52] For the sake of clarity, let us call these cells external and internal boundary cells, respectively. The external boundary cells are reshaped by discarding the portions that lie in the solid, while the portions of the internal boundary cells that lie outside the solid are absorbed by neighbouring fluid cells. [52] This procedure can be seen in figure 4.7. As a consequence, methods similar to those when dealing with unstructured grids are required to evaluate mass, convective and diffusive fluxes as well as pressure gradients on the modified cell faces [52]. The fluxes that require special treatment can be seen in figure 4.7 as shaded arrows.

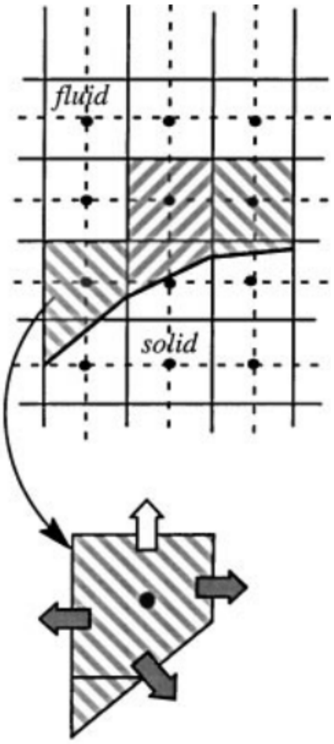


Figure 4.7: Reshaping procedure for boundary cells with arrows representing face fluxes. Shaded arrows denote fluxes that require special treatment. [52]

Downsides of the cut-cell methods include the increased complexity when calculating fluxes between diagonally adjacent cells and the formulation of the truncation process into a three-dimensional domain. [53] Although the authors report their scheme to be second-order accurate and satisfy the mass and momentum conservation equations, [52] it should be noted that they performed simulations only on Wannier flow, in which advection terms are not present. [53] Kirkpatrick et al. [53] developed the cut-cell method further to circumvent the problems associated with cell merging. Instead of merging cells, the truncated cell and its neighbouring cell are linked as a "master/slave" pair. Their method is essentially the same as the above described cut-cell method, with the exception that boundary nodes are relocated and interpolation factors, volumes and areas are calculated during the pre-processing stage. [53] The methods used for the modifications will not be discussed here, as they are not in the scope of this work. Instead, the interested reader is referred to the publication of Kirckpatrick et al. [53]

Cut-cell methods have been used to calculate e.g. microfluidic jets [54] and flows with moving boundaries such as flow-induced vibration of a cylinder [55] and free-falling objects [56].

5 The IB method in OpenFOAM

The method implemented by Jasak [9] into the Foam-Extend software [3] (later: "the software") is detailed below. Additionally, this chapter serves as a guide to the use of the IB method in Foam-Extend.

5.1 Numerical model

The applied IB method is the discrete forcing approach with direct boundary condition imposition according to the author. [9] However, the approach is much like the ghost-cell approach detailed in section 4.3.2.

The cells in the computational domain are divided into three categories based on their location with respect to the IB:

1. Fluid cells: cells that are far away from the boundary and completely in the fluid.
2. Solid cells: cells that have their centres inside of the boundary. The boundary may intersect the cells.
3. IB cells: cells that are intersected by the boundary and have their centres in the fluid.

As becomes evident from the IB cell determination procedure and figure 5.1, the employed approach can be considered to be a variation of the ghost-cell approach (see figure 4.5). The interpolation stencil that is used consists of the neighboring cells (including neighboring IB cells) and the boundary condition at the wall. The Dirichlet boundary condition for flow variable ϕ is evaluated from a quadratic polynomial fit [9]

$$\begin{aligned}\phi_P = \phi_{IB} + C_0(x_P - x_{IB}) + C_1(y_P - y_{IB}) + \\ + C_2(x_P - x_{IB})(y_P - y_{IB}) + C_3(x_P - x_{IB})^2 + C_4(y_P - y_{IB})^2 \ ,\end{aligned} \tag{5.1}$$

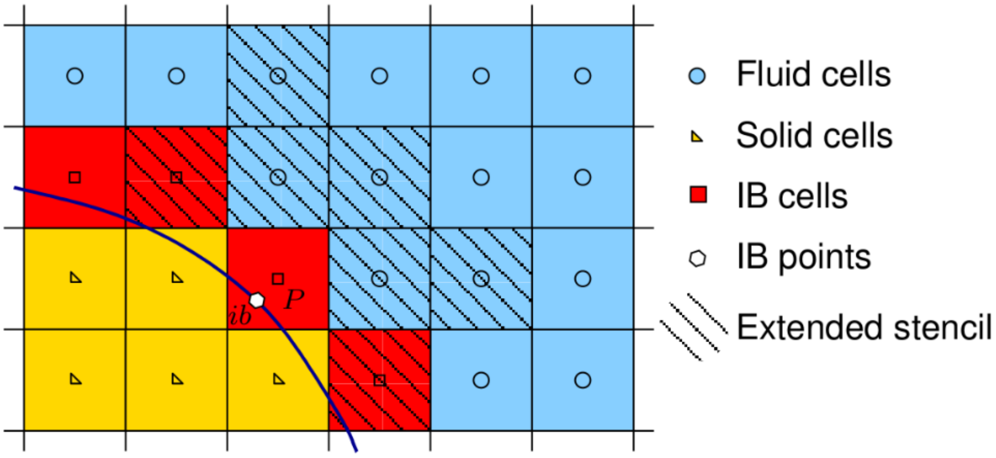
where x and y are the global coordinates and the subscripts P and IB correspond to the IB cell centre and the surface node ib, respectively. The Neumann boundary condition is calculated in the local coordinate system $x'y'$ (see figure 5.1b), where x' is the coordinate normal to the boundary surface at node ib and y' is the tangential coordinate. The value at the IB cell centre is retrieved with a quadratic polynomial fit [9]

$$\begin{aligned}\phi_P = C_0 + \left[\vec{n}_{IB} \cdot (\nabla \phi)_{IB}\right] x'_P + C_1 y'_P + C_2 x'_P y'_P + \\ + C_3 (x'_P)^2 + C_4 (y'_P)^2 \ .\end{aligned} \tag{5.2}$$

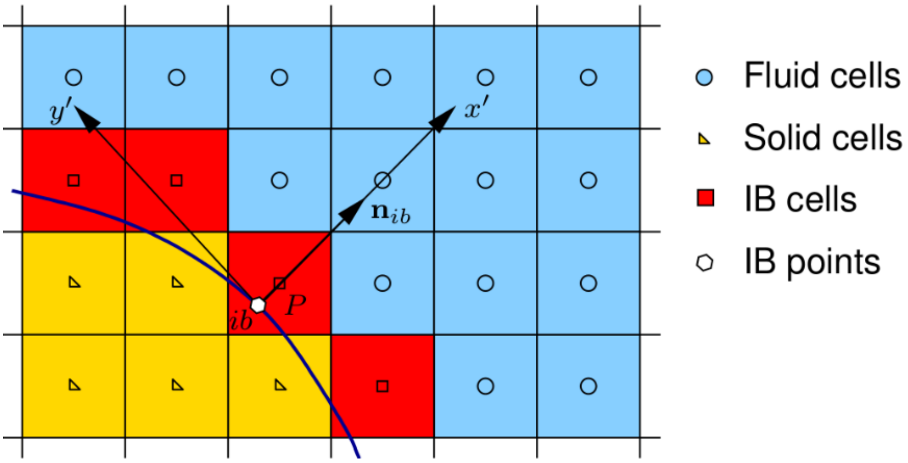
The coefficients C_i in equations (5.1) and (5.2) are determined from a weighted least square polynomial fit using the extended stencil seen in figure 5.1a. The following cosine weight function is considered:

$$w_i = \frac{1}{2}\left[1 + \cos\left(\pi \frac{r_i}{S r_{max}}\right)\right] \ . \tag{5.3}$$

The term r_i is the distance of the stencil node i from the IB cell centre P, $S = 1.1$ and r_{max} is the distance to the farthest stencil cell center.



(a) The interpolation stencil.



(b) Definition of the local coordinate system and boundary point (ib).

Figure 5.1: The schematics of the applied IB method.[9]

Pressure evaluation is not necessary in the IB cells as the velocity is retrieved by interpolating. However, the momentum equation requires the evaluation of pressure at the IB cell faces adjacent to fluid cells. The pressure equation (equation (3.6)) is modified to account for the presence of IB cells: [41]

$$\sum_f \left(\frac{1}{a_P}\right)_f \mathbf{n}_f \cdot (\boldsymbol{\nabla} p)_f S_f = \sum_f \mathbf{n}_f \cdot \left(\frac{\mathbf{H}_P}{a_P}\right)_f S_f + \sum_{f_{ib}} \mathbf{n}_{f_{ib}} \cdot \mathbf{v}_{f_{ib}} S_{f_{ib}} \ , \tag{5.4}$$

where the subscript f denotes the cell faces (f_{ib} if the face is between an IB cell and a fluid cell), $\mathbf{n}$ denotes the unit normal vector for the face f in question, S is the face area, $\mathbf{H}$ is the transport part of the Navier–Stokes equation, $\mathbf{v} = \frac{1}{2}\left(\mathbf{v}_P + \mathbf{v}_{N_{ib}}\right)$ is the velocity at the IB-face f_{ib} calculated as a central difference of the interpolated IB cell N_{ib} velocity $v_{N_{ib}}$ and fluid cell P velocity $\mathbf{v}_P$. The coefficient a_P is the diagonal contribution to the solution matrix and naturally dependent on the velocity field $\mathbf{U}$. The actual pressure at the IB faces and inside the IB cells is calculated after the solution of the modified pressure equation using the Neumann boundary condition of equation (5.2). The continuity equation is satisfied by scaling the interpolated IB face velocity $\mathbf{v}_{f_{ib}}$ so that zero net mass flux is imposed through a closed area around the boundary outlined by the dark green IB faces in figure 5.2.

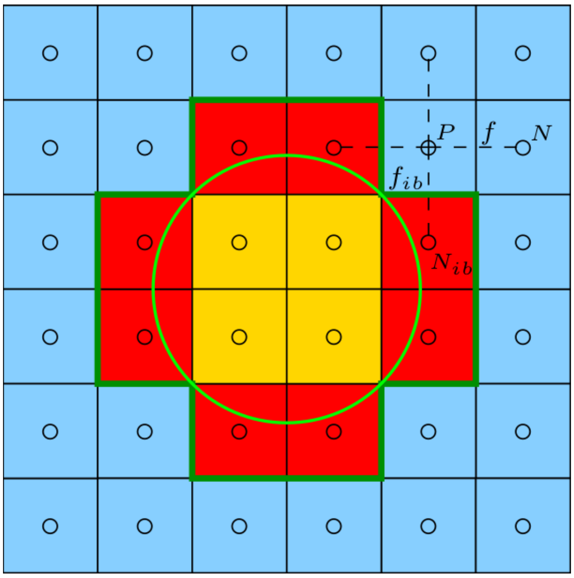


Figure 5.2: Schematics for the calculation of the modified pressure equation. [41]

5.2 Practical application

Next, we will focus on the practical application of the introduced IB method theory. The IB method has been implemented into the FOAM-Extend v. 4.0 library as three distinct classes:

1. `class immersedBoundaryPolyPatch`
2. `class immersedBoundaryFvPatch`
3. `class immersedBoundaryFvPatchField`

The first class generates the IB mesh overlay from an OpenFOAM triangulated format file (FTR) and handles its support over the computational mesh. The second identifies which cells and faces of the FV mesh shall be considered as IB cells and faces, thus providing "live" and "dead" (IB domain and rigid structure domain) cell and face indications. Additionally, it gives calculation capability of IB intersection points with their surface normals and distances, and the interpolation matrices (extended stencils for BCs shown in figure 5.1a). This includes the interpolation methods for IB values. Furthermore, the parallel processing framework is provided in the second class. The third class handles the patch field evaluation for the IB patch, the imposition of the boundary conditions (Dirichlet and Neumann), and the boundary updates. [41] To date, the method applied is capable of calculating flows with only one immersed boundary patch. [41]

5.3 User guide

The IB case setup consists of specific steps to incorporate the IB into the solution algorithm which are distinct from a BF method setup. The procedure will be outlined in this chapter.

The first step is the generation of an FTR file for the IB geometry. This can be done from a stereolithography (STL) output file extracted from e.g. OpenSCAD software. [57] The STL file can be transformed using the *surfaceConvert* utility, i.e. for an IB geometry stored in the file "ib.stl" the command

```
surfaceConvert ib.stl ib.ftr
```

will transform it into the FTR file. [1] The created FTR file has to be imported into the *case/constant/triSurface* folder.

The mesh can be generated in the normal fashion with the *blockMesh* utility. However, after mesh generation the *boundary* file inside the *case/constant/polyMesh* folder has to be modified to account for the IB. First, the number corresponding to the amount of patches detailed in *blockMeshDict* has to be updated to accommodate the addition of the immersed boundary patch. As stated earlier, the software is capable of calculating flows with only one immersed boundary patch, thus the number of patches is increased by one. In addition to the patch number update, the immersed boundary patch details must be added into the *boundary* file. The following string should be appended to the top of the list:

```
ib
{
    type            immersedBoundary;
    nFaces          0;
    startFace       4000;

    internalFlow    no;
}
```

The name of the patch must correspond to that of the FTR file, here "ib" for "ib.ftr". The *type* of patch is self-evident, it is the immersed boundary. The *nFaces* value is always 0, the *startFace* value must correspond to that value of the next patch. *internalFlow* value *no/yes* dictates whether the flow is considered to happen inside of the surface, i.e. 2D channel flow can be modelled with an immersed boundary patch corresponding to the fluid domain with *internalFlow* set to *yes*, or a patch corresponding to the solid domain with *internalFlow* set to *no*.

[1] Interestingly though, the author could not get this to work in the FOAM-Extend environment and had to use it in the OpenFOAM Foundation environment instead.

The next and final steps before the case can be run is the modification of the boundary conditions in the *case/0* folder, e.g. for p and U. The Dirichlet no-slip velocity boundary condition (left) and the Neumann boundary condition (right) can be imposed with the addition of

```
ib
{
    type                immersedBoundary;
    refValue            uniform (0 0 0);
    refGradient         uniform (0 0 0);
    fixesValue          yes;

    setDeadCellValue    yes;
    deadCellValue       (0 0 0);

    value               uniform (0 0 0);
}
```

```
ib
{
    type                immersedBoundary;
    refValue            uniform 0;
    refGradient         uniform 0;
    fixesValue          no;

    setDeadCellValue    yes;
    deadCellValue       0;

    value               uniform 0;
}
```

where the name "ib" has to correspond to that of the *boundary* file. The type is yet again self-explanatory. The value of *fixesValue* is *yes/no* and denotes the use of Dirichlet (*yes* for velocity) or Neumann (*no* for pressure) BCs at the immersed boundary cells. The *refValue* and the *refGradient* set the mass flux through the immersed boundary and are forced to zero to satisfy the continuity equation. The cells that are identified as "dead" are the solid cells, their value is set to zero with *setDeadCellValue* and *deadCellValue*.

The steps to incorporate a moving IB-patch are the same as those for the BF-patch after the above steps have been done. When the modifications are completed, the case is ready to be simulated. However, before executing the actual IB-solver, a potential flow solution should be initialized to help convergence. This can be done with the *potentialIbFoam*-solver. After the potential solution has converged a dedicated IB-solver such as *icoIbFoam* (stationary boundary) or *icoDyMIbFoam* (moving boundary) can be run to simulate the flow case.

6 Test cases

In this chapter, the validation cases are presented: 2D laminar channel flow, 3D laminar pipe flow, 2D cylinder in crossflow and 2D oscillating cylinder in cross flow. All of the test cases are run with body-fitted grids and immersed boundary grids to compare accuracy, robustness and computational demand. Furthermore, the results of the two different methods are compared to DNS and experimental results found in literature. The BF-simulations are done using OpenFOAM v. 5.0 [58] while the IB-simulations are done using the FOAM-Extend v. 4.0. [3]

The advantage of the IB method is that it is very quick to set up. However, the fast initialization is acquired with the cost of efficiency and accuracy. The BF grid outperformed the IB counterparts in each simulation case when considering accuracy and simulation time. Furthermore, the IB grid had large accuracy problems when simulating a moving boundary. Also, unresolved problems with parallel computing and cyclic boundary conditions were encountered.

6.1 Case 1: 2D laminar channel flow

We start the validation process by investigating 2D laminar channel flow. This is an attractive case, since the incompressible and laminar Newtonian fluid flow between two infinitely long plates, or plane Poiseuille flow, has an analytical solution that can be derived from the continuity equation and Navier–Stokes equations. Additionally, the case has low computational costs and different IB grid configurations can be benchmarked against each other for accuracy and computational time.

The problem consists of a duct of diameter $D = 2h = 4\,\mathrm{m}$ in the $\mathbf{y}$ direction and infinite in other directions ($\mathbf{x}$ and $\mathbf{z}$). A bulk flow ($\nu = 0.08\,\frac{\mathrm{m}^2}{\mathrm{s}}$) with a velocity $U_\infty = 1.0\,\frac{\mathrm{m}}{\mathrm{s}}$ is introduced at the inlet of the channel. The Reynolds number for the flow case depends on the kinematic viscosity ν, the bulk flow velocity U_∞ and the characteristic dimension L_{char}, so that $Re_{L_{char}} = \frac{U_\infty L_{char}}{\nu}$. The characteristic dimension of this flow case is the hydraulic diameter D_H, which, for a rectangular cross-section, can be determined from [59, p. 116]:

$$\frac{A}{P} = \frac{D_H}{4}, \tag{6.1}$$

where $A = WD$ is the cross-sectional area and $P = 2D + 2W$ is its perimeter. For a 2D-case, where $W >> D$, equation (6.1) reduces to $D_H = 2D$. The Reynolds number is then $Re_H = \frac{2DU_\infty}{\nu} = 100$, and the flow is clearly in the laminar regime ($Re_H < 2000$). [60]

This case will consist of the study of two distinct parameters to compare the accuracy of solutions. First, the shape and magnitude of the velocity profile for fully developed simulated flow shall correspond to that of the analytical solution derived in appendix A:

$$u(y) = \frac{3}{2}U_\infty\left(h^2 - y^2\right) \tag{6.2}$$

Second, inspection of the entrance length L_e will be conducted. By definition, it is the distance from the inlet at which the flow is fully developed. The flow shall be considered fully developed when it has attained 99% of its fully developed velocity. The corresponding empirical correlation is the following: [59, p. 107]

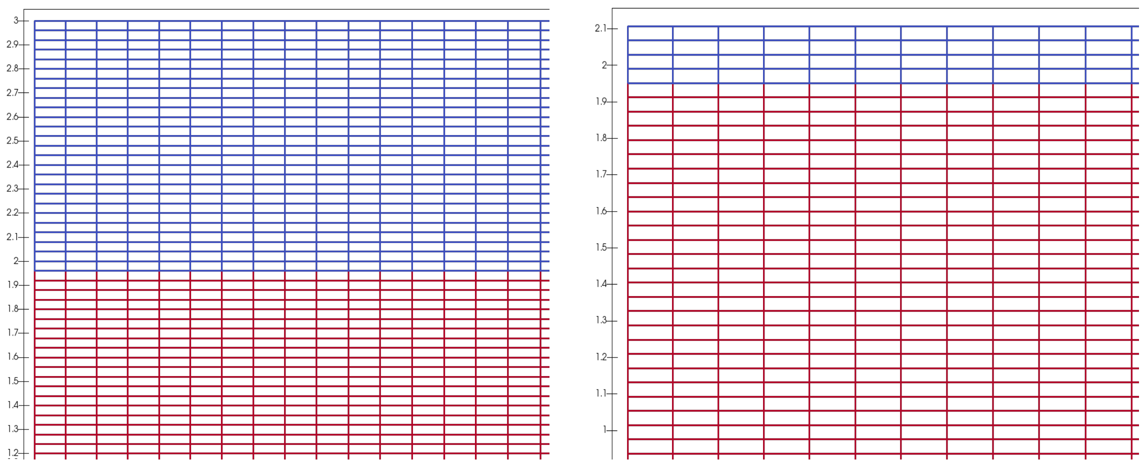
$$\frac{L_e}{D_h} \approx 0.5 + 0.05Re_H. \tag{6.3}$$

Solving equation (6.3) for the entrance length using the hydraulic diameter of a semi-infinite channel ($D_H = 2D$) and the Reynolds number $Re = 100$, gives us $L_e/D_H = 55$ or $L_e = 44\,\text{m}$. Thus, the grid length is chosen to be 60 m so that the developing region is fully encompassed inside of the domain. The determination of the entry region for simulated flow shall be done by comparing to the flow velocity at the outlet, i.e. the entrance length is the point where the velocity has achieved 99% of the outlet velocity.

A total of four grids were created for this case. Three of which were IB grids and one a body-fitted grid.

- Grid 1: conformal IB grid with thick walls. (Figure 6.1a)
- Grid 2: conformal IB grid with minimum thickness walls.
- Grid 3: nonconformal IB grid with thin walls. (Figure 6.1b)
- Grid 4: body-fitted grid.

The idea is to verify the robustness of the IB method in OpenFOAM by first comparing the computational costs of having extra cells that do not contribute to the solution accuracy inside of the immersed body (grids 1 and 2), an unfortunate consequence of the IB method. This is followed by determining the impact on accuracy of the IB boundary intersecting the cells at an arbitrary position (grids 2 and 3). Grid 4 serves as a reference solution for the conventional body-fitted method in the OpenFOAM environment.



(a) IB grid with thick walls

(b) Nonconformal IB grid

Figure 6.1: Different types of IB grids to test solver efficiency. Red denotes fluid cells, while blue denotes IB and solid cells.

The domain of the channel is 60 m × 4 m. Grids 1 and 2 are identical inside of the channel area, both having 100 cells in the **y**-direction and 300 cells in the **x**-direction. However, grid 1 has 1 m of wall on each side of the channel with 25 cells (50 cells all together) in the **y**-direction, whereas grid 2 only has 0.08 m of wall on each side of the channel consisting of 2 cells on each side (4 cells all together). Thus, the total number of cells are 45 000 and 31 200 for grids 1 and 2, respectively. Grid 3 is identical in the **x**-direction. In the **y**-direction, grid 3 consists of 108 cells, so that there are 100.6 cells inside of the channel area and the IB walls intersect the cells. The total width of the non-conformal grid 3 is 4.215 m. The total number of cells for grid 3 is 31 800.

The height of the first cell near the wall needs to be adequate to resolve the boundary layer and can be checked using the dimensionless wall distance y^+. To find out y^+ for the grid and case combination, we need to determine the wall shear stress. This can calculated analytically by derivating equation (6.2) and a value for y^+ can be obtained from equations

$$y^+ = \frac{U_\tau \Delta y_1}{\nu} \tag{6.4}$$

$$U_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{6.5}$$

where $\tau_w = \mu \frac{\mathrm{d}u}{\mathrm{d}y}|_{y=\pm h}$ is the wall shear stress and Δy_1 is the height of the first cell near the wall. Values for density ρ and dynamic viscosity μ are not required, since equation (6.4) reduces to:

$$y^+ = \frac{\Delta y_1}{\sqrt{\nu}} \frac{\mathrm{d}u}{\mathrm{d}y}\bigg|_{y=\pm h} \tag{6.6}$$

The dimensionless wall distance was determined to be sufficient, i.e. $y^+ < 1$ with the current case and grid combinations. Table 7.1 shows the specifications of the different grids.

Simulation was run until $T = 150$ s with a constant time step of $\Delta t = 0.02$ s to ensure that the CFL condition is satisfied for all grids.

6.2 Case 2: 3D laminar pipe flow

The validation process is continued with a 3D laminar pipe flow setup. The pipe diameter is $D = 1.0$ m and the kinematic viscosity of the fluid is $\nu = 0.025\,\frac{\mathrm{m}^2}{\mathrm{s}}$. The bulk flow velocity is $U_\infty = 1.0\,\frac{\mathrm{m}}{\mathrm{s}}$. The hydraulic diameter in this case is the pipe diameter. [59] This results in a Reynolds number of $Re = 40$, which is in the laminar flow regime. [60] The entrance length is approximated from equation (6.3) to be $L_e/D = 2.5$ m. The length of the domain was selected to be 6.0 m. Furthermore, the velocity profile of fully developed pipe flow is known to be [59, pp. 108-109]:

$$u(r) = 2U_\infty(1 - \left(\frac{r}{R}\right)^2) \tag{6.7}$$

where $R = 0.5$ m is the radius of the pipe and r is the radial coordinate.

For this case, two grids were constructed, one being an IB grid while the other a BF grid. The IB grid is a block with dimension $1.2\,\mathrm{m} \times 1.2\,\mathrm{m} \times 6\,\mathrm{m}$ with a cell count of 250 000 ($50 \times 50 \times 100$). The BF grid is a "butterfly" grid with a $0.4\,\mathrm{m} \times 0.4\,\mathrm{m} \times 6\,\mathrm{m}$ H-type core ($10 \times 10 \times 50$ cells) surrounded by four O-type blocks with 10 cells in the tangential direction, 20 cells in the radial direction and 50 cells in the axial direction. The total number of cells in the BF grid is 45 000. The outer dimensions of the BF grid naturally align with the case pipe. The used case and grid combinations give $y^+ < 1$. Table 7.2 shows the specifications of the two different grids.

The simulations were run until $t = 20\,\mathrm{s}$ so that a bit over 3 flow-through times were achieved (statistical convergence was achieved). A time step of $\Delta t = 4 \times 10^{-3}\mathrm{s}$ was used for both grids to keep the Courant number small. It should be noted that the BF grid could handle an order larger time step, thus drastically improving simulation time.

6.3 Case 3: 2D cylinder in crossflow

The next step in the validation process is to check the spatial accuracy of the IB method in steady and unsteady flow past a cylinder immersed in fluid. This case is a typical model problem used extensively in literature [2, 10, 29, 52, 61, 34].

The problem consists of a cylinder of diameter $D = 2R = 1\,\mathrm{m}$ immersed in an unbounded fluid which has a flow velocity of $U_\infty = 1.0\,\frac{\mathrm{m}}{\mathrm{s}}$. Two different cases are assessed: a steady-state case where the Reynolds number is $Re_D = 40$ and an unsteady case where the Reynolds number is $Re_D = 100$. The Reynolds number is controlled via the kinematic viscosity of the fluid, i.e. $Re = 40$ is given with $\nu = 0.025\,\frac{\mathrm{m}^2}{\mathrm{s}}$ and $Re = 100$ with $\nu = 0.01\,\frac{\mathrm{m}^2}{\mathrm{s}}$. At low Reynolds numbers ($Re < 40$), two standing vortices should form symmetrically around the centre axis while remaining attached to the cylinder. [29] When the Reynolds number is increased above a critical point ($Re > 46$), [52] the vortices detach and form an alternating pair of shed vortices, called the *von Kármán vortex street*. [29]

The crossflow over a cylinder in the regime of $1 \leq Re \leq 40$ is characterized by the formation of stagnant circulation bubbles behind the cylinder. [10] This low-Re flow is assessed by first determining the length of the recirculating wake vortices and the drag coefficient of the cylinder. These two values are then compared to experimental data found in literature. To determine the drag coefficient, we need to find the total drag generated by the cylinder. This can be done by interpolating the pressure distribution around the cylinder surface and then adding the viscous forces. Both of the OpenFOAM software have built-in functions called the *immersedBoundaryForces* and *forcesCoeffs* which can be adjusted to calculate these values for the IB method BF method respectively. The drag coefficient and total drag force are related by:

$$F_d = \frac{1}{2}\rho U_\infty^2 S C_d \tag{6.8}$$

where ρ is the density of the fluid and S is a reference area. The length of the recirculation zone is measured from the cylinder surface at the symmetry plane to the closure point of the steady vortices.

The unsteady von Kármán vortex street is characterized by the average drag coefficient, the root mean square of the induced oscillating lift coefficient and the Strouhal number

$$St = \frac{fD}{U_\infty} \tag{6.9}$$

where f is the frequency of the oscillations. The lift and drag coefficients can be determined similarly as with the low Reynolds number case.

The computational domain dimensions were chosen to correspond to that used by Uhlmann [28] with the cylinder located at the origin of a the domain $\Omega = [-6D, 20D] \times [-13D, 13D]$. One IB and one BF grid were generated for this flow case.

The IB grid is the same one used by Uhlmann and is a uniform grid with 1024×1024 nodes. The BF grid is a multiblock grid which is refined near the cylinder. Figures 6.2 and 6.3 describe the grids used for both steady and unsteady simulations. Overall, the IB grid has over 30 times as many cells as the BF grid.

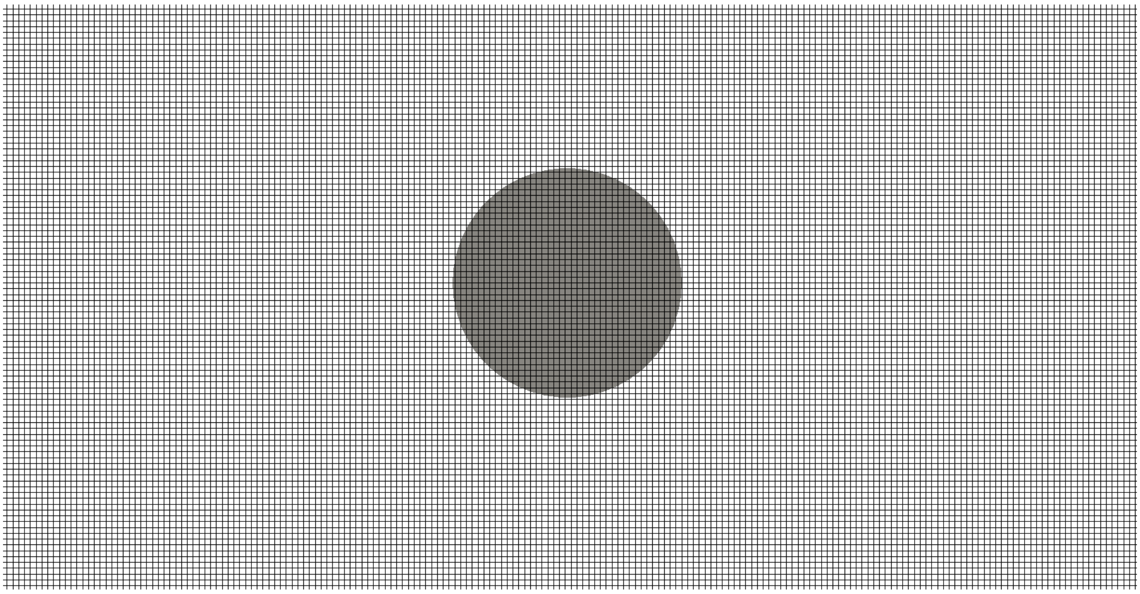


Figure 6.2: The uniform IB-grid used for the simulation of a cylinder in crossflow.

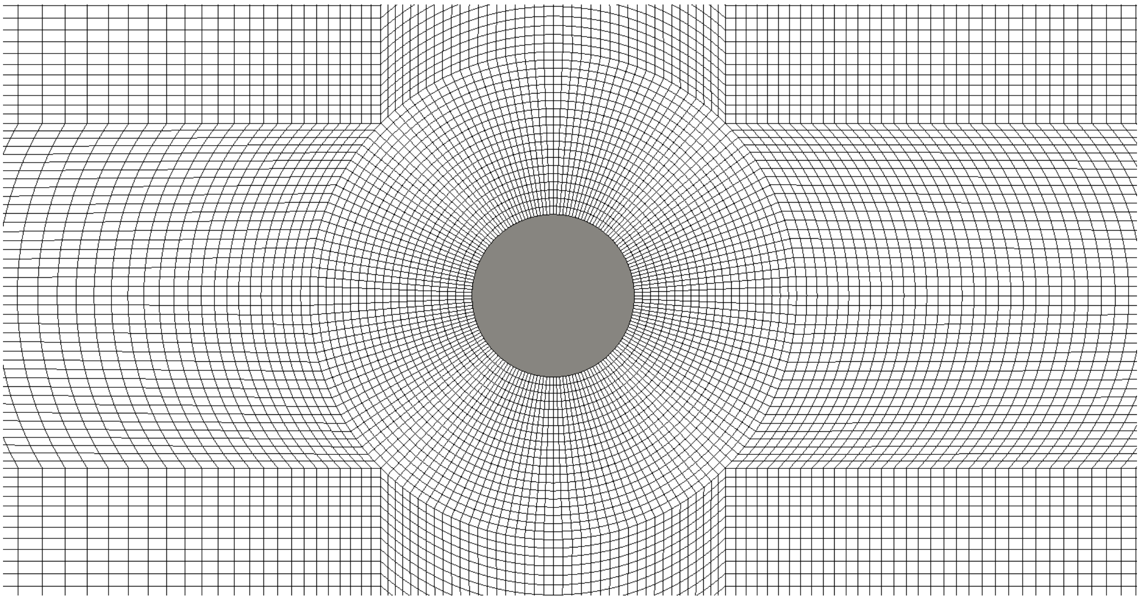


Figure 6.3: The multiblock BF-grid used for the simulation of a cylinder in crossflow.

y^+ values for the case and grid combinations are calculated using the following empirical correlation for friction coefficient:

$$C_f = 0.058Re^{-0.2}. \tag{6.10}$$

The wall shear stress is then estimated from

$$\tau_w = \frac{1}{2} C_f \rho U_\infty^2. \tag{6.11}$$

Values of $y^+ < 1$ are obtained for all flow cases.

Simulation time was selected to be 180 s so that the von Kármán vortex street had time to develop and stabilize. The time step used for the much more demanding IB grid was $\Delta t = 5 \times 10^{-3}$s. For the numerically lighter BF grid, a larger time step could have been possible. However, the same Δt was selected. This way, the Courant number was kept quite small for both cases.

6.4 Case 4: 2D oscillating cylinder in crossflow

The potential of the IB method cannot be fully harnessed in the first three cases. The cases are fairly simple to simulate with conventional BF techniques. One of the advantages of IB simulation is the adaption capability of the IB grid when dynamic movement is present. The next simulation case consists of a 2D cylinder oscillating transversely across fluid flow. It can be seen as an extension of Case 3 and used in variety of past publications studying IB methods. [28, 62, 63, 64, 65, 66, 67]

The case consists of a cylinder ($D = 1$ m) oscillating in fluid flow ($\nu = 0.004\,\frac{\mathrm{m}^2}{\mathrm{s}}$) for which the free-stream velocity is $U_\infty = 0.74\,\frac{\mathrm{m}}{\mathrm{s}}$. This setup gives a Reynolds number of $Re = 185$, which lies above the critical value for vortex shedding. The cylinder oscillations are sinusoidal in the transverse direction to the flow and characterized by:

$$d_y = A \sin(2\pi f t) \tag{6.12}$$

where $A = 0.2\,\mathrm{m}$ is the amplitude of the oscillations and $f = 0.116\,\mathrm{Hz}$ is the frequency of the oscillation corresponding to $0.8 f_s$. The values have been selected so that simulations can be compared to results from literature. [28, 65] The interesting parameter for this case is the variation of the computed lift and drag coefficients with respect to the position of the cylinder. These values are extracted to demonstrate the capabilities of the algorithm to handle motion.

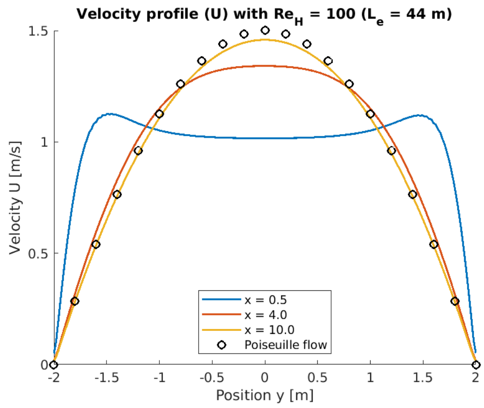
The computational domain dimensions are the same as in the previous Case and corresponds to that used by Uhlmann [28]. The IB grid is essentially the same as in the previous case. The cylinder oscillations are introduced to the overlying immersed boundary set and do not affect the underlying computational grid. The body-fitted grid is also similar to the one in case 3 with the exception that regridding is done between time steps to account for the cylinder motion. The BF case is solved using the dynamic mesh class provided in OpenFOAM. In particular, the *icoDyMFoam*-solver.

7 Results

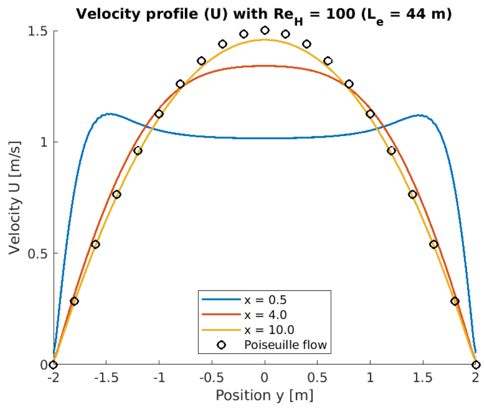
7.1 2D channel flow

2D simulation of channel flow with $Re_H = 100$ was done in case 1. Four different grids (3 IB and 1 BF) were generated and compared. The BF grid was used as a benchmark for OpenFOAM software performance. The Courant number stayed below 0.3 for all grids during simulation.

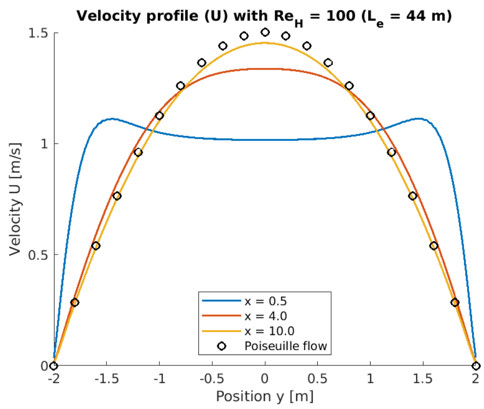
Figures 7.1a-7.1d show the cross-channel velocity profiles at different distances from the inlet for grids 1-4 respectively. It should be noted that the shown profiles are plotted with a maximum distance of $x = 10.0\,\mathrm{m}$. This is far less than the predicted entry length of 44 m. However, as can be seen from figure 7.2, the simulated entry length is much shorter for all grids ($\approx 9.5\,\mathrm{m}$).



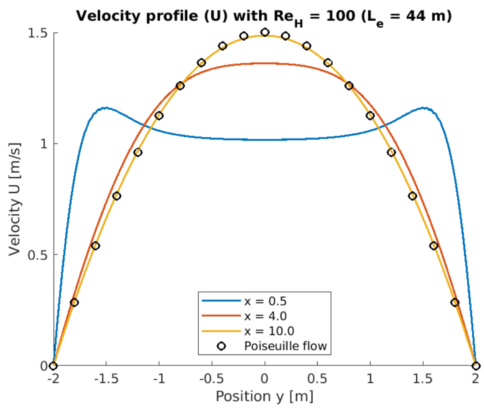
(a) Thick-walled IB grid.



(b) Thin-walled IB grid.



(c) Non-conformal IB grid



(d) BF grid.

Figure 7.1: Cross-sectional velocity profiles along **y**-axis at different distances from the entrance.

Figures 7.3 show the fully developed profiles achieved with different grids. The body-fitted grid velocity profile converges to the analytical Poiseuille solution as can be expected. However, the IB-grids do not. Figure 7.3b shows a peak velocity underestimate of around 1.9% for grids 1 and 2, which have converged to the same solution, and an underestimate of around 2.3% for the non-conformal grid 3. On the other hand the IB-grids overestimate the velocity behavior at the near-wall area (see 7.3c).

Table 7.1 shows the comparison of grid cell counts and simulation for different IB grids and the BF grid. Furthermore, it also contains data on the predictions of entry length and fully developed maximum velocity. The simulation time was determined to be the point where the solution had converged to the grids final solution ($U(T = 150\,\mathrm{s})$) with 99.99 % accuracy. This corresponded to a time of $T = 23.9\,\mathrm{s}$ for all grids. Clearly, the IB-grids cause much longer simulation times than the BF grid. However, this is only partly due to the increased cell count, since grid 2 with only a 4 % increase in cell amount compared to the BF grid, behaves in a way that the simulation time increases over 17-fold. The change in simulation time does not correlate linearly with the change in amount of computation cells, as can be seen from the slight time increase when switching to grid 1 with an almost 45 % increase in cell amount. Surprisingly, the non-conformal IB-grid (grid 3) is the most efficient IB-grid. This difference comes from the convergence speed of the different grids while the flow is still developing: the first two grids with larger simulation times tend to require more iterations (30-40 iterations) to achieve sufficient convergence when comparing to the non-conformal grid (10-20 iterations). This is probably due to the interpolation schemes in the IB-method helping the convergence of the solver near the wall.

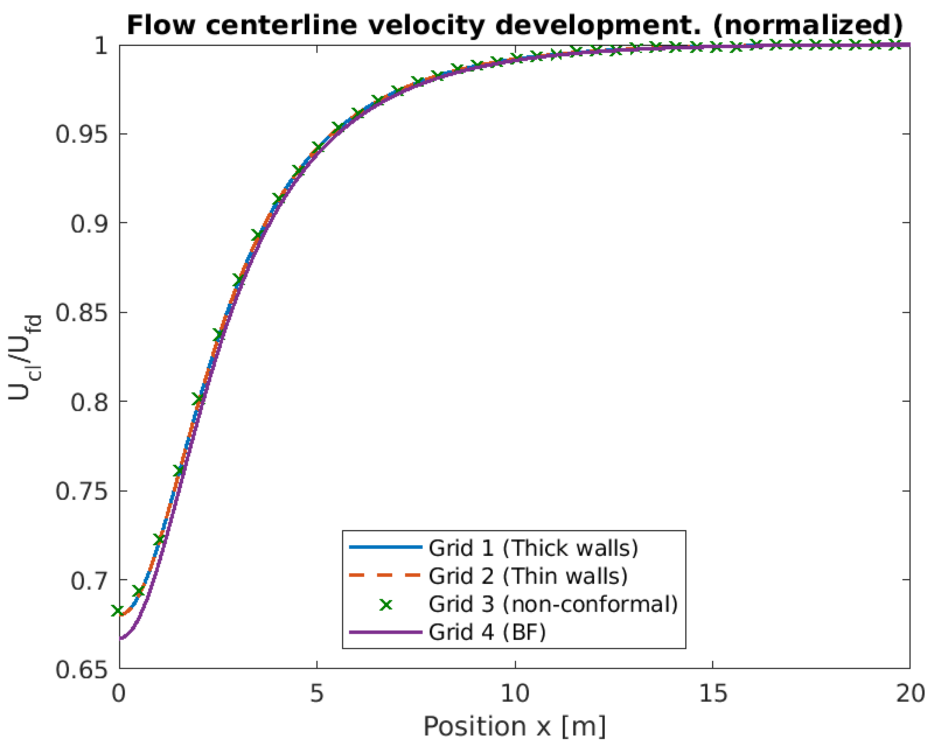
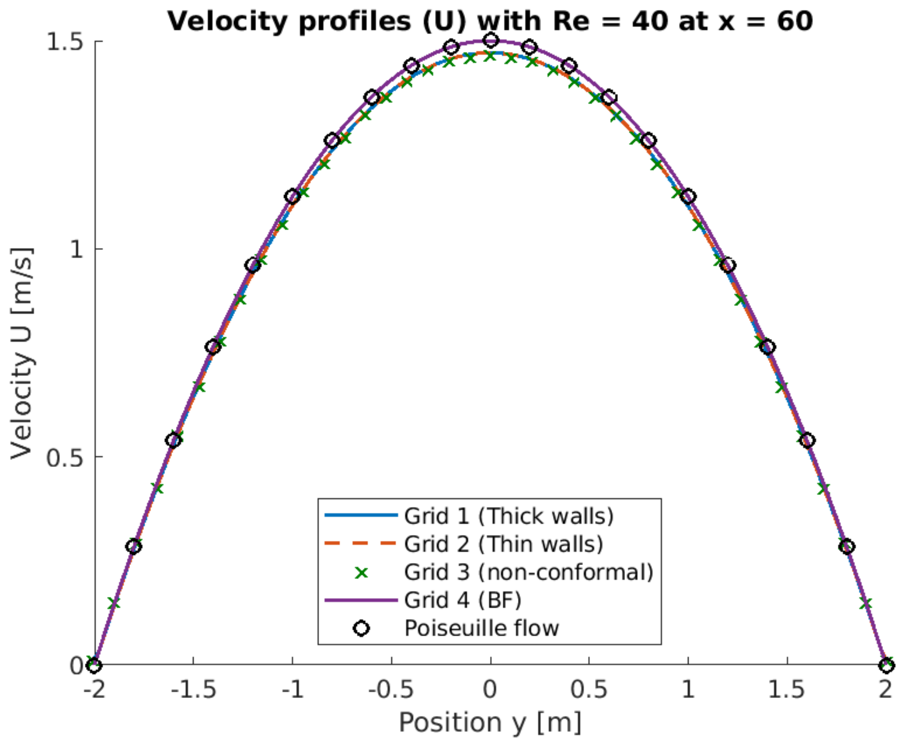
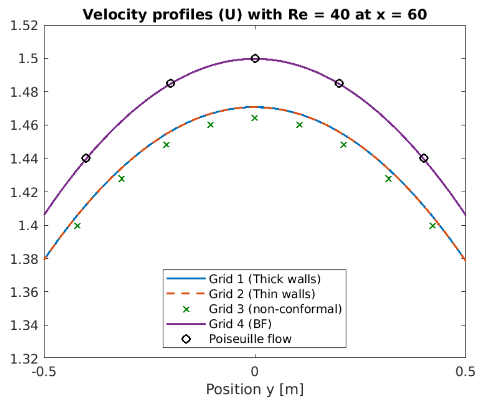


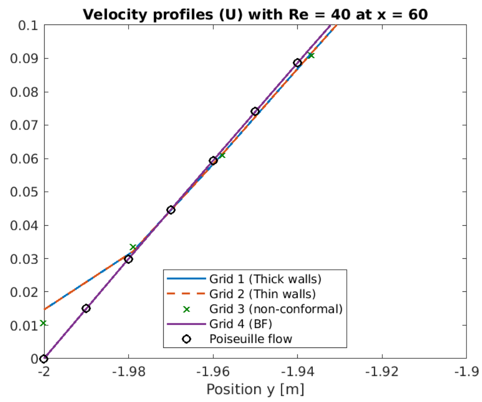
Figure 7.2: Centreline velocity development normalized with respect to outlet centreline velocity.



(a) Cross-channel profile.



(b) Peak values.



(c) Near wall area.

Figure 7.3: Velocity profile comparison of fully developed flow.

As can be seen from the results in table 7.1, the entrance length for a semi-infinite channel does not correlate with the empirical formula. The writer suspects that this is due to uncertainties related to the definition of the hydraulic diameter and 2D nature. Hence, 3D laminar pipe flow cases were constructed for IB and BF grids to validate the accuracy.

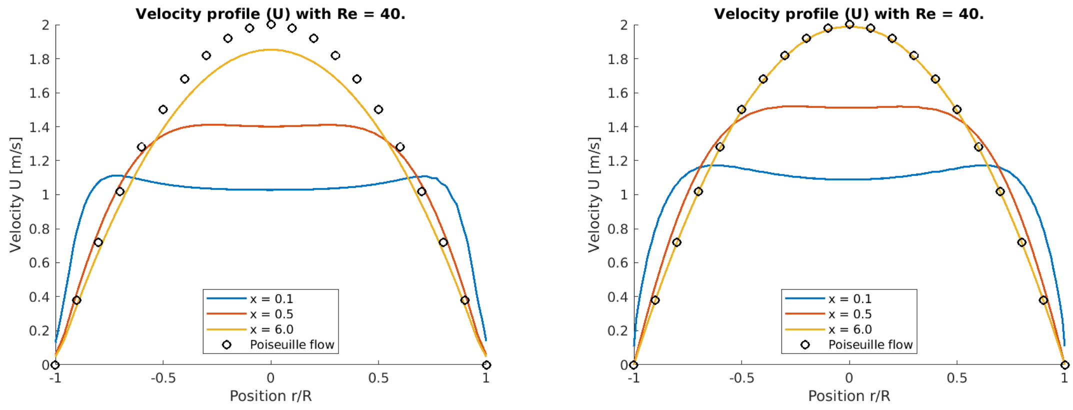
Table 7.1: 2D Channel flow grid size and simulation time comparisons for different IB and BF grids.

	IB			BF
	Grid 1	Grid 2	Grid 3	Grid 4
Total cells	45 000	31 200	32 400	30 000
Percentage	138.9	96.3	100	92.6
Simulation time [s]	1327	1292	757	69
Percentage	175.3	170.7	100	9.1
U_{max}/U_{∞}	1.471	1.471	1.465	1.500
L_e/D_H	1.184	1.184	1.178	1.212

7.2 3D pipe flow

The second simulation case consisted of 3D pipe flow with $Re = 40$ for which one IB and one BF grid were constructed. For both grids the Courant number stayed small, i.e. $C < 0.5$, throughout the simulations. The IB method forced the grid to have a high cell density, with an over 5-fold increase in cell amount compared to the Bf-grid. However, due to the nature of the IB method and the way the grid was generated (as a block), only around 54 % of these cells were inside the pipe volume. At around $T = 3.4\,\mathrm{s}$ the results on both grids had converged to 99.99 % of the final value at $T = 20\,\mathrm{s}$. The IB-simulation took 11 times longer than the BF-simulation to reach this value.

Similar results as in case 1 have been generated. Figures 7.4a and 7.4b show the velocity profiles at different distances from the entrance, figure 7.5 describes the centreline velocity development and figure 7.6 presents comparisons of the simulated fully developed flows (taken at $x = 6\,\mathrm{m}$). Although the parabolic nature of the velocity profile predicted by the IB method is in agreement with literature as can be seen from figure 7.4a, it underestimates the maximum velocity of the fully developed flow by 7.5 %. The velocity profile of the BF grid once again has excellent correspondence with the analytical solution with respect to both shape and magnitude. Both of the methods predict around an equal entry length as can be seen from figure 7.5. Entry length values of $\frac{L_e}{D}(IB) = 2.276$ and $\frac{L_e}{D}(BF) = 2.400$ have been retrieved and they correspond to an underestimate of 9 % and 4 % with respect to the empirical value $\frac{L_e}{D}(Re = 40) = 2.5$ for IB and BF grids, respectively. Similarly to the previous 2D case, the near-wall velocities are overpredicted with the IB method (see figure 7.6c). Results and simulation statistics have been gathered into table 7.2.



(a) IB grid.

(b) BF grid.

Figure 7.4: Cross-sectional velocity profiles along **r**-axis at different distances from the entrance for the 3D grids.

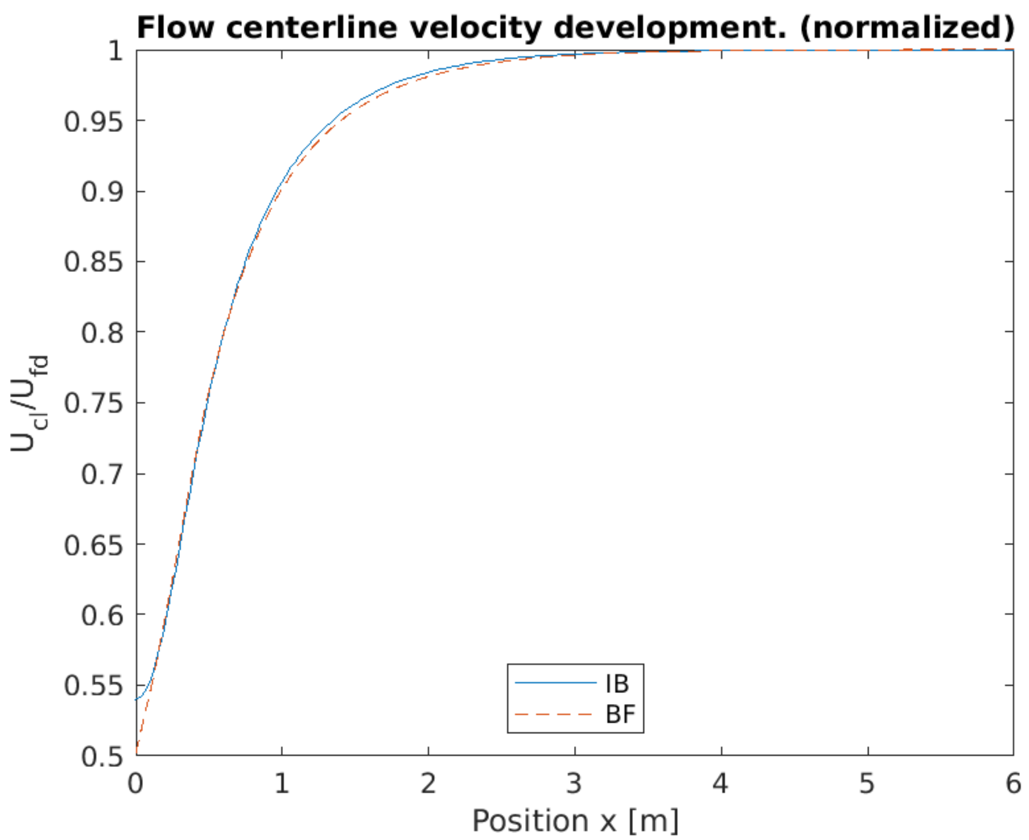
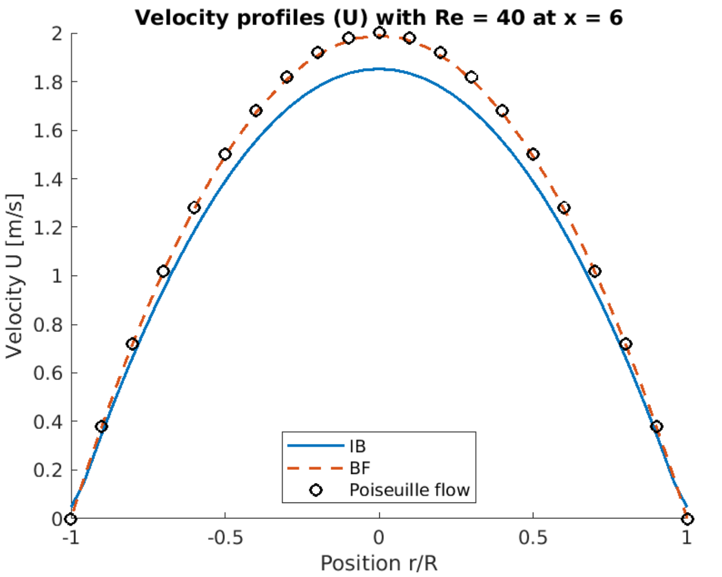
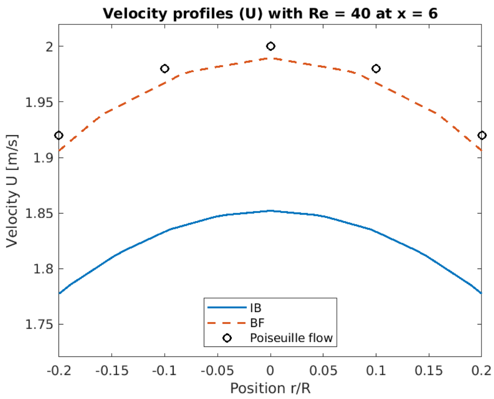


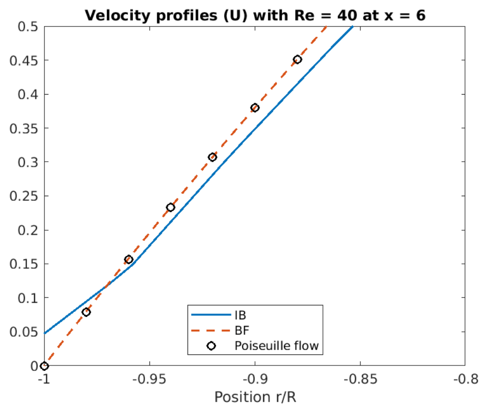
Figure 7.5: 3D simulation centerline velocity development normalized with respect to outlet centreline velocity.



(a) Profile



(b) Peak values



(c) Near wall area

Figure 7.6: 3D simulation velocity profile comparison of fully developed flow.

Table 7.2: Results and simulation properties for 3D pipe flow.

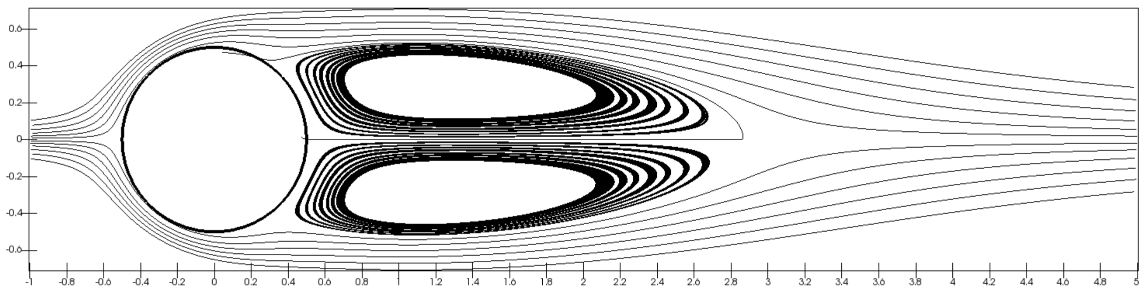
	IB	BF
Total cells	250 000	45 000
Percentage	555.6	100.0
Cells inside pipe	135 000	45 000
Percentage	300	100
Simulation time [s]	2066	178
Percentage	1160.7	100
U_{max}/U_∞	1.852	1.991
L_e/D	2.276	2.400

7.3 Cylinder in crossflow

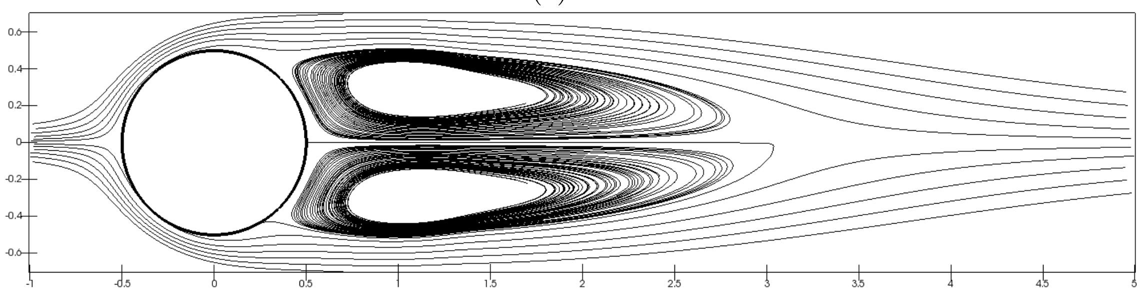
The third case was the steady ($Re = 40$) and unsteady ($Re = 100$) simulation of a cylinder in crossflow. Altogether, one IB grid and one BF grid were generated and used for both Reynolds number flows. The dimensionless wall distance y^+ values during steady state simulation were calculated to be 0.12 and 0.24 for the IB and BF grids, respectively. y^+ values for the unsteady state were estimated to be 0.28 (IB) and 0.54 (BF) using equations (6.10) and (6.11).

7.3.1 Steady state, Re = 40

The circulation bubble length L_w, which can be seen in figures 7.7a and 7.7b showing the streamlines for the case, was determined by evaluating the horizontal velocity component on the centreline axis ($\mathbf{y} = 0$). The length L_w corresponds to the point where no more back flow occurs, i.e. $U_x \geq 0$. The calculated circulation lengths were 2.374 and 2.550 for the IB and BF grid respectively, as can be seen in figure 7.8.



(a) IB



(b) BF

Figure 7.7: Steady state streamlines at for $Re = 40$.

Retrieving the pressure distribution along the cylinder wall is problematic when using the IB method as can be seen from figure 7.9 (yellow line). This behaviour is expected when having finite sized IB cells intersected by a curved surface and using interpolation schemes to retrieve values of pressure. However, when plotting the pressure coefficient along a circular surface of a 10% larger radius ($r = 0.55\,\mathrm{m}$), the calculated pressure coefficient C_p corresponds better to the results found in literature (see figure 7.9). The IB method overestimates the pressure coefficient on the first 45°, after which it is well in line with the reference values. On the contrary, the BF method resolves the pressure coefficient more accurately in the first 45° but

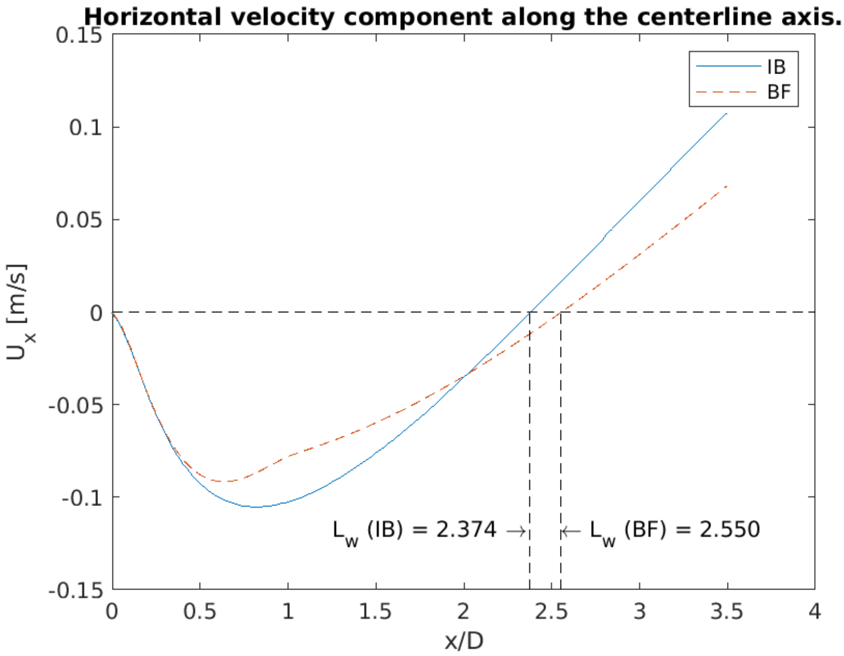


Figure 7.8: Horizontal centreline velocity component behind the cylinder.

underestimates it for the remaining 135°. The friction coefficient is a bit trickier for the immersed boundary method. As with the pressure, near-wall velocity is not resolved very well in the retained results. Manual transformation of the Cartesian velocity vector to cylindrical coordinates was done for $r = 0.55\,\mathrm{m}$. As can be expected with the above method the values severely underestimate, due to the much larger velocity gradients near the wall than at $r = 0.55\,\mathrm{m}$. However, a qualitative analysis of figure 7.10 suggests that the results are in good agreement, i.e. the shape of the friction coefficient distribution is in accordance with that of the BF method. The circulation zone lengths and the drag coefficients calculated by software are gathered in table 7.3 with results from the unsteady case. Finally, figures 7.11a and 7.11b show the vorticity contours for both grids. Judging by the wiggles in the vorticity contours for the BF method, the generated grid could be improved for better accuracy.

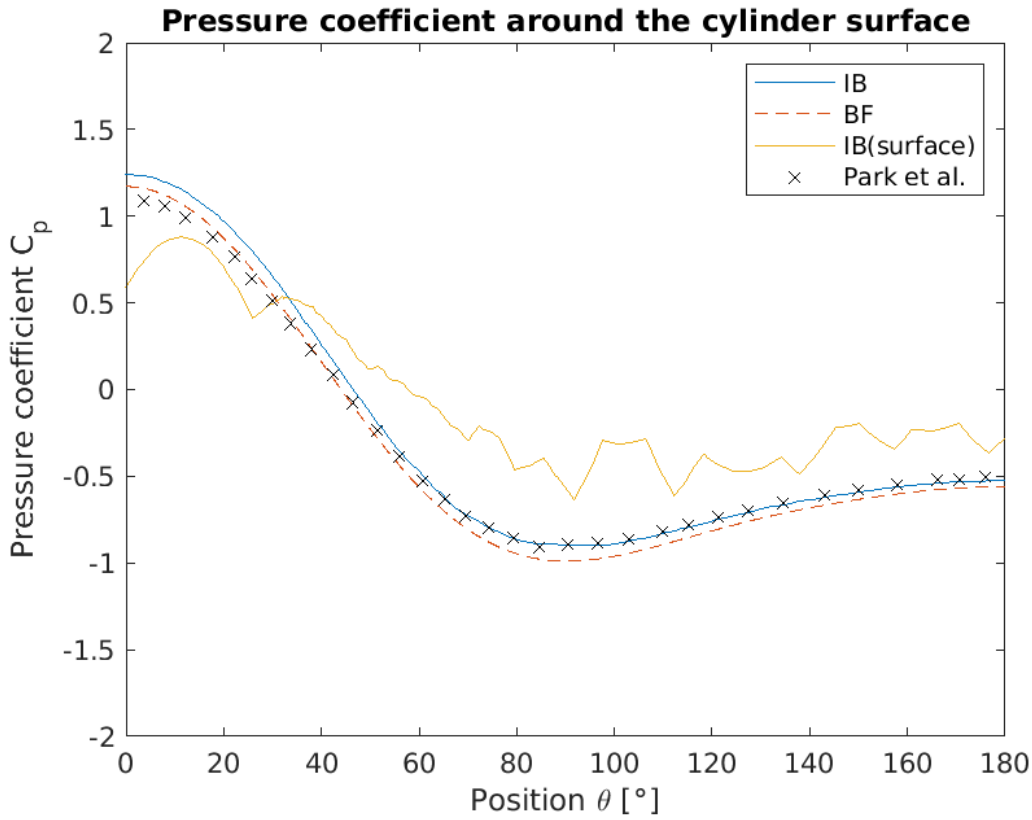


Figure 7.9: Pressure coefficient from the front stagnation point to the aft.

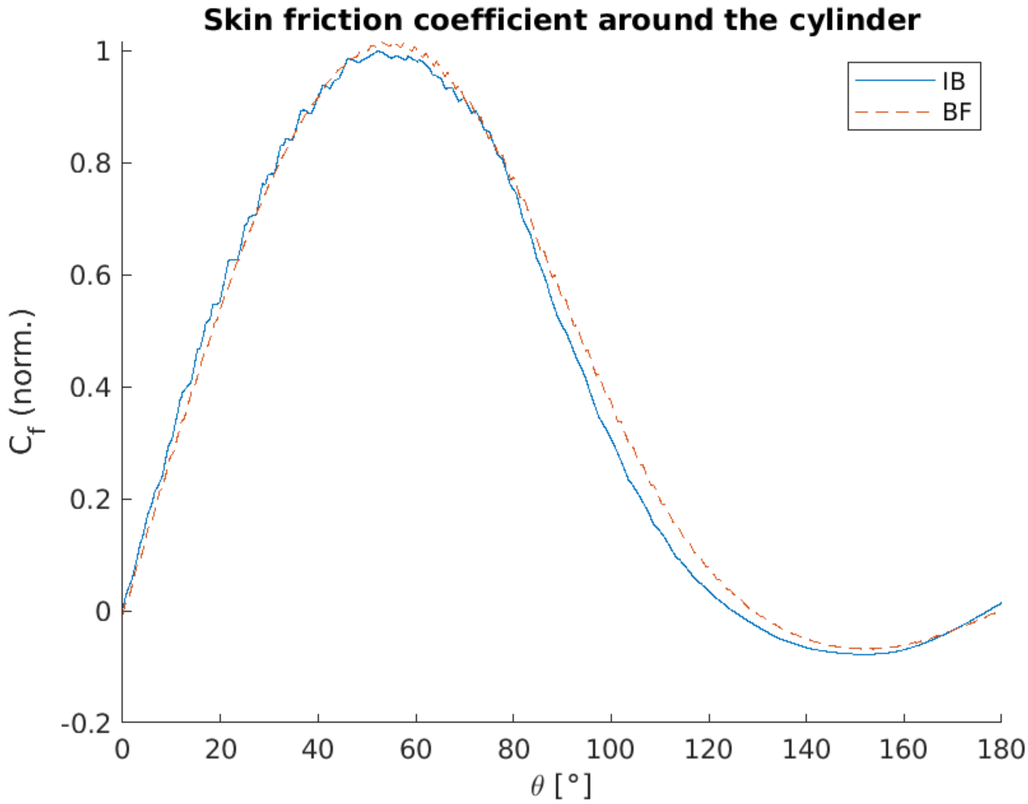
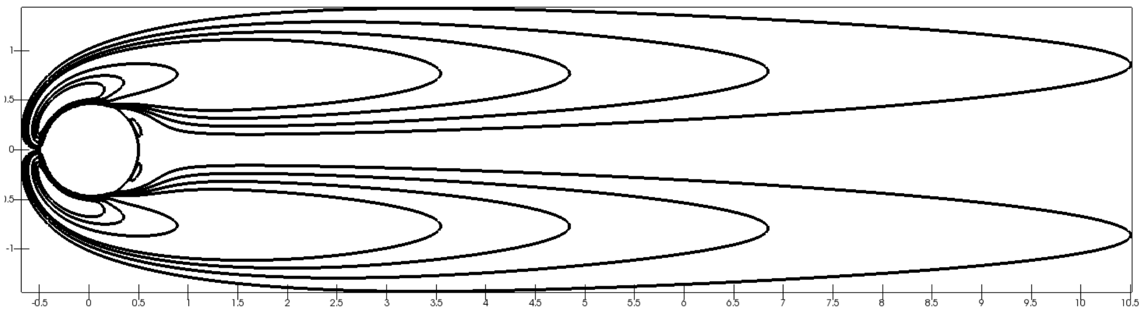
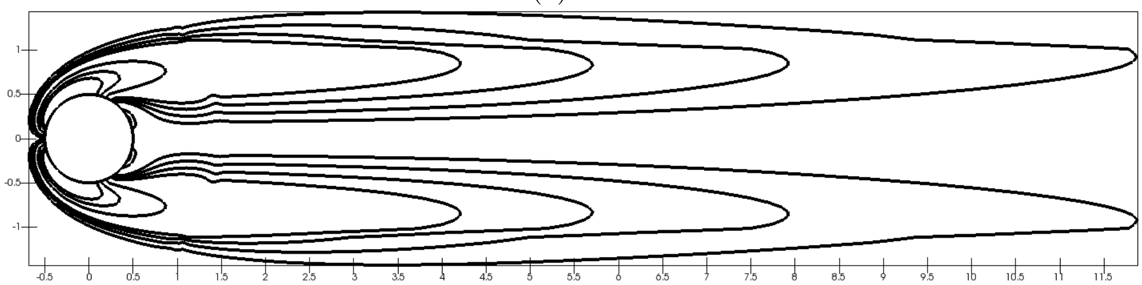


Figure 7.10: Normalized friction coefficient from the front stagnation point to the aft.



(a) IB

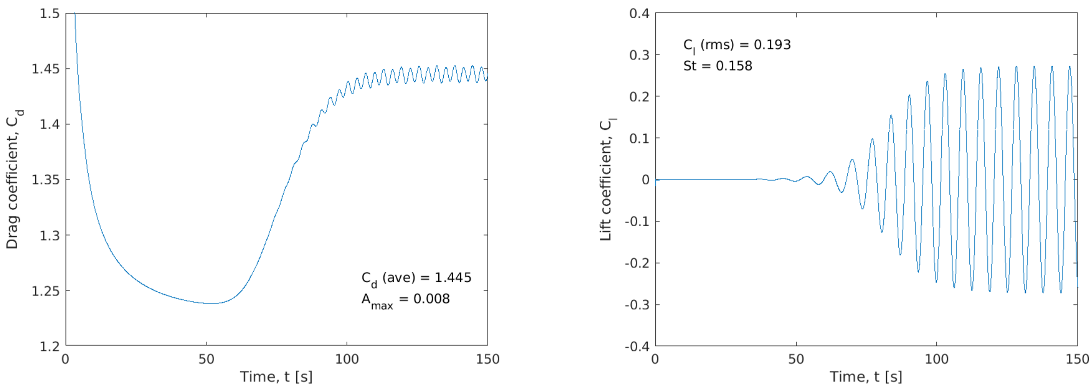


(b) BF

Figure 7.11: Steady state vorticity contours for $Re = 40$.

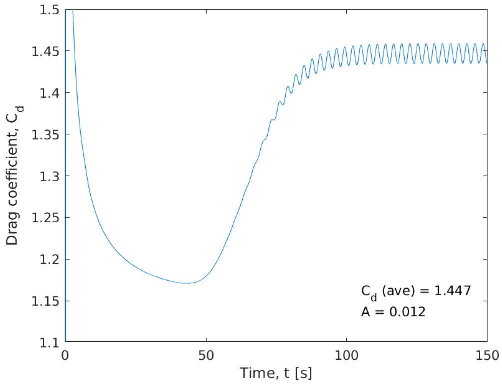
7.3.2 Unsteady state, Re = 100

Next, we discuss the simulation results of the unsteady flow over a stationary cylinder. Figures 7.12 and 7.13 show the time-dependent lift- and drag-coefficients (C_l and C_d, respectively) for both boundary methods. Instability starts at around T =40 s for both methods, after which harmonic oscillations are observed. The wavelength of the oscillations is $\lambda_{s_{IB}}$ =6.33 s and $\lambda_{s_{BF}}$ =5.780 s corresponding to the shedding frequency of $f_{s_{IB}}$ =0.158 Hz and $f_{s_{BF}}$ =0.173 Hz, for the IB and BF methods respectively. The shedding frequency is related to the Strouhal number by equation (6.9) to give $St_{IB} = 0.158$ and $St_{BF} = 0.173$. Analysis of the C_d plots yield average drag coefficients of $C_{d_{IB}}(ave) = 1.445$ and $C_{d_{BF}}(ave) = 1.447$. The oscillations of the drag coefficient with the IB method look peculiar. Conducting a fast Fourier transform (FFT) on the time signal of the drag coefficient confirms two shedding frequencies corresponding to $f_{s_{IB}}$ =0.158 Hz and $f_{s_2} = 2f_{s_{IB}}$ =0.316 Hz as can be seen in figure 7.15. The maximum amplitude of the drag coefficient for the IB method during one period of oscillation was $A_{C_d}(IB) = 0.008$ which consisted of two amplitudes of $A_{f_s} = 0.001$ and $A_{2f_s} = 0.007$. Only one amplitude of $A_{C_d}(BF) = 0.012$ was observed for the BF grid. Likewise, the root mean squares of the lift coefficient $C_{l_{IB}}(rms) = 0.193$ and $C_{l_{BF}}(rms) = 0.279$ are determined from the C_l plot. The lift coefficient did not show similar behaviour as the drag coefficient. Also, instantaneous vorticity contours for the same phase of oscillation were generated and can be seen in figures 7.14a and 7.14b. From here it becomes apparent that the BF simulation has inconsistencies due to grid refinement, which might explain some of the errors in the results when compared to literature. The comparison of all results can be found in table 7.3.

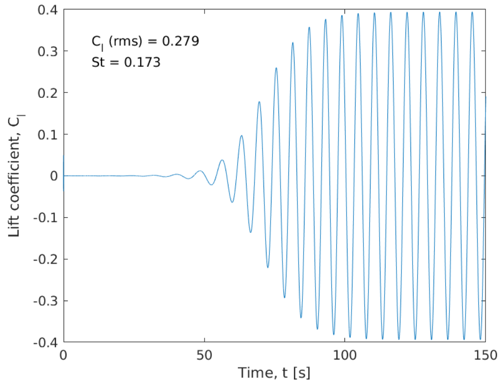


(a) Drag coefficient (b) Lift coefficient

Figure 7.12: Evolution of the aerodynamic coefficients for the IB method.

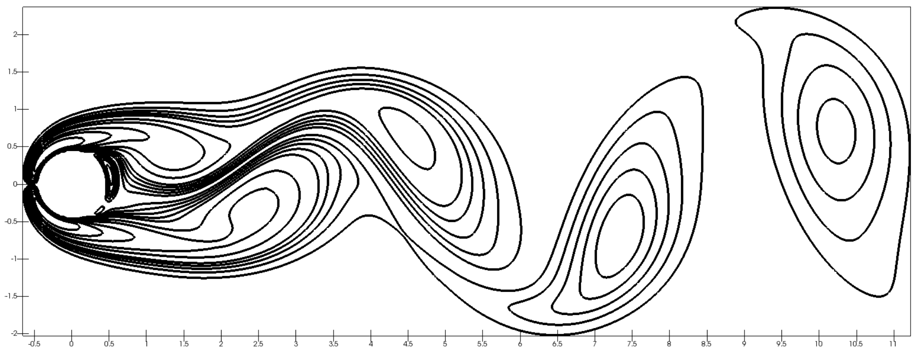


(a) Drag coefficient

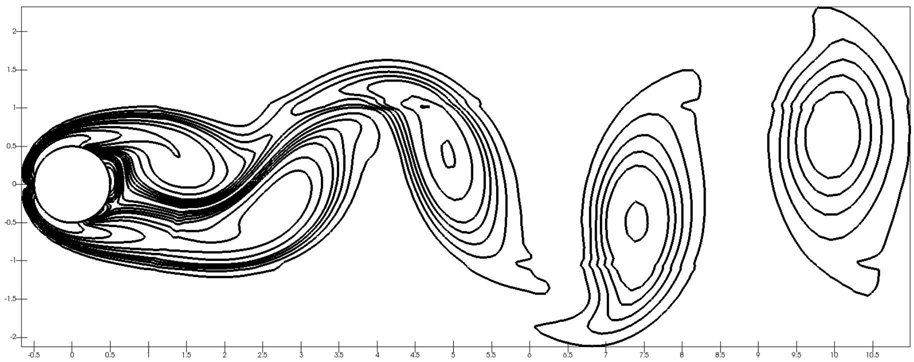


(b) Lift coefficient

Figure 7.13: Evolution of the aerodynamic coefficients for the BF method.



(a) IB



(b) BF

Figure 7.14: Vorticity contours for Re = 100 flow past a circular cylinder.

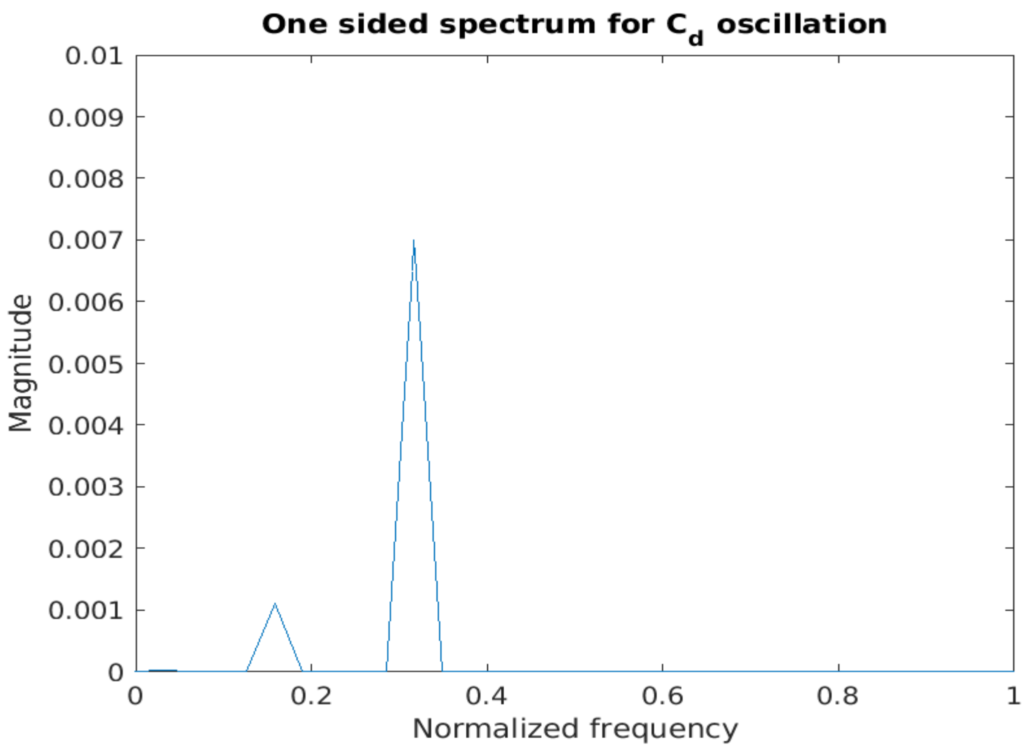


Figure 7.15: The frequency spectrum of the drag coefficient during one oscillation period with the IB method.

Table 7.3: Comparison of results for a cylinder in crossflow.

	Re = 40		**Re = 100**			
	L_w/D	C_d	$C_d(ave)$	A_{C_d}	$C_l(rms)$	St
Present IB	2.374	1.700	1.445	0.008	0.193	0.158
Present BF	2.550	1.608	1.447	0.0122	0.279	0.173
Park et al. [68] (BF)	2.27	1.51	1.33	0.0091	-	0.164
Constant et al. [2]	-	-	1.37	0.013	0.24	0.165
Tseng and Ferziger [10]	2.21	1.53	0.164	-	0.29	0.164
Lai and Peskin [29]	-	-	1.4473	-	0.3299	0.165
Ye et al. [52]	2.27	1.52	-	-	-	-
Lima et al. [61]	2.535	1.54	1.39	-	-	-
Saiki and Biringen [31]	-	-	1.26	-	-	0.171

7.4 Oscillating cylinder in crossflow

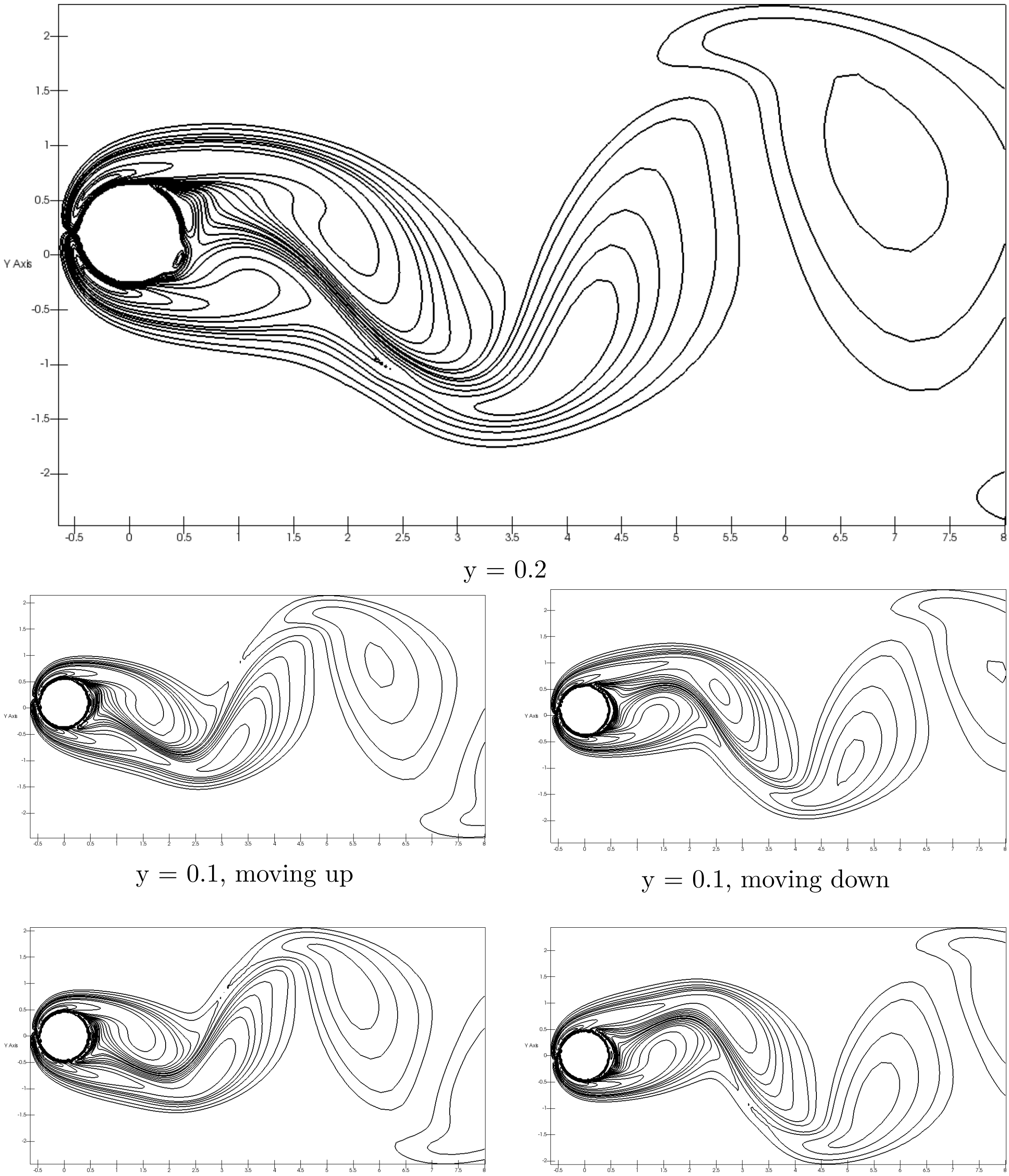
The final case was the oscillating cylinder in crossflow. The dimensionless wall distances were estimated to be 0.50 and 0.94 for the IB and BF grids, respectively.

Figure 7.16 shows the vorticity contours for the top half of an oscillation cycle with the IB grid. Qualitatively the results look decent and similar to those obtained by Constant et al. [2] However, when inspecting the time-periodic variation of the drag and lift coefficients in figures 7.17, the presence of strong wiggles may be noted. Despite the wiggles, the shapes do seem to correspond to those found in literature [28, 65], and the average drag coefficient ($C_d(ave) = 1.436$) is in the correct ballpark with an error of around 10-15%. However, the lift coefficient is greatly overestimated with a root mean square $C_l(rms) = 0.190$. The body-fitted method can resolve the coefficients smoothly with correct shape and values ($C_d(ave) = 1.300$ and $C_l(rms) = 0.081$). The results for the IB and BF grids have been gathered in table 7.4 with values found in literature.

Table 7.4: Comparison of results for an oscillating cylinder in crossflow.

	$C_d(ave)$	$C_l(rms)$
Present IB	1.436	1.90
Present BF	1.300	0.081
Uhlmann [28]	1.354	0.166
Yang et al. [65]	1.273-1.281	0.074-0.083
Guilmineau and Queutey [69] [2] (BF)	1.195	0.08

[2] These values were digitized by Yang et al. [65]



y = 0.2

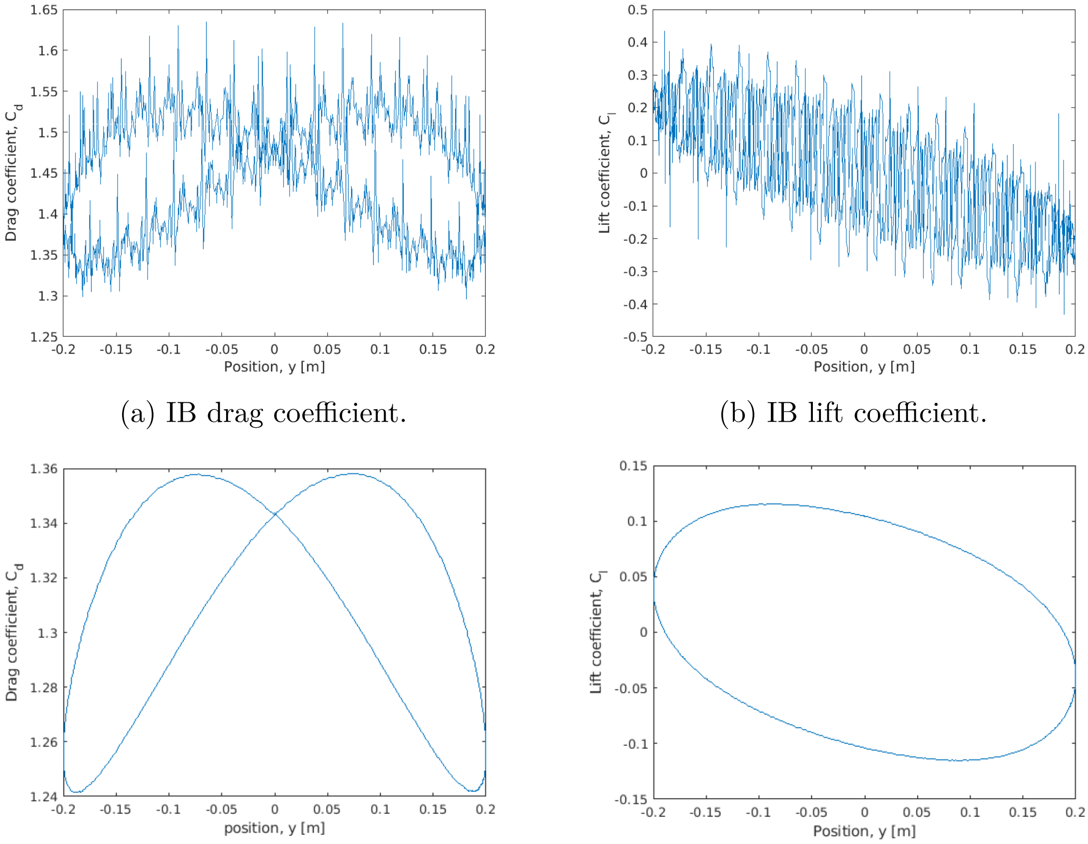
y = 0.1, moving up

y = 0.1, moving down

y = 0, moving up

y = 0, moving down

Figure 7.16: Vorticity contours during the top half-cycle of oscillation for the IB grid.



(a) IB drag coefficient.

(b) IB lift coefficient.

(c) BF drag coefficient.

(d) BF lift coefficient.

Figure 7.17: Time-periodic variation of the aerodynamic coefficients for the oscillating cylinder in crossflow.

8 Discussions

The simplest case of 2D laminar channel flow was mainly used to benchmark different types of IB grids: one with an arbitrary amount of "dead" cells outside of the area of interest, one with an optimized grid to minimize the number of "dead" cells and one non-conformal IB grid with the immersed surface intersecting the computational cells at arbitrary positions between the cells. As could be expected, the simulation time for the IB method with the most cells was the greatest. Surprisingly, the most efficient IB grid was the non-conformal grid. The amount of iterations required at each time step to converge the calculations was halved when using the non-conformal grid. This in turn greatly decreased the simulation time. However, even that grid was very inefficient with a simulation time of an order of magnitude greater than that with the BF grid. In addition, while qualitatively accurate, i.e. the resulting fully developed velocity profile having the parabolic shape, the magnitude of the fully developed centreline velocity predicted by the IB methods was underestimated by around 2% even for this simple case with an analytical solution. The entrance length calculated by all methods, be it IB or BF, was not in the ballpark of empirical formulation and was discarded as a metric for the performance of the IB method.

The second case was 3D laminar pipe flow, which was done to evaluate the software's ability of predicting the entrance length. Instability problems with cell ratio changes forced the use of a uniform grid. As could be expected with the increase in calculation cells, the simulation time with the IB method was an order of magnitude greater than that of the BF method. As earlier, the shape of the velocity profile was predicted quite accurately, but centreline velocity was this time underestimated by around 8%. Also, the entrance length was closer to empirical formulation with an underestimation of around 9%.

The third simulation case consisted of two different Reynolds number simulations of flow past a stationary cylinder: steady state with Re = 40 and unsteady with Re = 100. For the steady state case, the predicted circulation bubble length was overestimated by 5% when compared to most results found in literature. Furthermore, the calculated drag coefficient was overestimated by 13%. The manual generation of the pressure distribution around the wall was not straightforward. As described in chapter 5, the pressure in IB cells is determined for force calculation purposes. However, it appears not to be stored for later use. As expected, the extracted pressure distribution at the cylinder wall is unusable as is. However, an approximate of the distribution can be obtained by modifying the sampling points from the surface to the first fluid node outside of the cylinder. The approximate pressure distribution seems to be in line with that predicted by the BF method and that found in literature. The skin friction coefficient is even more tedious to obtain, since the interpolation procedure laid out in chapter 5 is once again not stored for later use. However, an approximate can be made and is in good agreement with the one predicted by the BF method. The unsteady state case gave a 4% underprediction of the Strouhal number. The average drag coefficient values found in literature had a large variation (1.26 – 1.447), and the ones predicted by both the IB and BF methods fell into the range at 1.445 and 1.447, respectively. The amplitude of the drag coefficient variation and the root mean square of the lift coefficient were clearly underestimated by the IB method. In general, the results seem to be in good agreement when considering the confinement effect due to the boundaries. [29] Uhlmann [28] experienced similar results and improved them by increasing the domain dimensions to 40D $\times$ 40D while keeping the same resolution. This was not done here, since the IB method was quite CPU intensive already, and further increasing the cell count would have led to exhaustive simulation times.

The simulation of the transversely oscillating cylinder in crossflow showed that the applied IB method faces difficulties in accurately resolving moving boundaries. As can be seen from figures 7.17a and 7.17b, the calculated boundary forces show ripples during simulation. Although the shapes of the time-periodic variations and the average drag coefficient seem to be fairly accurate, the method has difficulties in predicting the lift coefficient. The reason for the ripples remains unclear, however it is suspected to be caused by the changing interpolation stencil for IB cell velocity prediction in combination with the interpolations of the dynamic mesh method. Figure 8.1 shows a comparison of IB cells near the cylinder surface between two time steps.

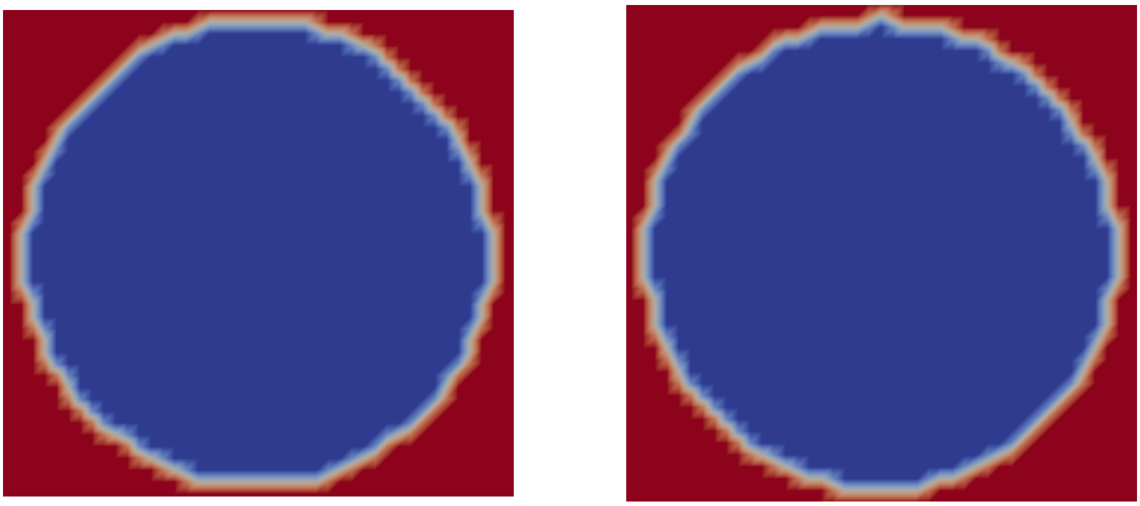


Figure 8.1: The change in IB cells during one time step.

Generally, the implemented IB method is able to predict the results for fairly simple flow cases. Most of the problems associated with accuracy could be addressed by increasing grid resolution. The implementation is very inefficient when compared to the BF method. IB methods have caused slow convergence rates for the pressure equation in the past. [10] Due to the inefficiency of the applied algorithm, a non-uniform grid or parallel simulation would be required to keep the simulation times at bay. However, adjusting grid spacing turned out to be problematic. With some adjustments the simulation would run, and with others it would crash. Additionally, inconsistencies in simulation results were observed when running the same flow case in parallel. Furthermore, it appeared that if the computational domain was divided so that processor boundaries intersected the immersed boundary at certain positions, the whole parallel computation would fail. This problem was emphasized in the case of a moving boundary. Although, processor boundaries were manually generated so that the IB surface was not intersected in e.g. the case of the cylinder in crossflow, the results differed to those obtained using a single processor. The underlying reasons for the problems described above remain unresolved but may be caused by problems in solving the Poisson equation. Finally, stability problems were encountered when trying to achieve large-eddy simulation of 3D pipe flow. The conclusion was that the implemented IB method was unable to handle cyclic boundary conditions at the pipe inlet and outlet.

8.1 Future of IB in FOAM-Extend

As of the end of 2018 Jasak [70] is working on a new IB method for the FOAM-Extend toolbox called the immersed boundary surface (IBS) method. In the new method, all cells that are intersected by the boundary are considered active IBS cells in contrast to the old method, where all cells with their centre inside the boundary were considered "dead" regardless of being intersected by the surface or not. Furthermore, the new method seems to be a variation of the cut-cell method described in section 4.4 where the intersected cells are modified during simulation to mimic a body-fitted mesh. This is done by a piecewise linear approximation of the surface similar to that in figure 4.5b. The new method has been tested by simulating flow around a Francis turbine in different operating conditions [71] and hydrodynamics of a ship hull [72]. However, further validation is required.

9 Conclusions

The history and underlying theory of different types of immersed boundary methods were presented in chapter 4. Currently, there are multiple variations of the original immersed boundary method available. The methods can be divided into continuous and discrete forcing approaches, according to how the boundary forces are introduced into the governing equations. However, there are also methods, such as the cut-cell finite-volume approach that do not rely on the modification of the governing equation but are still considered IB methods.

The application of a discrete forcing IB approach to the open-source computational fluid dynamics toolbox OpenFOAM was described in chapter 5. Also, a user guide was presented.

In chapters 6 and 7, four benchmark cases were simulated with the IB method and compared to results of a body-fitted, conventional, gridding method in addition to results found in literature. The advantages of the IB method in the initialization of flow simulation were clear: the required grid was simple for every flow case and its generation was quick. Although the IB method shows promising results for flow cases with stationary structures, it is clearly inefficient when compared to the BF method. Furthermore, it has difficulties in accurately resolving moving boundaries.

In addition, problems associated with parallel computing rendered the current implementation impractical for computationally demanding CFD simulations. A new IB method, which uses the cut-cell finite-volume approach, is under development for the FOAM-Extend library. It may be able to tackle some of the aforementioned problems, but it is yet to be validated.

References

[1] F. Sotiropoulos and X. Yang, "Immersed boundary methods for simulating fluid-structure interaction," *Prog. Aerosp. Sci.*, vol. 65:1, pp. 1-21, Feb. 2014. [accessed 1.12.2019] ISSN 0376-0421 (electronic). Available at https://doi.org/10.1016/j.paerosci.2013.09.003.

[2] E. Constant, J. Favier, M. Meldi, P. Meliga and E. Serre, "An immersed boundary method in *OpenFOAM*: Verification and validation," *Comput. Fluids*, vol. 157:1, pp. 55-72, Nov. 2017. [accessed 1.12.2019] ISSN 0045-7930 (electronic). Available at https://doi.org/10.1016/j.compfluid.2017.08.001.

[3] H. Jasak, H. Rusche, B. Gschaider, H. Nilsson, M. Beaudoin and V Škurić, "FOAM-Extend. (v. 4.0)". [accessed 23.11.2019]. Available at https://sourceforge.net/projects/foam-extend/files/foam-extend-4.0/.

[4] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, "Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations," *J. Comput. Phys.*, vol. 161:1, pp. 35-60, 2000. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1006/jcph.2000.6484.

[5] G. Iaccarino and G. Kalitzin, "Towards a Fast RANS Solver Based on the Immersed Boundary Technique," in *41st Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2003. [accessed 20.12.2019]. ISBN 978-1-62410-099-4 (electronic). Available at https://doi.org/10.2514/6.2003-770.

[6] R. Mittal, G. Iaccarino, "Immersed Boundary Methods," *Annu. Rev. Fluid. Mech.*, vol. 37:1, pp. 239-261, Jan. 2005. [accessed 1.12.2019]. ISSN 1545-4479. Available at https://doi.org/10.1146/annurev.fluid.37.061903.175743.

[7] E.P.C. Koh, H.M. Tsai, F. Liu, "Euler Solution Using Cartesian Grid with Least Squares Technique," in *41st Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2003. [accessed 1.12.2019]. ISBN 978-1-62410-099-4 (electronic). Available at https://doi.org/10.2514/6.2003-1120.

[8] W. Kim and H. Choi, "Immersed boundary methods for fluid-structure interaction: A review," *Int. J. Heat Fluid Fl.*, vol. 75, pp. 301-309, Feb. 2019. [accessed 1.12.2019]. ISSN 0142-727X (electronic). Available at https://doi.org/10.1016/j.ijheatfluidflow.2019.01.010.

[9] H. Jasak, D. Rigler, Ž. Tuković, "Design and Implementation of Immersed Boundary Method with Discrete Forcing Approach for Boundary Conditions," in Proc. of the 6th] European Conference on Computational Fluid Dynamics (ECFD VI), Barcelona, 2014, pp. 5319-5332. ISBN 978-84-942844-7-2.

[10] Y.H. Tseng and J.H. Ferziger, "A ghost-cell immersed boundary method for flow in complex geometry," *J. Comput. Phys.*, vol. 192:2, pp. 593-623,

Dec. 2003. [accessed 1.12.2019] ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/j.jcp.2003.07.024.

[11] C. Hirsch, "Structured and Unstructured Grid Properties," in *Numerical Computation of Internal and External Flows*. Oxford: John Wiley & Sons Ltd, 2007, Chapter 6, pp. 249-277. ISBN 978-0-7506-6594-0.

[12] D.J. Mavriplis, "Unstructured Mesh Related Issues in Computational Fluid Dynamics (CFD) – Based Analysis and Design," at the 11th International Meshing Roundtable, 2002, Ithaca, New York. 2002.

[13] T. Siikonen, "Laskentahilan laatiminen," in *Virtaussimulointi*. Espoo, Finland: Aalto University, 2014, Chapter 2, pp. 35-67.

[14] R. Krishnamurthy, “The Art and Science of Meshing Airfoils,” 2017. [accessed 26.8.2019]. Available at http://blog.gridpro.com/the-art-and-science-of-meshing-airfoils/.

[15] S.H. Lee, P. Jenny, H.A Tchelepi, "A Finite-Volume Method with Hexahedral Multiblock Grids for Modelling Flow in Porous Media," *Computat. Geosci.*, vol. 6:3-4, pp. 353-379, Sep. 2002. [accessed 1.12.2019]. ISSN 1573-1499 (electronic). Available at https://doi-org.libproxy.aalto.fi/10.1023/A:1021287013566 .

[16] J. Chao, A. Haselbacher, S. Balachandar, "A Massively Parallel Multi-Block Hybrid Compact-WENO Scheme for Compressible Flows," *J. Comput. Phys.*, vol. 228:19, pp. 7473-7491, Oct. 2009. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/j.jcp.2009.07.005.

[17] R. Krishnamurthy, "Role of CFD in designing efficient fuel injection systems," 2017. [accessed 9.12.2019]. Available at http://blog.gridpro.com/fuel-injection/

[18] M.M. Rai, "Patched-Grid Calculations with the Euler and Navier–Stokes Equations: Theory and Application," National Aeronautics and Space Administration, Washington D.C., NASA Technical Memorandum no. 88228, Feb. 1986.

[19] J.L. Steger, F.C. Dougherty, J.A. Benek, "A Chimera Grid Scheme," *Advances in Grid Generation*, eds. K.N. Ghia and U.Ghia, ASME FED, Vol. 5, pp. 59-71, Jun. 1983.

[20] A. Kageyama and T. Sato, "Ying-Yang grid: An overset grid in spherical geometry," *Geochem. Geophys. Geosyst.*, vol. 5:3, Sep. 2004. [accessed 1.12.2019]. ISSN 1525-2027 (electronic). Available at https://doi.org/10.1029/2004GC000734.

[21] A.J. Chen and Y. Kallinderis, "Adaptive Hybrid (Prismatic-Tetrahedral) Grids for Incompressible Flows," *Int. J. Numer. Meth. Fl.*, vol. 26:9, pp. 1085-1105, May 1998. [accessed 1.12.2019]. ISSN 0271-2091 (electronic). Available at https://doi.org/10.1002/(SICI)1097-0363(19980515)26:9<1085::AID-FLD683>3.0.CO;2-5.

[22] D. Kim and H. Choi, "A Second-Order Time-Accurate Finite Volume Method for Unsteady Incompressible Flow on Hybrid Unstructured Grids," *J. Comput. Phys.*, vol. 162:2, pp. 411-428, Aug. 2000. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1006/jcph.2000.6546.

[23] T.E. Tezduyar, "Finite Element Methods for Fluid Dynamics with Moving Boundaries and Interfaces," in *Encyclopedia of Computational Mechanics, Volume 3: Fluids*, eds. E. Stein, R. De Bors and T.J.R. Hughes, Oxford: John Wiley & Sons Ltd, 2004, Chapter 17, pp. 123-123. ISBN 9780470846995.

[24] M.M. Selim and R.P. Koomullil, "Mesh Deformation Approaches - A Survey," *J. of Phys. Math.*, vol. 7:2, 2016. [accessed 1.12.2019]. ISSN 2090-0902 (electronic). Available at http://dx.doi.org/10.4172/2090-0902.1000181.

[25] C. Hirsch, "The Basic Equations of Fluid Dynamics," in *Numerical Computation of Internal and External Flows.* Oxford: John Wiley & Sons Ltd, 2007, Chapter 1, pp. 27-64. ISBN 978-0-7506-6594-0.

[26] H. Jasak, "Error Analysis and Estimation for the Finite Volume Method with Application to Fluid Flows," Ph.D. thesis, Department of Mechanical Engineering, Imperial College of Science, Technology and Medicine, London, 1996.

[27] C.S. Peskin, "Flow Patterns Around Heart Valves: A Digital Computer Method for Solving the Equations of Motion," Ph.D. dissertation, Albert Einstein College of Medicine, Yeshiva University, NY, 1972.

[28] M. Uhlmann, "An immersed boundary method with direct forcing for the simulation of particulate flows," *J. Comput. Phys.*, vol. 209:2, pp. 448-476, Nov. 2005. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/j.jcp.2005.03.017.

[29] M.C. Lai and C.S. Peskin, "An Immersed Boundary Method with Formal Second-Order Accuracy and Reduced Numerical Viscosity," *J. Comput. Phys.*, vol. 160:2, pp. 705-719, May 2000. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1006/jcph.2000.6483.

[30] R.P. Beyer and R.J. Leveque, "Analysis of a One-Dimensional Model for the Immersed Boundary Method," *SIAM J. Numer. Anal.*, vol. 29:2, pp. 332-364, Apr. 1992. [accessed 1.12.2019]. ISSN 0036-1429 (electronic). Available at http://www.jstor.org/stable/2158129.

[31] E.M. Saiki and S. Biringen, "Numerical Simulation of a Cylinder in Uniform Flow: Application of a Virtual Boundary Method," *J. Comput. Phys.*, vol. 123:2, pp. 450-465, Feb. 1996. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1006/jcph.1996.0036.

[32] Y. Mori and C.S. Peskin, "Implicit second-order immersed boundary methods with boundary mass," *Comput. Methods Appl. Mech. Engrg.*, vol. 197:25-28,

pp. 2049-2067, Apr. 2008. [accessed 1.12.2019]. ISSN 0045-7825 (electronic). Available at https://doi.org/10.1016/j.cma.2007.05.028.

[33] D. Goldstein, R. Handler, L. Sirovich, "Modeling a No-Slip Flow Boundary with an External Force Field," *J. Comput. Phys.*, vol. 105:2, pp. 354-366, Apr. 1993. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1006/jcph.1993.1081.

[34] K. Khadra, P. Angot, S. Parneix, J.P. Caltagirone, "Fictious domain approach for numerical modelling of Navier–Stokes equations," *Int. J. Numer. Meth. Fl.*, vol. 34:8, pp. 651-684, Dec. 2000. [accessed 1.12.2019]. ISSN 0271-2091 (electronic). Available at https://doi.org/10.1002/1097-0363(20001230)34:8<651::AID-FLD61>3.0.CO;2-D.

[35] H.C. Brinkman, "A calculation of the viscosity and the sedimentation constant for solutions of large chain molecules taking into account the hampered flow of the solvent through these molecules," *Physics*, vol. 13:8, pp. 447-448, Sep. 1947. [accessed 1.12.2019]. ISSN 0031-8914 (electronic). Available at https://doi.org/10.1016/0031-8914(47)90030-X.

[36] R. Glowinski, T.W. Pan, J. Périaux, "Distributed Lagrange multiplier methods for incompressible viscous flow around moving rigid bodies," *Comput. Methods Appl. Mech. Engrg.*, vol. 151:1-2, pp. 181-194, Jan. 1998. [accessed 1.12.2019]. ISSN 0045-7825 (electronic). Available at https://doi.org/10.1016/S0045-7825(97)00116-3.

[37] L. Lee and R.J. Leveque, "An Immersed Interface Method for Incompressible Navier–Stokes Equations," *SIAM J. Sci. Comput.*, vol. 25:3, pp. 832-856, 2003. [accessed 1.12.2019]. ISSN 1095-7197 (electronic). Available at https://doi.org/10.1137/S1064827502414060.

[38] K. Taira and T. Colonius, "The immersed boundary method: A projection approach," *J. Comput. Phys.*, vol. 225, pp. 2118-2137, Aug. 2007. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/j.jcp.2007.03.005.

[39] C.S. Peskin, "The fluid dynamics of heart valves: experimental, theoretical, and computational methods," *Ann. Rev. Fluid Mech.*, vol. 14:1, pp. 235-259, 1982. [accessed 1.12.2019]. ISSN 1545-4479 (electronic). Available at https://doi.org/10.1146/annurev.fl.14.010182.001315.

[40] S.O. Unverdi and G. Tryggvason, "A Front-Tracking Method for Viscous, Incompressible, Multi-fluid Flows," *J. Comput. Phys.*, vol. 100, pp. 25-37, May 1992. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/0021-9991(92)90307-K.

[41] H. Jasak and Ž. Tuković, CFD with OpenSource Software Course. Class Lecture, Topic "Immersed Boundary Method in FOAM - Theory, Implementation and Use." Department of Mechanics and Maritime Sciences, Chalmers

University, Gothenburg, Sep. 29, 2015. [accessed 25.7.2019]. Available at http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2015/HrvojeJasak/ImmersedBoundary.pdf

[42] J. Mohd-Yusof, "Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries," in *Annual Research Briefs 1997*, ed. D. Spinks, Stanford: NASA Center for Turbulence Research, 1997, pp. 317-327. [accessed 1.12.2019]. Available at https://ctr.stanford.edu/annual-research-briefs-1997.

[43] J. Kim, D. Kim and H. Choi, "An Immersed-Boundary Finite-Volume Method for Simulation of Flow in Complex Geometries," *J. Comput. Phys.*, vol. 171:1, pp. 132-150, Jul. 2001. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1006/jcph.2001.6778.

[44] R. Verzicco, G. Iaccarino, M. Fatica and P. Orlandi, "Flow in an impeller stirred tank using an immersed boundary method," *AIChE J.*, vol. 50:6, pp. 1109-1118, 2004. [accessed 1.12.2019]. ISSN 0001-1541 (electronic). Available at https://doi.org/10.1002/aic.10117.

[45] S. Majumdar, G. Iaccarino and P. Durbin, "RANS solvers with adaptive structured boundary non-conforming grids," *Annual Research Briefs 2001*, eds. P. Moin, W.C. Reynolds and N.N. Mansour, Stanford: NASA Center for Turbulence Research, 2001, pp. 353-366. [accessed 1.12.2019]. Available at https://ctr.stanford.edu/annual-research-briefs-2001.

[46] A. Mark and B.G.M van Wachem, "Derivation and validation of a novel implicit second-order accurate immersed boundary method," *J. Comput. Phys.*, vol. 227:13, pp. 6660-6680, Jun. 2008. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/j.jcp.2008.03.031.

[47] D. Pan and T.-T. Shen, "Computation of incompressible flows with immersed bodies by a simple ghost cell method," *Int. J. Numer. Meth. Fl.*, vol. 60:12, pp. 1378-1401, Aug. 2009. [accessed 1.12.2019]. ISSN 0271-2091 (electronic). Available at https://doi.org/10.1002/fld.1942.

[48] A.F. Shinn, M.A. Goodwin and S.P. Vanka, "Immersed boundary computations of shear- and buoyancy-driven flows in complex enclosures," *Int. J. Heat Mass Tran.*, vol. 52:17-18, pp. 4082-4089, Aug. 2009. [accessed 1.12.2019]. ISSN 0017-9310 (electronic). Available at https://doi.org/10.1016/j.ijheatmasstransfer.2009.03.044.

[49] R. Ghias, R. Mittal and T.S. Lund, "A non-body conformal grid method for simulation of compressible flows with complex immersed boundaries," in *42nd Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2004. [accessed 1.12.2019]. ISBN 978-1-62410-078-9 (electronic). Available at https://arc.aiaa.org/doi/abs/10.2514/6.2004-80.

[50] G. Kalitzin and G. Iaccarino, "Towards an immersed boundary RANS flow solver," in *41st Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2003. [accessed 1.12.2019]. ISBN 978-1-62410-099-4 (electronic). Available at https://doi.org/10.2514/6.2003-770.

[51] D.K. Clarke , M.D. Salas and H.A. Hassan, "Euler Calculations for Multielement Airfoils Using Cartesian Grids," *AIAA J.*, vol. 24:3, pp. 353-358, Mar. 1986. [accessed 1.12.2019]. ISSN 1533-385X (electronic). Available at https://doi.org/10.2514/3.9273.

[52] T. Ye, R. Mittal. H.S. Udaykumar, W. Shyy, "An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries", *J. Comput. Phys.*, vol. 156:2, pp. 209-240, Dec. 1999. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1006/jcph.1999.6356.

[53] M.P. Kirkpatrick, S.W. Armfield and J.H. Kent, "A representation of curved boundaries for the solution of the Navier–Stokes equations on a staggered three-dimensional Cartesian grid," *J. Comput. Phys.*, vol. 184:1, pp. 1-36, Jan. 2003. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/S0021-9991(02)00013-X.

[54] H.S. Udaykumar, R. Mittal and P. Rampunggoon, "Interface tracking finite volume method for complex solid-fluid interactions on fixed meshes," *Commun. Numer. Meth. Engrg.*, vol. 18:2, pp. 89-97, Dec. 2001. [accessed 1.12.2019]. ISSN 1069-8299 (electronic). Available at https://doi.org/10.1002/cnm.468.

[55] R. Mittal, C. Bonilla, H.S. Udaykumar, "Cartesian grid methods for simulating flows with moving boundaries," in: C.A. Brebbia, G.M. Carlomagno and P. Anagnostopoulos, editors. Computational Methods and Experimental Measurements XI: Proceedings of the 11th International Conference on Computational Methods and Experimental Measurements (CMEM); 2003 May 12-14; Chalkidike, Greece. Boston: WIT, 2003, pp 557-556. ISBN 978-1-85312-969-8.

[56] R. Mittal, V. Seshadri and H.S. Udaykumar, "Flutter, Tumble and Vortex Induced Autorotation," *Theoret. Comput. Fluid Dynamics*, vol. 17:3, pp. 165-170, Apr. 2004. [accessed 1.12.2019]. ISSN 1432-2250 (electronic). Available at https://doi.org/10.1007/s00162-003-0101-5.

[57] OpenSCAD. (v. 2019.05). [accessed 23.11.2019]. Available at https://www.openscad.org/.

[58] OpenFOAM. (v. 5.0). [accessed 1.12.2019]. Available at https://openfoam.org/release/5-0/.

[59] F.M. White, "Solutions of the Newtonian Viscous-Flow Equations," in *Viscous Fluid Flow*. New York: McGraw-Hill, 2006, Chapter 3, pp. 96-214. ISBN 007-124493-X.

[60] B. Hof, C.W.H. van Doorne, J. Westerweel, F.T.M. Nieuwstadt, H. Faisst, B. Eckhardt, H. Wedin, R.R. Kerswell and F. Waleffe, "Experimental Observation of Nonlinear Traveling Waves in Turbulent Pipe Flow," *Science*, vol. 305:5690, pp. 1594-1598, Sep. 2004. [accessed 1.12.2019]. ISSN 0036-8075 (electronic). Available at https://search.proquest.com/docview/213573617?accountid=27468.

[61] A.L.F. Lima E Silva, A. Silveira-Neto, J.J.R. Damasceno, "Numerical simulation for two-dimensional flows over a circular cylinder using the immersed boundary method," *J. Comput. Phys.*, vol. 189:2, pp. 351-370, Aug. 2003. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/S0021-9991(03)00214-6.

[62] N. Zhang, Z.C. Zheng and S. Eckels, "Study of heat-transfer on the surface of a circular cylinder in flow using an immersed-boundary method," *Int. J. Heat Fluid Fl.*, vol. 29:6, pp. 1558-1566, Oct. 2008. [accessed 1.12.2019]. ISSN 0142-727X (electronic). Available at https://doi.org/10.1016/j.ijheatfluidflow.2008.08.009.

[63] D. Kim and H. Choi, "Immersed boundary method for flow around an arbitrarily moving body," *J. Comput. Phys.*, vol. 212:2, pp. 662-680, Mar. 2006. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/j.jcp.2005.07.010.

[64] N. Zhang, Z.C. Zheng, "An improved direct-forcing immersed-boundary method for finite difference applications," *J. Comput. Phys.*, vol 221:1, pp. 250-268, Jan. 2007. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/j.jcp.2006.06.012.

[65] X. Yang, X. Zhan, Z. Li, G.-W. He, "A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations," *J. Comput. Phys.*, vol. 228:20, pp. 7821-7836, Nov. 2009. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1016/j.jcp.2009.07.023.

[66] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, "A Sharp Interface Cartesian Grid Method for Simulating Flows with Complex Moving Boundaries," *J. Comput. Phys.*, vol. 174:1, pp. 345-380, Nov. 2001. [accessed 1.12.2019]. ISSN 0021-9991 (electronic). Available at https://doi.org/10.1006/jcph.2001.6916.

[67] C. Liao, Y. Chang, C. Lin, J.M. McDonough, "Simulating flows with moving rigid boundary using immersed-boundary method," *Comput. Fluids*, vol. 39:1, pp. 152-167, Jan. 2010. [accessed 1.12.2019]. ISSN 0045-7930 (electronic). Available at https://doi.org/10.1016/j.compfluid.2009.07.011.

[68] J. Park, K. Kwon and H. Choi, "Numerical Solutions of Flow Past a Circular Cylinder at Reynolds Numbers up to 160," *KSME Int. J.*, vol. 12, pp. 1200-1205,

Nov. 1998. [accessed 1.12.2019]. ISSN 1976-3824 (electronic). Available at https://doi.org/10.1007/BF02942594.

[69] E. Guilmineau and P. Queutey, "A numerical simulation of vortex shedding from an oscillating circular cylinder," *J. Fluid. Struct.*, vol. 16:6, pp. 773-794, Aug. 2002. [accessed 19.12.2019]. ISSN 0889-9746 (electronic). Available at https://doi.org/10.1006/jfls.2002.0449.

[70] H. Jasak, "Immersed Boundary Surface Method in foam-extend," presented at the OpenFOAM in Hydraulic Engineering Conference, 2018, Karlsruhe, Germany. 2018. [accessed 1.12.2019]. ISSN 1976-3824 (electronic). Available at https://foam-extend.fsb.hr/wp-content/uploads/2018/12/04_Jasak_Immersed_Boundary.pdf.

[71] R. Anderluh, "Validation of the immersed boundary surface method in computational fluid dynamics," M.S. thesis, Faculty of Mech. Eng. and Naval Arch., University of Zagreb, Zagreb, Croatia, 2019. [accessed 1.12.2019]. Available at https://cfd.fsb.hr/wp-content/uploads/2019/03/RoberAnderluhMSc_2019.pdf.

[72] I. Gatin, V. Vukčević and H. Jasak, "Advances in dynamic mesh modelling in foam-extend: Overset and immersed boundary," in Faculty of Mechanical Engineering and Naval Architecture, Zagreb, 2019. Naval Hydro Workshop, Zagreb.

A Derivation of the analytical Poiseuille flow solution

Initially, a bulk flow of velocity U_∞ is introduced into the channel. During the entrance region, $0 < \mathbf{x} < L_e$, where L_e is the entrance length, the flow varies in $\mathbf{x}$ and $\mathbf{y}$ directions. After the entrance region, the flow is *fully developed* and only varies with $\mathbf{y}$. For fully developed 2D incompressible flow, the continuity and momentum equations reduce to

$$\frac{\partial u}{\partial x} = 0 \tag{A1}$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{1}{\mu}\frac{\mathrm{d}p}{\mathrm{d}x} = constant \tag{A2}$$

$$\frac{\partial p}{\partial y} = 0. \tag{A3}$$

By integrating equation (A2) twice with respect to $\mathbf{y}$, we get:

$$u(\mathbf{y}) = \frac{1}{2\mu}\frac{\mathrm{d}p}{\mathrm{d}x}\mathbf{y}^2 + d_1 y + d_2 \tag{A4}$$

The constants d_1 and d_2 can be determined from the no-slip conditions at the walls, i.e. $y = \pm h$. Thus, we can solve that $d_1 = 0$ and $d_2 = \frac{1}{2\mu}\frac{\mathrm{d}p}{\mathrm{d}x}h$. The axial velocity can then be determined from:

$$u(\mathbf{y}) = \frac{1}{2\mu}\frac{\mathrm{d}p}{\mathrm{d}x}(h^2 - \mathbf{y}^2). \tag{A5}$$

To determine the pressure gradient, we first calculate the average velocity for the parabolic velocity profile. This is done by integrating the velocity with respect to the cross-sectional area:

$$u_{ave} = \frac{\int_A v_x dA}{\int_A dA} = \frac{1}{A}\int_A v_x dA \quad , \tag{A6}$$

where the cross-sectional channel area is $A = Dz = 2hz$. Since $\mathrm{d}A = \mathrm{d}z\mathrm{d}y$ and the velocity is only dependent of $\mathbf{y}$, we get:

$$\frac{1}{\mathbf{zy}}\int_0^z \int_{-h}^h v_x \mathrm{d}y\mathrm{d}z = \frac{1}{2h}\int_{-h}^h \frac{1}{2\mu}\frac{\mathrm{d}p}{\mathrm{d}x}\left(h^2 - y^2\right)\mathrm{d}y \tag{A7}$$

Evaluating equation (A7) gives us:

$$u_{ave} = \frac{1}{4\mu h}\frac{\mathrm{d}p}{\mathrm{d}x}\left[h^2 y - \frac{1}{3}y^3\right]_{-h}^{h} = \frac{1}{3\mu}\frac{\mathrm{d}p}{\mathrm{d}x}h^2. \tag{A8}$$

Due to the conservation of mass, the average velocity of the Poiseuille flow velocity profile corresponds to the inlet bulk flow velocity U_∞, thus the pressure gradient is:

$$\frac{\mathrm{d}p}{\mathrm{d}x} = \frac{3\mu U_\infty}{h^2} \tag{A9}$$

Combining equation (A9) with the velocity equation (A5) at $\mathbf{y} = 0$, we get an equation for the centreline axial velocity, or $u(y = 0) = U_{max}$:

$$U_{max} = \frac{3}{2} U_{\infty}. \tag{A10}$$

Thus, the velocity profile gets the final analytical form of:

$$u(y) = \frac{3}{2} U_{\infty} \left(h^2 - y^2 \right). \tag{A11}$$