University of Zagreb
Faculty of Mechanical Engineering and Naval Architecture

MASTER'S THESIS

Robert Anderluh

Zagreb, 2019

University of Zagreb
Faculty of Mechanical Engineering and Naval Architecture

MASTER'S THESIS

VALIDATION OF THE IMMERSED BOUNDARY SURFACE METHOD IN COMPUTATIONAL FLUID DYNAMICS

Mentor:
prof. dr. sc. Hrvoje Jasak

Student:
Robert Anderluh

Zagreb, 2019

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have fully cited all used sources and I have only used the ones given in the list of references.

I would like to express my sincere gratitude to my mentor, Professor Hrvoje Jasak, for an incredible opportunity to learn and grow as a young engineer. He can be credited for creating a phenomenal framework for all of his students to do so.

I would further like to thank everyone in Professor Jasak's team, but especially Tessa Uroić, as her will and ability to help were tireless.

My student colleagues from room 816 made the process of creating my thesis much less strenuous and much more entertaining, which was a great help.

Last but not least, I would like to thank all of my family and friends for the support and good times had during my student years.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
procesno-energetski, konstrukcijski, brodostrojarski i inženjersko modeliranje i računalne simulacije

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

DIPLOMSKI ZADATAK

Student: **Robert Anderluh** Mat. br.: 0035195170

Naslov rada na hrvatskom jeziku: **Validacija metode uronjene granične površine u numeričkoj mehanici fluida**

Naslov rada na engleskom jeziku: **Validation of the Immersed Boundary Surface Method in Computational Fluid Dynamics**

Opis zadatka:

The most cumbersome tasks in Computational Fluid Dynamics (CFD) simulations are mesh generation for complex geometry and handling of general deforming boundary problems. While the moving deforming polyhedral mesh support mitigates the problem, it still does not offer a robust and cost-efficient solution. Recently developed Immersed Boundary Surface (IBS) method shows promise by automatically handling complex deforming geometry at the discretisation matrix level, but at a cost of increased mesh-induced discretisation error near the boundary. This is particularly cumbersome in high-Reynolds number turbulent flows.

The objective of this study is to establish the accuracy of the IBS method for turbulent incompressible flows in comparison with body-fitted meshes on canonical problems.

The candidate shall perform the following tasks within this project:

- Perform literature survey of immersed boundary methods in CFD. Describe in detail the numerical aspects of the Immersed Boundary Surface and its implementation in OpenFOAM;
- Select a set of validation cases in 2-D and 3-D which feature turbulent boundary layers in steady-state incompressible flow, to be used in the comparison. Where possible, identify experimental available data;
- Generate body-fitted meshes for the selected cases and perform numerical simulations using the wall-function near-wall treatment;
- Set up the equivalent cases using the IBS method and perform the simulations. Compare the results of the body-fitted and IBS simulations in global parameters (lift, drag, pressure drop) and local solution (velocity and turbulence profiles). Report on the near-wall y+ values for body-fitted and IBS and comment on the accuracy of results.

The Thesis shall list the bibliography and any assistance received during this study.

Zadatak zadan:	Rok predaje rada:	Predviđeni datumi obrane:
17. siječnja 2019.	21. ožujka 2019.	27., 28. i 29 ožujka 2019.

Zadatak zadao: Predsjednica Povjerenstva:

Prof. dr. sc. Hrvoje Jasak

Prof. dr. sc. Tanja Jurčević Lulić

Contents

List of Figures

List of Tables

Nomenclature

Latin Characters

$[A]$	-	Matrix composed of unknown variables coefficients
a	m/s	Speed of sound
$\mathbf{a}$	-	Gauss' theorem generic variable
$\mathbf{a}_f$	-	Gauss' theorem generic variable on a face center
$a_{i,j}$	-	Components of $[A]$
a_N	-	Off-diagonal contribution to the solution matrix
a_P	-	Diagonal contribution to the solution matrix
B_1, B_2	-	IBM boundary conditions enforcement locations
$[b]$	-	Right-hand side contribution of the discretized system of equations
b	-	Right-hand side contribution in a discretized equation
b_i	-	Components of $[b]$
C_D	-	Drag coefficient
C_L	-	Lift coefficient
C_P	-	Pressure coefficient
$\mathbf{d}$	m	Vector connecting two neighbouring cell centers
e	m^2/s^2	Energy density
F	-	Flux through a face
F_i	-	ith fluid cell center
$\mathbf{F}_k$	kg/ms^2	kth Lagrangian point stress vector
f_e	-	East face
f_i	-	Immersed boundary face
f_{sw}	-	Southwest face
$\mathbf{f}_b$	kg/m^2s^2	Volume forces
$\mathbf{f}_m$	kg/s^2	IBM momentum equation forcing term
$\mathbf{g}$	m/s^2	Gravitational acceleration
G	-	Ghost cell center
K	-	Permeability of a medium
k	m^2/s^2	Turbulence kinetic energy

$\mathbf{k}$	-	Non-orthogonal correction coefficient
l	m	Characteristic length scale
Ma	-	Mach number
m	-	Number of shadow face neighbours for master patches
n	-	Number of master face neighbours for shadow patches
n_c	m	Normal coordinate
p	m^2/s^2	Kinematic pressure
Q	kg/ms^2	Volume energy source
$\mathbf{q}$	kg/s^2	Heat flux
q_v	-	Volume source or sink of a generic transported variable
q_p	-	Source term component dependent on ϕ
q_u	-	Source term component not dependent on ϕ
Re	-	*Reynolds number*
$\mathbf{r}$	m	Polar coordinates position vector
$\mathbf{S}$	m^2	Surface area vector
$\lvert S_M \rvert$	m^2	Surface area of master facet
$\lvert S_M \rvert$	m^2	Surface area of shadow facet
$\lvert S_{\cap_{MtoS}} \rvert$	m^2	Intersection surface area between master and shadow facets
T	s	A specified time period
t	s	Time
t_c	m	Tangent coordinate
$\mathbf{u}$	m/s	Velocity vector
$u_{x,y,z}$	m/s	Velocity in x, y, z direction
$\mathbf{u}_I$	m/s	Velocity in an inertial reference frame
$\mathbf{u}_R$	m/s	Velocity in a relative reference frame
V	m^3	Volume
V_P	m^3	Current cell volume
$W_{M_n toS_i}$	-	Master facet to shadow facets weighting factor
$W_{S_m toM_j}$	-	Shadow facet to master facets weighting factor
$\mathbf{x}$	m	Position vector
$\mathbf{x}_k$	m	kth Lagrangian point position vector

$\mathbf{x}_k^e$	m	Equilibrium location of the kth Lagrangian point
$\mathbf{x}_P$	m	Current cell position vector
y^+	-	Dimensionless wall distance

Greek Characters

$\alpha,\ \beta$	-	General IBM forcing term coefficients
α_P	-	Pressure equation under-relaxation coefficient
α_u	-	Momentum equation under-relaxation coefficient
γ	-	Diffusion coefficient
γ_f	-	Face area correction
δ	-	Dirac delta function
κ	kg/s^2	Positive spring constant
μ	kg/ms	Dynamic viscosity
μ_t	kg/ms	Eddy viscosity
ν	m^2/s	Kinematic viscosity
ν_t	m^2/s	Kinematic eddy viscosity
ρ	kg/m^3	Density
σ	kg/ms^2	Surface forces in the fluid
τ	kg/ms^2	Stress tensor
τ^R	kg/ms^2	Reynolds stress tensor
ϕ	-	Generic transported variable
$[\phi]$	-	Solution vector
$[\phi]^n$	-	Solution vector in iteration n
ϕ_i	-	Components of the solution vector
$\overline{\phi}$	-	Generic variable mean component
ϕ'	-	Generic variable fluctuating component
ϕ_G	-	Generic variable value in a ghost cell center
ϕ_P	-	Current cell ϕ cell center value
ϕ_N	-	Neighbour cell ϕ cell center value
ϕ_f	-	Face center ϕ value
ϕ^t	-	Current time-step ϕ value
ϕ_{Si}	-	Shadow patch variable on ith shadow patch face
ϕ_{Mj}	-	Master patch variable on jth master patch face
ω	1/s	Eddy turnover time
ω_r	1/s	Angular velocity

Abbreviations

BF - body-fitted

CFD - Computational Fluid Dynamics

DNS - Direct Numerical Simulation

FVM - Finite Volume Method

GGI - General Grid Interface

GTE - General Transport Equation

IBM - Immersed Boundary Method

LES - Large Eddy Simulation

MRF - Multiple Reference Frame

RANS - Reynolds-Averaged Navier-Stokes Equations

SIMPLE - Semi-Implicit Method for Pressure Linked Equations

STL - Stereolithography

Abstract

The aim of this thesis is to describe the Immersed Boundary Method version implemented in $foam-extend$ 4.1, both its advantages and shortcomings.

The main goal of the Immersed Boundary Method is to simplify the mesh generation process in Computational Fluid Dynamics, which can lead to drastic reductions of human time needed for setting up simulations, especially for simulations with complex geometries. Additionally, it can offer certain advantages in simulations with moving meshes, as it can decrease the computational requirements of such cases.

The main shortcoming of the Immersed Boundary Method is loss of solution accuracy on immersed boundaries (surfaces of simulated objects).

The $foam-extend$ 4.1 Immersed Boundary Method is here validated on three cases: internal 2-D flow over a backward facing step, external flow around the Onera M6 wing, and the flow in a model Francis turbine, which is an especially interesting case, concerning the Immersed Boundary Method. The results of the Immersed Boundary Method simulations are compared to the results of equivalent body-fitted (conventional) simulations.

The simulation results are generally satisfactory, as the loss of accuracy was modest enough to assess the $foam-extend$ 4.1 implementation of the Immersed Boundary Method as successful.

Key words: *Computational Fluid Dynamics*, *CFD*, *OpenFOAM*, *foam-extend*, *Immersed Boundary Method*, *IBM*.

Sažetak

Cilj ovog rada je predstaviti teorijsku i praktičnu pozadinu metode uronjene granice implementirane u $foam-extend$ 4.1, odnosno njene prednosti i nedostatke.

Glavni cilj metode uronjene granice je pojednostavljenje izrade mreža u računalnoj dinamici fluida, što može dovesti do značajnog smanjenja količine ljudskog rada koji se mora uložiti pri pripremanju simulacija u računalnoj dinamici fluida, pogotovo kod simulacija sa složenim geometrijama. Također, metoda uronjene granice može donijeti određene prednosti kod simulacija s pomičnim mrežama, u vidu smanjenja računalne zahtjevnosti takvih simulacija.

Glavni nedostatak metode uronjene granice je smanjenje točnosti rješenja na uronjenim granicama (površinama simuliranih objekata).

Metoda uronjene granice implementirana u $foam-extend$ 4.1 je ovdje validirana na trima slučajevima: unutarnje strujanje u 2-D slučaju u cijevi sa naglim proširenjem, vanjsko strujanje oko Onera M6 krila i strujanje u Francisovoj turbini, što je pogotovo zanimljiv slučaj za metodu uronjene granice. Rezultati simulacija izvedenih uporabom metodom uronjene granice su uspoređeni sa rezultatima simulacija izvedenim konvencionalnim načinom izrade mreže.

Rezultati simulacija su zadovoljavajući, odnosno, smanjenje točnosti rješenja na uronjenim granicama je dovoljno maleno da implementaciju metode uronjene granice u $foam-extend$ 4.1 možemo ocjeniti kao dobru.

Ključne riječi: *računalna dinamika fluida*, *OpenFOAM*, *foam-extend*, *metoda uronjene granice*.

Prošireni sažetak

Računalna dinamika fluida je alat od velikog značaja u znanosti i inženjerstvu. Ona nam omogućava numeričko rješavanje jednadžbi koje opisuju strujanje fluida. U području energetike, turbostrojevi su najočitiji primjer kod čijeg razvoja se može primjeniti računalna dinamika fluida. Povećanje efikasnosti turbostrojeva dovodi do značajnih energetskih i financijskih ušteda. Dakle, razvoj računalne dinamike fluida može dovesti do poboljšanja procesa konstruiranja i proizvodnje turbostrojeva. Metoda uronjene granice je metoda čija uporaba potencijalno može pojednostaviti postupak pripreme kompleksnih simulacija u području računalne dinamike fluida. Cilj ovog rada je validirati metodu uronjene granice koja je implementirana u $foam-extend$ 4.1.

Simulacije koje su predstavljene u ovom radu su rađene u programskom paketu OpenFOAM [1], odnosno njegovoj verziji $foam-extend$ 4.1 [2].

Matematički model

Opća transportna jednadžba može poslužiti kao polazište za sve ostale jednadžbe mehanike fluida:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \mathbf{u}) - \nabla \cdot (\gamma \nabla \phi) = q_v. \tag{1}$$

Ako se u nju, umjesto općenite vrijednosti ϕ, umetnu gustoća ρ i količina gibanja $\rho\mathbf{u}$ te se uvede pretpostavka nestlačivog strujanja, što značajno pojednostavljuje simulaciju, dolazimo do:

- Jednadžbe kontinuiteta:

$$\nabla \cdot \mathbf{u} = 0. \tag{2}$$

- Jednadžbe količine gibanja:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = \mathbf{g} - \nabla p + \nabla \cdot (\nu \nabla \mathbf{u}). \tag{3}$$

Za sve simulacije izvođene u ovom radu je pretpostavljeno nestlačivo strujanje (Machov broj manji od 0,3) te su stoga jednadžbe 2 i 3 relevantne za rješavanje problema u ovom radu.

Modeliranje turbulencije

Turbulencija je nestacionarna, trodimenzionalna, nelinearna i stohastična pojava kod strujanja fluida pri visokim vrijednostima Reynoldsova broja. Pojava turbulencije značajno otežava rješavanje jednadžbi strujanja te se radi pojednostavljenja pribjegava modeliranju utjecaja kojeg turbulencija ima na strujanje. Najčešće se koristi Reynoldsovo osrednjavanje, kod kojeg se pretpostavlja da sve vrijednosti u domeni (polja brzine, tlaka i sl.) variraju oko srednje vrijednosti [3]. Budući da je za inženjere releventna srednja vrijednost, koriste se modeli koji računaju utjecaj varijacija na osrednjene vrijednosti. U ovom radu je odabran $k-\omega$ *SST* model turbulencije [4]. To je zonski model turbulencije s dvije jednadžbe: jedna za kinetičku energiju turbulencije k, a druga za specifičnu disipaciju turbulencije ω. Naziva se zonskim modelom jer se, ovisno o udaljenosti od zida, ponaša kao $k-\omega$ model (blizu zida) ili $k-\varepsilon$ model (dalje od zida).

Metoda kontrolnih volumena

Korištenjem metode kontrolnih volumena, jednadžbe strujanja se linearizirаju i numerički diskretiziraju kako bi se dobio sustav algebarskih linearnih jednadžbi [5]. Diskretizacija se vrši za svaki član opće transportne jednadžbe 1. Nakon diskretizacije jednadžbi, sustav jednadžbi ima sljedeći oblik [3]:

$$[A][x] = [b], \tag{4}$$

koji se rješava direktnim ili, češće, iterativnim metodama.

Za spregu jednadžbi korišten je SIMPLE algoritam [6], koji se koristi za rješavanje stacionarnih, nestlačivih strujanja fluida. U ovom radu su korištene i GGI i MRF metode. GGI [7] je sučelje koje služi za povezivanje regija u domeni, a na čijim granicama, koje su u dodiru, točke nisu raspodijeljene jednako. MRF [8] je način za kvazistacionarno modleiranje strujanja fluida u turbostrojevima. Simulira se tranzijentna pojava, ali na statičan način, odnosno rotacija bez promjene mreže, prevođenjem jednadžbi koje opisuju strujanje fluida u rotirajući referentni koordinatni sustav.

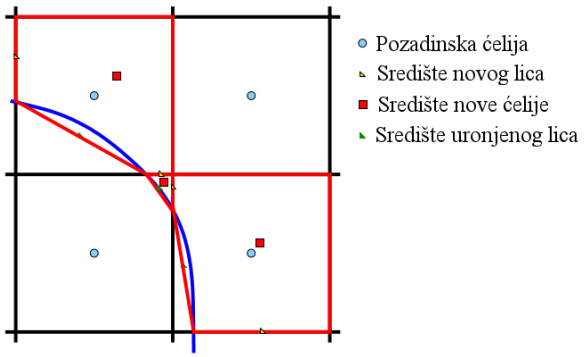
Metoda uronjene granice

Teško je jednoznačno opisati matematičku pozadinu metode uronjene granice jer su, od njenog prvog spomena do danas, u uporabi brojne verzije, s različitim ciljevima i načinima implemen-

tacije. Općenito se te metode mogu podijeliti na one s kontinuiranim izvorskim članom i s diskretnim izvorskim članom [9].

Kod metoda sa kontinuiranim izvorskim članom, definicija uronjene granice se uvrštava u jednadžbu prije diskretizacije. Kod metoda s diskretnim izvorskim članom se definicija uronjene granice uvrštava poslije diskretizacije. Dodatno se te metode mogu podijeliti na one kod kojih se rubni uvjeti na uronjenoj granici definiraju indirektno i na one kod kojih se rubni uvjeti na uronjenoj granici definiraju direktno [9].

U ovom radu koristimo metodu uronjene granice koja je implementirana u $foam-extend$ 4.1 [10]. Cilj implementacije je uzimanje u obzir prisutnosti uronjene granice što sličnije konvencionalnom pristupu s proračunskom mrežom prilagođenom geometriji, što podrazumijeva operaciju na pozadinskoj mreži koja uzima u obzir topologiju uronjenog tijela. Na taj način se zadržava detaljniji opis uronjene granice. Na slici 1 prikazana je shema umetanja uronjene granice u pozadinsku mrežu te, na taj način, stvaranja novih ćelija.



Slika 1: $foam-extend$ 4.1 shema umetanja uronjene granice u pozadinsku mrežu [10]

Na slici 1, crno je pozadinska mreža, plavo je uronjena granica, a crveno je nova, modificirana mreža. Nakon umetanja uronjene granice, računa se presjecište sa pozadinskom mrežom. Potom se odbacuju neaktivne ćelije (unutar uronjene granice) i modificiraju ćelije i lica koje uronjena granica presjeca. Modificiranim ćelijama se mijenja volumen i centar, a modificiranim licima površina i centar.

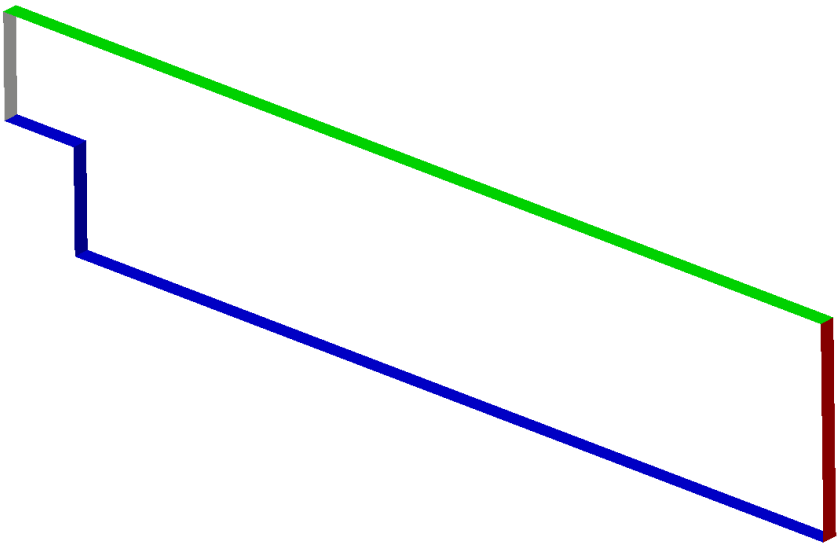
Validacija i rezultati validacije

Metoda uronjene granice validirana je na trima različitim geometrijama: 2-D strujanje u cijevi s naglim proširenjem, opstrujavanje Onera M6 krila i strujanje u Francisovoj turbini, uključujući

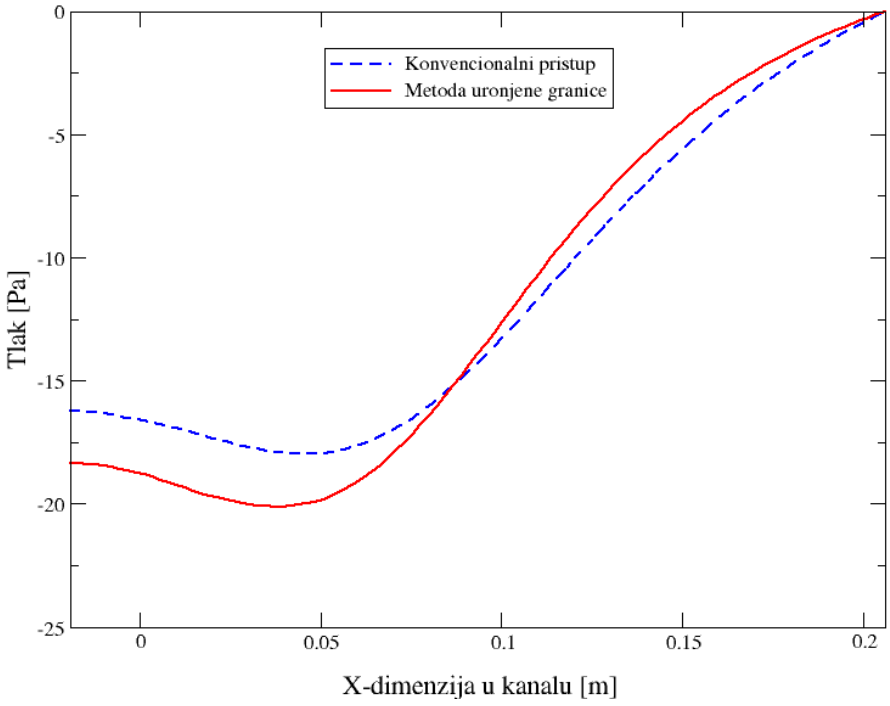
rotor i statorske lopatice. Simulacije su izvedene uz pretpostavku stacionarnog, nestlačivog strujanja algoritmima simpleFoam, odnosno MRFSimpleFoam u slučaju Francisove turbine. Kod turbulentnih simulacija korišten je $k-\omega\ SST$ model turbulencije. Sve tri grupe simulacija pokazale su zadovoljavajuće rezultate, od kojih će neki biti prikazani i u ovom proširenom sažetku, a detaljniji rezultati mogu se naći u poglavlju 5 u glavnom djelu ovog rada.

Cijev s naglim proširenjem

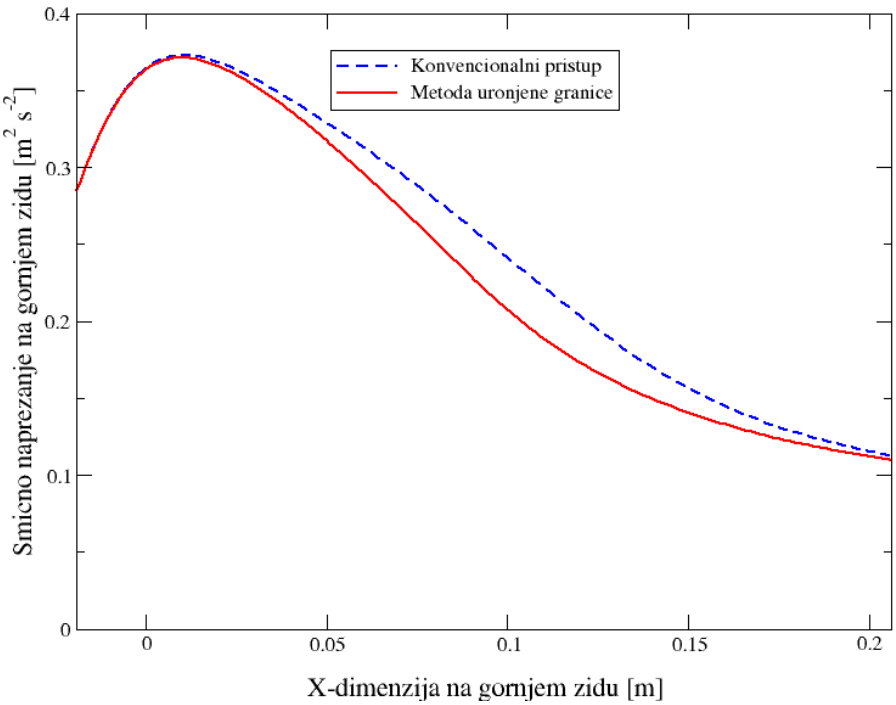
Geometrija cijevi prikazana je na slici 2. Za slučaj cijevi rađene su 4 simulacije: laminarna i turbulentna s mrežom prilagođenom obliku cijevi te laminarna i turbulentna s metodom uronjene granice. Simulacije laminarnog strujanja dale su gotovo identične rezultate za oba načina izrade mreže, a simulacije turbulentnog strujanja daju zadovoljavajuću razinu podudarnosti, što je vidljivo na slikama 3 - 6.



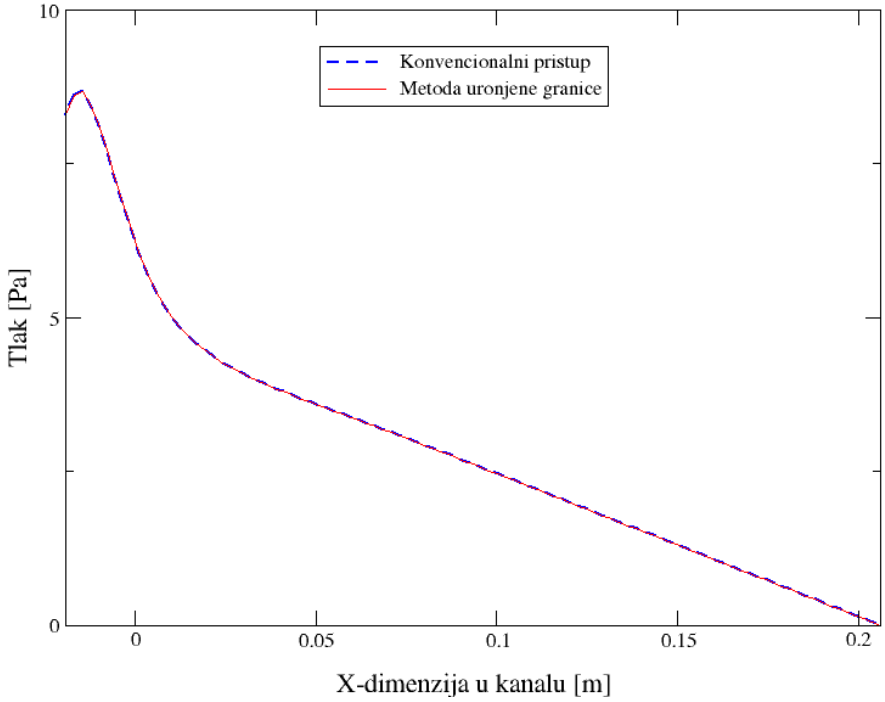
Slika 2: Geometrija cijevi s naglim proširenjem



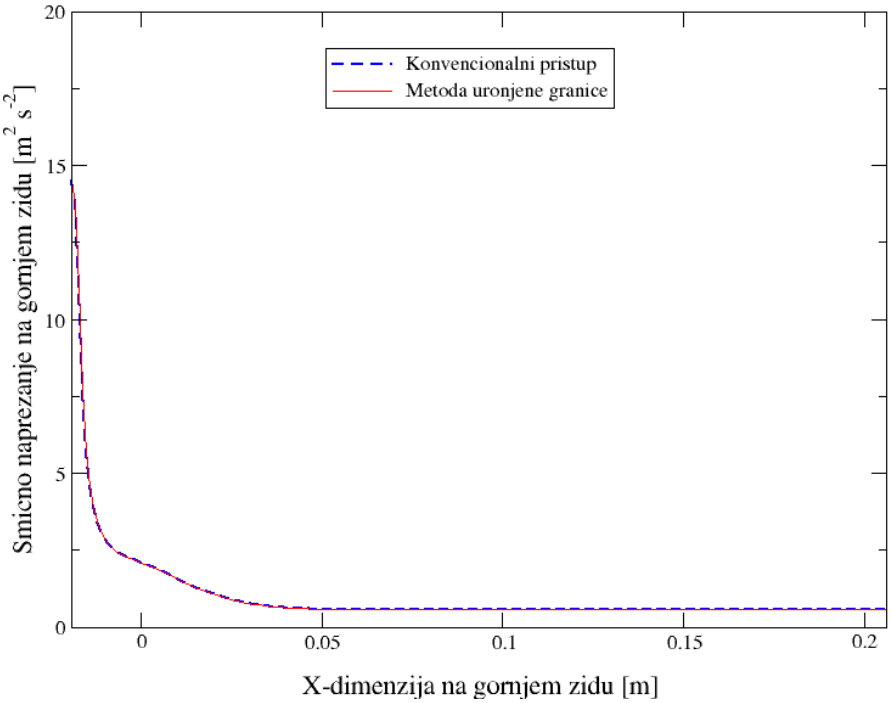
Slika 3: Cijev s naglim proširenjem, usporedba pada tlaka kroz kanal, simulacije turbulentnog strujanja



Slika 4: Cijev s naglim proširenjem, usporedba smičnih naprezanja na gornjem zidu, simulacije turbulentnog strujanja



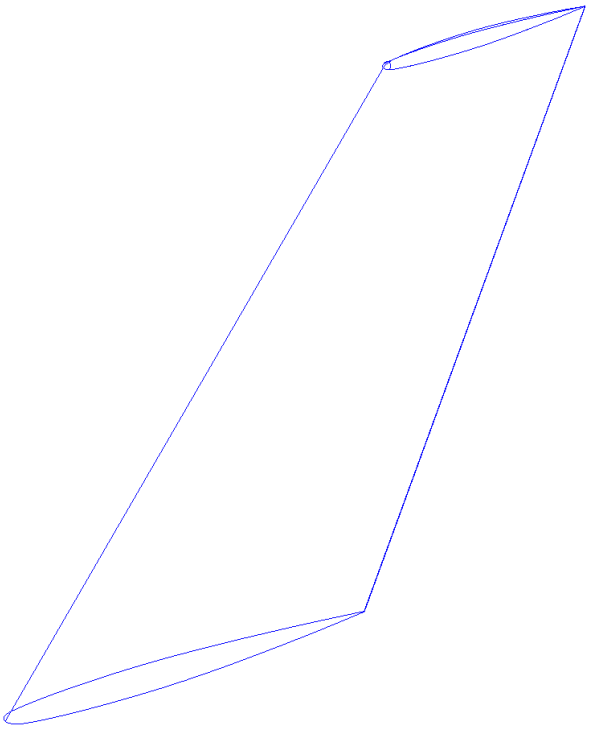
Slika 5: Cijev s naglim proširenjem, usporedba pada tlaka kroz kanal, simulacije laminarnog strujanja



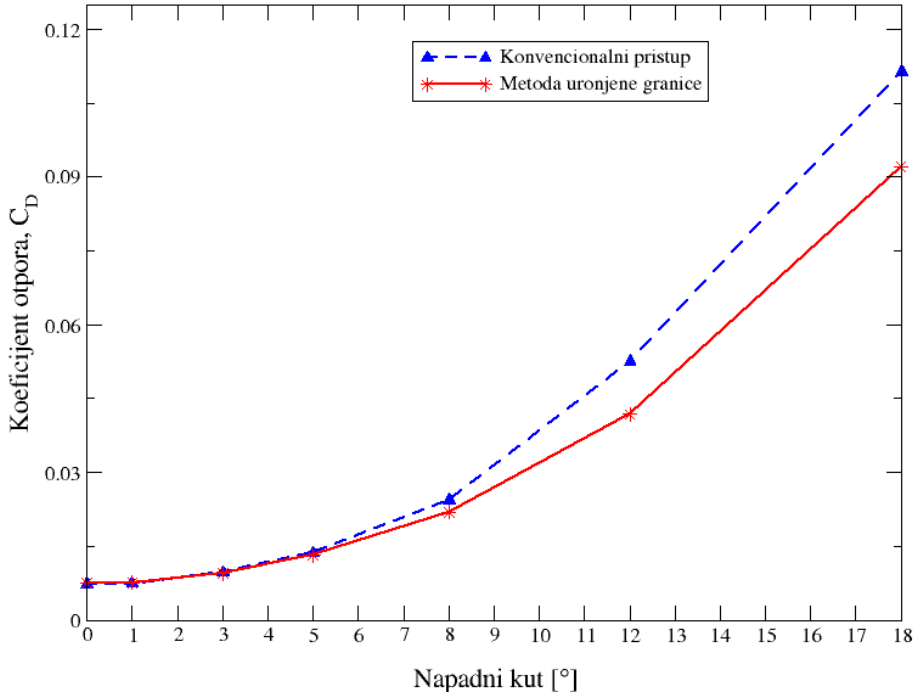
Slika 6: Cijev s naglim proširenjem, usporedba smičnih naprezanja na gornjem zidu, simulacije laminarnog strujanja

Onera M6 krilo

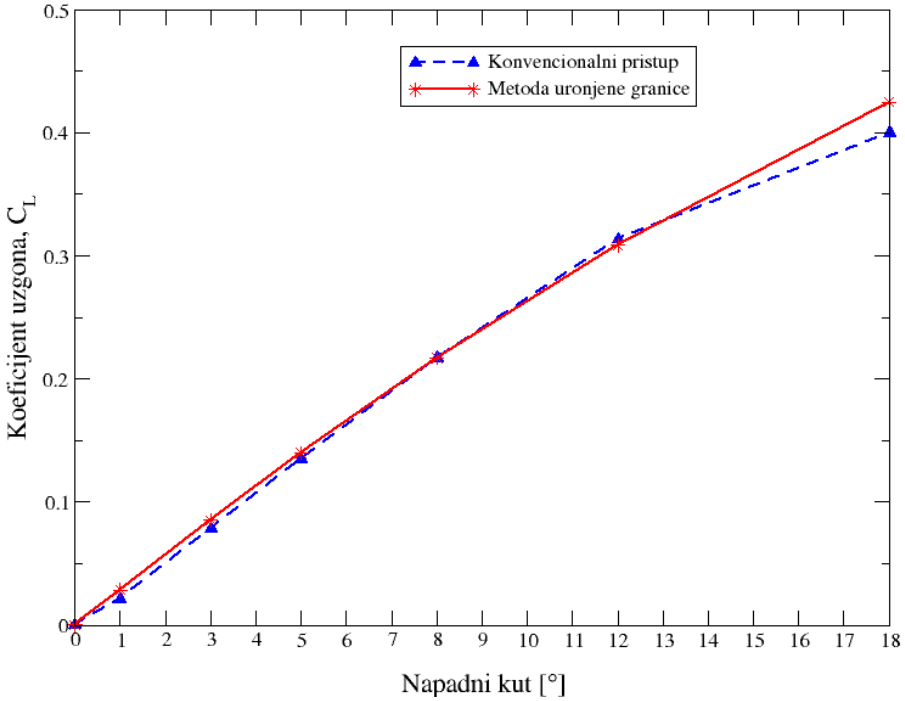
Geometrija Onera M6 krila prikazana je na slici 7. Za slučaj Onera M6 krila provedeno je 14 simulacija, za različite napadne kuteve krila (0, 1, 3, 5, 8, 12 i 18 stupnjeva), a za svaki napadni kut po jedna simulacija s mrežom prilagođenom obliku objekta i po jedna s metodom uronjene granice. Rezultati dobiveni konvencionalnim načinom izrade mreže i metodom uronjene granice daju slične rezultate, a pogotovo pri manjim napadnim kutevima.



Slika 7: Geometrija Onera M6 krila



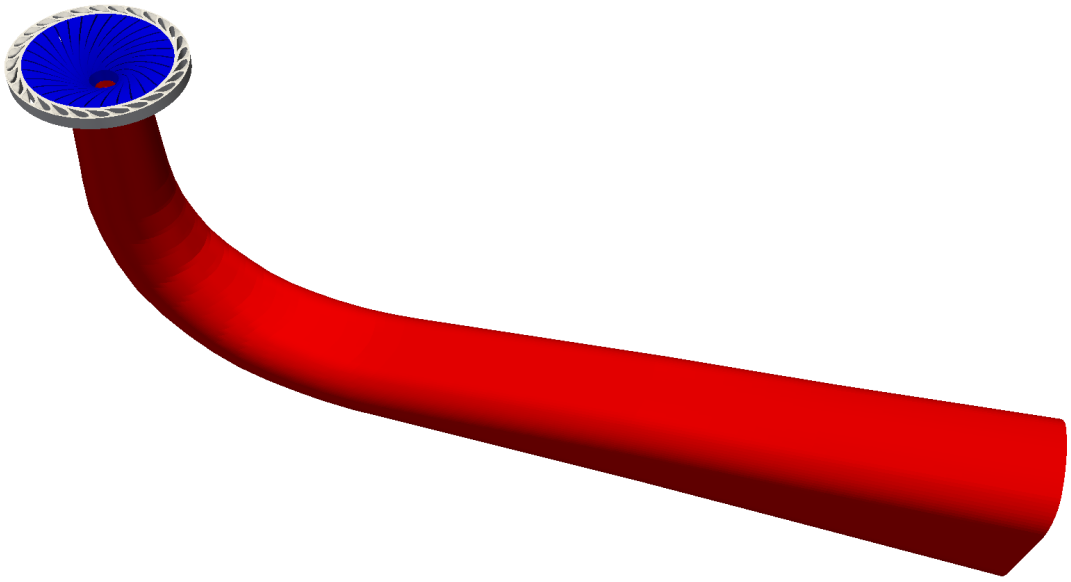
Slika 8: Onera M6 krilo, usporedba koeficijenata otpora za različite napadne kuteve



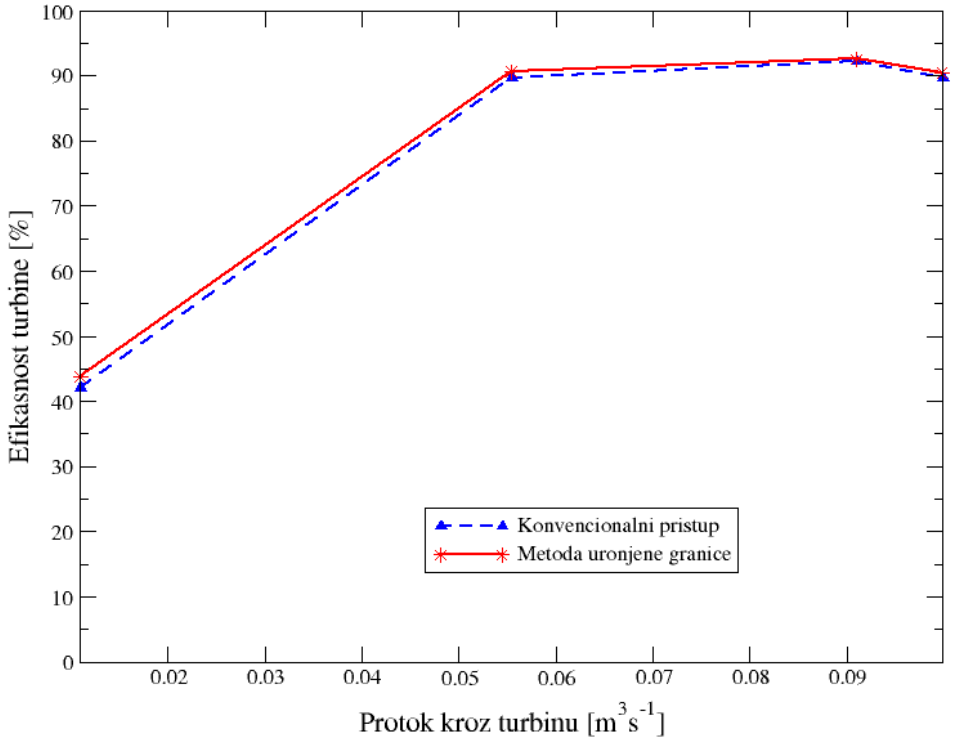
Slika 9: Onera M6 krilo, usporedba koeficijenata uzgona za različite napadne kuteve

Francis-99 turbina

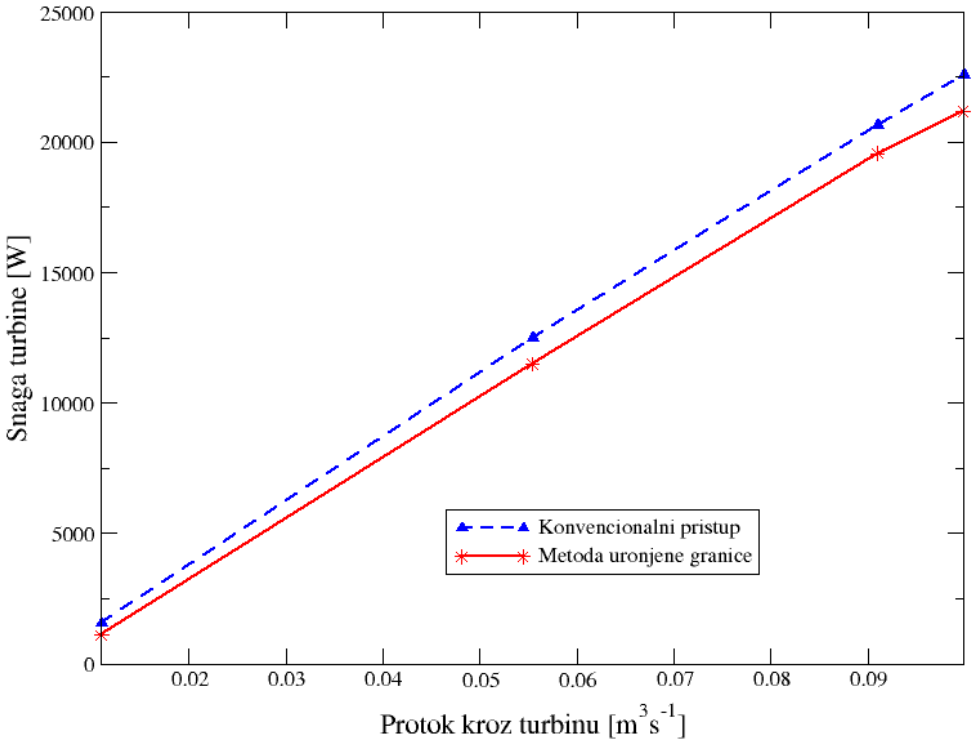
Geometrija modela Francisove turbine prikazana je na slici 10. Provedeno je osam različitih simulacija: za četiri različite pozicije pomičnih statorskih lopatica, po jedna simulacija s metodom uronjene granice i po jedna s mrežom prilagođenom obliku objekta. Rezultati dobiveni konvencionalnim načinom izrade mreže i metodom uronjene granice se podudaraju, osim kod simulacija s najmanjim protokom kroz turbinu, gdje je kod simulacije metodom uronjene granice prisutno značajno odstupanje visine pada energije u turbini.



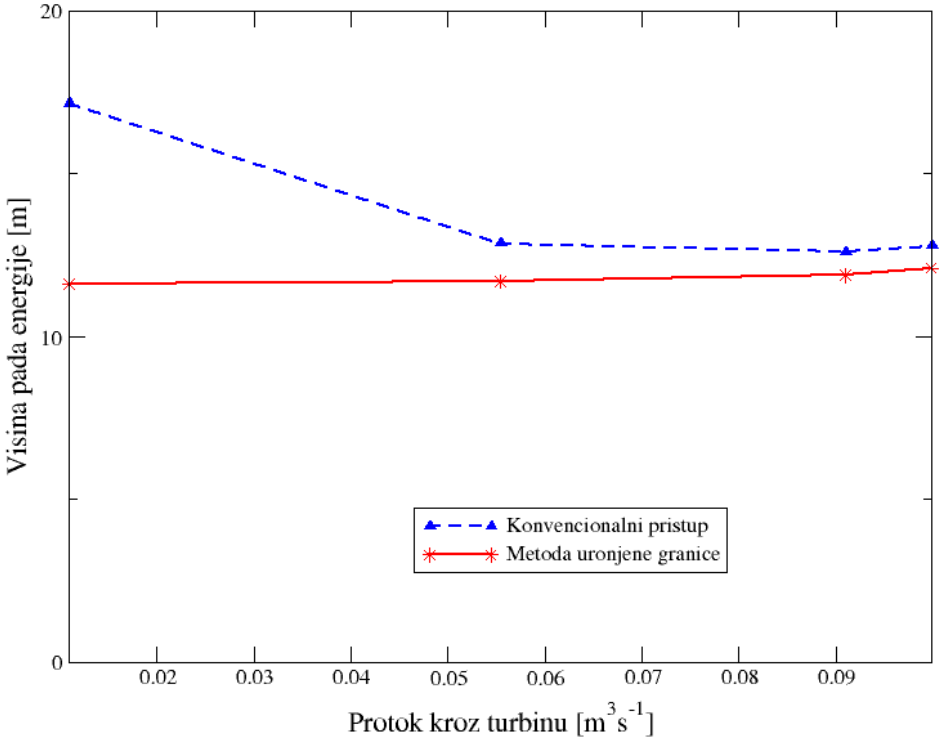
Slika 10: Geometrija Francisove turbine



Slika 11: Francisova turbina, usporedba efikasnosti turbine s obzirom na protok



Slika 12: Francisova turbina, usporedba snage turbine s obzirom na protok



Slika 13: Francisova turbina, usporedba visine pada energije u turbini s obzirom na protok

Zaključak

U ovom radu je predstavljena teorija metode uronjene granice te je njena verzija koja je implementirana u $foam-extend$ 4.1 validirana na trima različitim slučajevima. Rezultati provedenih simulacija pokazuju zadovoljavajuću razinu točnosti.

Kod 2-D cijevi sa naglim proširenjem, simulacije laminarnog strujanja kod kojih je primjenjen konvencionalan način izrade mreže te simulacije kod kojih je primjenjena metoda uronjene granice pokazuju gotovo identične rezultate. Turbulentne simulacije pokazuju blago odstupanje rezultata, npr. razlika u padu tlaka od ulaza do izlaza je 10,4%.

Kod Onera M6 krila, simulacije provedene pomoću metode uronjene granice i konvencionalnim načinom izrade mreže pokazuju vrlo slične rezultate za manje napadne kuteve (do 8 stupnjeva), dok kod većih napadnih kuteva (8 - 18 stupnjeva), razlika rezultirajućih koeficijenata otpora je 15 - 20%, a koeficijenata uzgona oko 5%.

Kod Francisove turbine je simuliran rad turbine pri 4 različita protoka, a sve simulacije, osim za najmanji protok, pokazuju dobru podudarnost među onima provedenima metodom

uronjene granice i onima provedenim konvencionalnim načinom izrade mreže. Razlika u dobivenim učinkovitostima turbine je oko 1%, a u snagama i visinama pada energije u turbini 5 - 9%.

Simulacije provedene u ovom radu su samo maleni dio onoga što se može simulirati računalnom dinamikom fluida. Unatoč tome što su rezultati zadovoljavajući, mora se provesti još validacijskih studija kako bi se metoda uronjene granice smatrala primjenjivom na slučajevima u kojima su prisutni kompresibilnost, izmjena topline, višefazno strujanje itd. Jedna od takvih studija je, također na Fakultetu strojarstva i brodogradnje u Zagrebu, rađena za simulacije strujanja oko broda te se ti rezultati, također zadovoljavajući, mogu naći u [11].

Chapter 1

Introduction

1.1 Background

Computational Fluid Dynamics (CFD) is a tool of increasing importance in research and development of new products and technologies, which is primarily enabled by rapid development of computers and derivation of new numerical methods. The aim of CFD is to numerically solve the equations which describe fluid flow and enable engineers and scientists to obtain various information with the purpose to develop new and enhance existing engineering solutions. Products and equipment which can be developed using CFD are numerous. In the field of energy and power engineering, the most obvious example is turbomachinery, where pumps and turbines are the most interesting examples. Even a small increase of efficiency of turbomachines can bring enormous reductions in the amount of the energy losses in energy transformation processes and the associated economical and environmental costs. For that reason, CFD is an extremely appropriate tool to be utilized in equipment design in the energy sector.

Equations which describe fluid flow are generally well understood, but are fairly complicated to solve. The mission of CFD is to solve these equations, using numerical iterative techniques. The main idea is to discretize these equations in time and space and reduce them to a set of dicretized equations. Then, iterative solvers are used to solve that set composed of a finite number of equations. In this thesis, OpenFOAM was used to accomplish that task.

OpenFOAM [1] is a free, open-source CFD software released in 2004. It has a large user base of users in various fields of science and engineering, both in academia and industry. Its

development is carried out by dedicated OpenFOAM developers, as well as by user contributions. It offers a variety of options for modeling problems in the field of continuum mechanics. Various fluid flow phenomena can be modeled: compressible and incompressible flows, subsonic, transonic and supersonic flows, heat transfer, combustion etc. In this thesis, the $foam-extend$ 4.1 version of OpenFOAM was used. $foam-extend$ [2] is a fork of the OpenFOAM open source library. The goal of this project is to integrate community contributions to the $foam-extend$ simulation toolbox. It is an open project welcoming and integrating contributions from all users and developers.

1.2 Previous and Related Studies

The Immersed Boundary Method (IBM) is a fitting example of a method which has the potential to both decrease the computational effort and the human time needed to set up complex CFD cases. It is a mesh generation method which uses a background mesh combined with a surface file which represents the geometry of a solid inside of the flow field, or of a solid at the borders of the flow field. The meshing process, which is usually the most time consuming part in the process of setting up CFD simulations, can be substantially simplified using IBM compared to body-fitted (BF) meshing. The surface geometry of solids in a simulation can sometimes be very complicated, which also makes the process of body-fitted meshing very complicated. The expected disadvantage of IBM is loss of solution accuracy on the immersed boundaries.

Another aspect in which IBM shows good potential is moving mesh simulations. For body-fitted meshes, the mesh is transformed in ever time step to account for the new position of the solid inside the simulation domain (*e.g.* simulating flow inside an internal combustion engine). With IBM, only the surface of the immersed boundary is moving, while the background mesh remains unchanged, which decreases computational time requirements for dynamic simulations.

The idea of IBM was first introduced by Peskin in [12], where it was used to describe the interaction between a fluid and an elastic boundary inside the flow field. In later studies, a large number of different versions of IBM were proposed. All of those versions differ drastically in terms of simulation goals and mathematical models which are used in deriving them. For this reason, it is impossible to determine a single mathematical formulation of IBM. A review of

different versions is presented in [9] and will be summarised in this thesis in Chapter 4.

1.3 Thesis Outline

The rest of the thesis is organised as follows.

First, the underlying theory of basic fluid flow in CFD will be presented: the governing equations, the process of discretization and solving the dicretized equations. Then, the $foam-extend$ 4.1 version of IBM will be described and compared to previous versions. The final aim is to evaluate and, hopefully, validate the results acquired from simulations in which the geometries were spatially discretized using the $foam-extend$ 4.1 IBM, as opposed to equivalent simulations in which the geometries were meshed using the conventional body-fitted approach. The first simulation is the simplest one: internal flow over a 2-D backward facing step, both with laminar and turbulent conditions. The second simulation is an external flow Onera M6 wing simulation with turbulent conditions. The last and the most complex simulation is a simulation of a Francis turbine with different operating conditions (varying load, turbine power, $etc.$). The final case is very challenging because it involves significant variations of the geometry.

Chapter 2

Mathematical Model

In the previous chapter, a brief introduction into the subject of this thesis was given. In the following chapter, fundamental equations of fluid mechanics will be presented.

2.1 Introduction

The set of equations which will be laid out in the following section is a set of equations which govern fluid flow. They describe a wide array of fluid flow configurations - for example, flows in turbomachinery, external flows surrounding structures like aeroplanes, vehicles or submarines, internal pipe or duct flow, laminar and turbulent flows, single-phase and multi-phase flows, etc.

These equations are essentialy a set of coupled differential equations which are, in practice, too difficult to solve analytically, except for some very basic fluid flow problems which implement a lot of simplifications to the equations. For problems of engineering-level complexity, computers are used to solve approximations of the equations.

2.2 Governing Equations of Fluid Flow

A large number of equations describing fluid flow are actually just versions, more or less complex, of the General Transport Equation (GTE):

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \mathbf{u}) - \nabla \cdot (\gamma \nabla \phi) = q_v. \tag{2.2.1}$$

In the GTE, ϕ is a generic transported variable, $\mathbf{u}$ is the convective velocity, γ is the diffusion coefficient and q_v is a volume source or sink of the transported variable ϕ.

The GTE consists of four terms:

- Temporal derivative $\frac{\partial \phi}{\partial t}$, which represents the inertia of the system;
- Convection term $\nabla \cdot (\phi \mathbf{u})$, which represents the convective transport by the prescribed velocity field. The term is of hyperbolic nature;
- Diffusion term $\nabla \cdot (\gamma \nabla \phi)$, which represents gradient transport. The term is of elliptic nature;
- Sources and sinks q_v account for non-transport effects: local volume production and destruction of the transported variable ϕ;

By substituting the transported variable ϕ in the GTE with an appropriate property (mass, momentum, energy, etc.), the fundamental governing equations of fluid flow can be derived.

2.2.1 Conservation of Mass

The mass conservation equation can be derived from the GTE and in its general form is:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \tag{2.2.2}$$

Here, fluid density ρ substitutes the general variable ϕ and becomes the transported variable. As there is no diffusion of mass, no diffusion term exists in the equation. Likewise, there is also no source or sink term, as mass is assumed to be impossible to vanish or be generated.

2.2.2 Conservation of Linear Momentum

The linear momentum conservation equation can be derived from the GTE and in its general form is:

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \rho \mathbf{g} + \nabla \cdot \sigma. \tag{2.2.3}$$

Here, momentum $\rho \mathbf{u}$ becomes the transported variable. The first term is the inertial term, the second term is the convective term, the first term on the right hand side $(\rho \mathbf{g})$ is a source term

which describes the effect of gravitational forces on the fluid and the last term $(\nabla \cdot \sigma)$ describes the surface forces in the fluid. The surface forces are a sum of forces produced as an effect of the existence of a pressure gradient and by the viscous stresses in the fluid.

2.2.3 Conservation of Energy

The energy conservation equation can be derived from the GTE and in its general form is:

$$\frac{\partial(\rho e)}{\partial t} + \nabla \cdot (\rho e \mathbf{u}) = \rho \mathbf{g} \cdot \mathbf{u} + \nabla \cdot (\sigma \cdot \mathbf{u}) - \nabla \cdot \mathbf{q} + \rho Q. \tag{2.2.4}$$

To obtain this equation, energy content in the fluid ρe is input into the GTE as a transported variable. The first term is the one which describes the temporal variation of energy in the fluid. The second term is the convective term and it determines the convective flux of energy in the fluid. The first term on the right-hand hand is the gravitational force term. The second term describes the rate of work due to the existence of surface forces. The third term on the right-hand side is the diffusive term and it describes the heat flux due to the existence of a temperature gradient. The last term is the volume energy source term.

2.2.4 Incompressible Flows

It is possible to implement some simplifications for incompressible flow, in which the density of the fluid is assumed to be constant, which can substantially ease the process of solving the governing equations of fluid flow. Compressibility is often a term associated with gases. In practice, both gas and liquid flows can be either compressible or incompressible. The effect compressibility has on the flow is actually dependent on the conditions of the flow, as well as the properties of the fluid. Usually, flow is considered incompressible if the peak Mach number of the flow is up to 0.3. Mathematically, the Mach number is:

$$Ma = \frac{|\mathbf{u}|}{a}, \tag{2.2.5}$$

where a is the speed of sound in the fluid.

In practice, a variety of flows in the field of engineering are incompressible due to being limited to Mach numbers of less than 0.3. In all cases in this thesis, Mach numbers are small and therefore, the flows are assumed to be incompressible.

By definition, incompressibility implies constant density ($\rho = const.$), which mathematically manifests in the governing equations in the following way:

- Conservation of mass transforms into the continuity equation:

$$\nabla \cdot \mathbf{u} = 0. \tag{2.2.6}$$

- Conservation of linear momentum transforms into the following:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = \mathbf{g} - \nabla p + \nabla \cdot (\nu \nabla \mathbf{u}), \tag{2.2.7}$$

  where ν is the kinematic viscosity.

- Conservation of energy equation becomes decoupled from the rest of the governing equations and no longer plays a part in determining the flow field.

2.3 Turbulence Modeling

The use of turbulence models is required in simulations involving turbulent flow behaviour, which includes the majority of engineering applications, and nature in general. Turbulence occurs at high values of the Reynolds number *Re*:

$$Re = \frac{\rho l |\mathbf{u}|}{\mu}, \tag{2.3.1}$$

where l is the characteristic length scale of the flow configuration (*e.g.* pipe diameter for internal pipe flow), $|\mathbf{u}|$ is the velocity magnitude and μ is the dynamic viscosity of the fluid.

Turbulence is a transient, irregular, three-dimensional, non-linear and stochastic phenomenon. Turbulent flow is characterised by chaotic behaviour, high diffusivity and high energy dissipation. As the nature of turbulence makes turbulent flow equations very hard to solve, although Direct Numerical Simulation (DNS) is possible, utilization of turbulence models is required. DNS is simply too troublesome for any practical use, as it has a high demand of computational resources. Two other main approaches, other than DNS, are Large Eddy Simulation (LES) and

Reynolds-Averaged Navier-Stokes Equations (RANS). The former is generally less computationaly demanding than DNS, but more so than RANS. RANS is exclusively used in the cases analyzed in this thesis, and will be described in the following subsection.

2.3.1 Reynolds-Averaged Navier-Stokes Equations

The main presumption of RANS models is that in turbulent flows, although various flow values (pressure, velocity, etc.) are of stochastic nature, they vary about a mean value.

Mathematicaly, if ϕ is is the value of a flow variable, at the position $\mathbf{x}$ and at time t, ϕ can be decomposed into a mean component $\overline{\phi}$ and a fluctuating component ϕ' [3]:

$$\phi(\mathbf{x},t) = \overline{\phi}(\mathbf{x},t) + \phi'(\mathbf{x},t). \tag{2.3.2}$$

For steady turbulent flows, the mean value $\overline{\phi}$ (which is the value of interest for engineers) is computed by Reynolds time averaging. Time averaging produces the average of a quantity over a time interval. If T is the interval over which averaging is performed, then $\overline{\phi}$ (which is only location-dependent) is computed as:

$$\overline{\phi}(\mathbf{x}) = \lim_{T\to\infty} \frac{1}{T} \int_t^{t+T} \phi(\mathbf{x},t)dt. \tag{2.3.3}$$

Velocity and pressure fields are decomposed into:

$$\mathbf{u} = \overline{\mathbf{u}} + \mathbf{u}', \tag{2.3.4}$$

$$p = \overline{p} + p', \tag{2.3.5}$$

$$\overline{\mathbf{u}} = \overline{u_x}\mathbf{i} + \overline{u_y}\mathbf{j} + \overline{u_z}\mathbf{k}, \tag{2.3.6}$$

$$\mathbf{u}' = u'_x\mathbf{i} + u'_y\mathbf{j} + u'_z\mathbf{k}, \tag{2.3.7}$$

Using equations 2.3.4 - 2.3.7, the Reynolds averaged form of governing equations is:

$$\nabla \cdot (\rho\overline{\mathbf{u}}) = 0, \tag{2.3.8}$$

$$\frac{\partial(\rho\overline{\mathbf{u}})}{\partial t} + \nabla\cdot(\rho\overline{\mathbf{u}}\overline{\mathbf{u}}) = -\nabla\overline{p} + \left[\nabla\cdot(\overline{\tau} - \rho\overline{\mathbf{u}'\mathbf{u}'})\right] + \overline{\mathbf{f}}_b. \tag{2.3.9}$$

The above Reynolds averaged equations are quite similar to the original governing equations, with the exception of the term $-\rho\overline{\mathbf{u}'\mathbf{u}'}$, known as the Reynolds stress tensor τ^R. It is a symmetric tensor and it introduces six new unknowns into the governing equations set. Any linear averaging of the equations, like the Reynolds averaging techniques, cannot reduce the order of the problem. Also, adding equations to solve for the new unknows produces additional unknowns. To overcome this problem, a turbulence model needs to close the system of equations by expressing the non-linear fluctuating stress components only in terms of mean components.

The modeling of the Reynolds stress tensor is based on the Boussinesq hypothesis:

$$\tau^R = -\rho\overline{\mathbf{u}'\mathbf{u}'} = \mu_t\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right] - \frac{2}{3}\rho k\mathbf{I}, \tag{2.3.10}$$

where k is the turbulence kinetic energy:

$$k = \frac{1}{2}\overline{\mathbf{u}'\mathbf{u}'}. \tag{2.3.11}$$

μ_t is the turbulence eddy viscosity which is, unlike molecular viscosity μ, flow-dependent and not fluid-dependent. With the Boussinesq hypothesis, the problem of calculating the Reynolds stress tensor components is transformed into a problem of calculating the turbulence kinetic energy and turbulent viscosity.

Following the Boussinesq hypothesis, several groups of turbulence models have been developed:

- Algebraic models
- One-equation models
- Two-equation models
- Second-order closure models

Each group has advantages and shortcomings.

Two-equation turbulence models are the most popular in industrial applications. They require the solution of two transport equations to model the Reynolds stress tensor. The most

popular are the $k-\varepsilon$ and $k-\omega$ groups of models. The $k-\omega$ *SST* model was exclusively used in this thesis and will be described in the following section. For further details about turbulence modeling, see [3].

The $k-\omega$ *SST* Model

The Shear Stress Transport $k-\omega$ model is a zonal two-equation turbulence model. The two equations are the turbulence kinetic energy equation (k) and the eddy turnover time equation (ω). In this model, the k - ω formulation is used in the inner parts of the boundary layer. Away from the inner parts of the boundary layer, the model switches to the k - ε formulation using a blending function. In this way, good behaviour in adverse pressure gradients and separating flow is achieved. Also, the heavy dependency on freestream values, a characteristic of the standard $k-\omega$ model, is reduced. The implementation of $k-\omega$ *SST* which is used in OpenFOAM is presented in [4], with coefficients updates from [13], and it is mathematically formulated in the following way:

- Transport equation for the turbulence kinetic energy:

$$\frac{\partial(\rho k)}{\partial t}+\nabla\cdot(\rho\overline{\mathbf{u}}k)=\widetilde{P}_k-\beta^*\rho\omega k+\nabla\cdot(\Gamma_k\nabla k). \tag{2.3.12}$$

- Transport equation for the eddy turnover time:

$$\frac{\partial(\rho\omega)}{\partial t}+\nabla\cdot(\rho\overline{\mathbf{u}}\omega)=\frac{\gamma}{\nu_t}P_k-\beta\rho\omega^2+\nabla\cdot(\Gamma_k\nabla\omega)+(1-F_1)2\rho\sigma_{\omega2}\frac{1}{\omega}(\nabla k)(\nabla\omega). \tag{2.3.13}$$

- First set of additional relations:

$$\Gamma_k=\mu+\frac{\mu_t}{\sigma_k}, \tag{2.3.14}$$

$$\Gamma_\omega=\mu+\frac{\mu_t}{\sigma_\omega}, \tag{2.3.15}$$

$$P_k=\tau\nabla\cdot\overline{\mathbf{u}}, \tag{2.3.16}$$

$$\widetilde{P_k} = min(P_k; c_1\varepsilon), \tag{2.3.17}$$

$$\mu_t = \rho \frac{a_1 k}{max(a_1\omega; SF_2)}. \tag{2.3.18}$$

- Model coefficients:

$$\sigma_{k1} = 0.85,\ \sigma_{k2} = 1.0,\ \sigma_{\omega 1} = 0.5,\ \sigma_{\omega 2} = 0.856,\ \beta_1 = 0.075,\ \beta_2 = 0.0828,$$

$$\beta^* = 0.09,\ \gamma_1 = 0.5532,\ \gamma_2 = 0.44,\ a_1 = 0.31,\ b_1 = 1.0,\ c_1 = 10.0.$$

Coefficients with index 1 come from the $k-\omega$ model, and the ones with index 2 belong to the $k-\varepsilon$ model. The coefficients φ used in equations depend on F_1 and are obtained using the following formula:

$$\varphi = F_1\varphi_1 + (1-F_1)\varphi_2, \tag{2.3.19}$$

where φ_1 and φ_2 stand for the coefficients of the $k-\omega$ and the $k-\varepsilon$ model.

- Second set of additional relations:

$$F_1 = tanh\left(arg_1^4\right);\ arg_1 = min\left(max\left(\frac{\sqrt{k}}{\beta^*\omega y}; \frac{500\nu}{y^2\omega}\right); \frac{4\rho\sigma_{\omega 2}k}{CD_{k\omega}y^2}\right), \tag{2.3.20}$$

$$CD_{k\omega} = max\left(2\rho\sigma_{\omega 2}\frac{1}{\omega}(\nabla k)(\nabla\omega); 1.0e^{-10}\right), \tag{2.3.21}$$

$$F_2 = tanh\left(arg_2^2\right);\ arg_2 = max\left(\frac{\sqrt{k}}{\beta^*\omega y}; \frac{500\nu}{y^2\omega}\right), \tag{2.3.22}$$

$$\tau = \mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\right) - \frac{2}{3}\rho k\delta_{ij}. \tag{2.3.23}$$

Chapter 3

Finite Volume Method

In the previous chapter, the governing equations were presented. In the following chapter, the numerical methods used to solve those equations will be presented.

3.1 Discretization of the Governing Equations

The goal of discretization of equations is to obtain a finite number of governing equations, which describe the spatialy discretized domain in a temporaly discretized simulation time. The final equation after discretization for a single cell has the following form:

$$a_P \phi_P + \sum_N a_N \phi_N = b, \tag{3.1.1}$$

where a_P is the diagonal contribution to the solution matrix, a_N is the off-diagonal contribution, b holds the right-hand side contributions and ϕ_P and ϕ_N are the field values in the current and neighbouring cells.

The discretization procedure shown here is taken from [5], where it is described in detail.

The GTE was shown in 2.2.1:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \mathbf{u}) - \nabla \cdot (\gamma \nabla \phi) = q_v.$$

The GTE is a second-order partial differential equation and if good accuracy is to be preserved, it is necessary for the order of the discretization to be of equal or higher order than the equation

that is being discretized. The terms of the transport equation will be treated separately (temporal derivative, convection term, diffusion term and source term).

In order to obtain a second-order accuracy, the assumed variation of the function $\phi = \phi(\mathbf{x}, t)$ in space and time around the point P (the center of a finite volume) must be linear in space and time:

$$\phi(\mathbf{x}) = \phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P, \tag{3.1.2}$$

$$\phi(t + \Delta t) = \phi^t + \Delta t \left(\frac{\partial \phi}{\partial t}\right)^t, \tag{3.1.3}$$

where

$$\phi_P = \phi(\mathbf{x}_P), \tag{3.1.4}$$

$$\phi_t = \phi(t). \tag{3.1.5}$$

3.1.1 Discretization of the Spatial Terms

The Gauss' theorem is often used in the discretization procedure:

$$\int_V \nabla \cdot \mathbf{a} \, dV = \oint_{\partial V} d\mathbf{S} \cdot \mathbf{a}, \tag{3.1.6}$$

where ∂V is the closed surface bounding the volume V and dS is an infinitesimal surface element with associated outward pointing normal on ∂V.

It can be shown that:

$$\int_{V_P} \phi(\mathbf{x}) dV = \phi_P V_P, \tag{3.1.7}$$

where V_P is the volume of the cell. A second order accurate discretized form of the Gauss' theorem is:

$$(\nabla \cdot \mathbf{a}) V_P = \sum_f \mathbf{S} \cdot \mathbf{a}_f, \tag{3.1.8}$$

where the subscript f implies the value of the variable in the middle of the face and $\mathbf{S}$ is the outward-pointing face area vector.

Convection Term

The discretisation of the convection term is obtained using equation 3.1.8:

$$\int_{V_P} \nabla \cdot (\rho \mathbf{u} \phi) dV = \sum_f \mathbf{S} \cdot (\rho \mathbf{u})_f \phi_f = \sum_f F \phi_f, \tag{3.1.9}$$

where F represents the mass flux through the face. For equation 3.1.9, F and the face values (subscript f) are required. The calculation of F is given in [5] and the face values are calculated from the values in the cell centers, which are obtained using the convection differencing scheme.

The role of the convection differencing scheme is to determine the value of ϕ on the face from the values in the cell centers. Some of them are:

- Central Differencing (second-order accurate but oscillatory)
- Upwind Differencing (first-order accurate but bounded, nonoscillatory)
- Blended Differencing (both boundedness and accuracy of the solution is attempted to be preserved as good as possible using a blending factor)

In choosing the right convection scheme, a compromise between accuracy and boundedness of the solution is sought.

Diffusion Term

Using the assumption of linear variation of ϕ and equation 3.1.8 it follows:

$$\int_{V_P} \nabla \cdot (\rho \gamma \nabla \phi) dV = \sum_f (\rho \gamma)_f \mathbf{S} \cdot (\nabla \phi)_f. \tag{3.1.10}$$

If the mesh is orthogonal, *i.e.* vectors $\mathbf{d}$ and $\mathbf{S}$ in 3.1.1 are parallel, it is possible to use the expression:

$$\mathbf{S} \cdot (\nabla \phi)_f = |\mathbf{S}| \frac{\phi_N - \phi_P}{\mathbf{d}}. \tag{3.1.11}$$

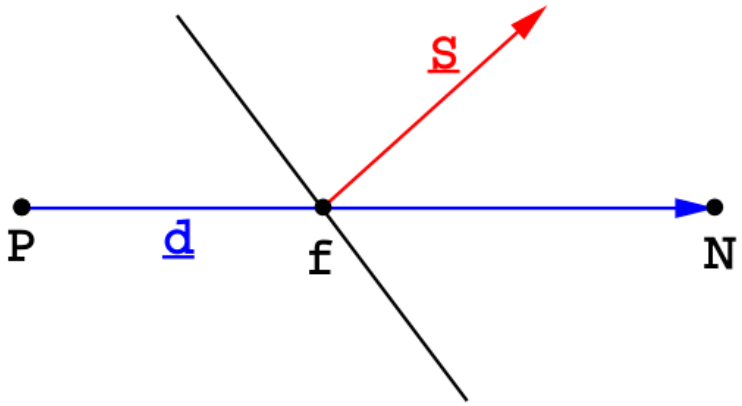


Figure 3.1.1: Vectors **d** and **S** on a non-orthogonal mesh [5]

Using the equation 3.1.11, the face gradient of ϕ can be calculated from the two values around the face. Alternatively, the cell-centered gradient could be calculated using the values of ϕ in neighbouring face centers (ϕ_f) and then interpolate the cell-centered gradient to the face.

In practice, meshes are rarely orthogonal. For that reason, non-orthogonality correction is utilized. It can potentially create unboundendness, particulary if the non-orthogonality is high. If the preservation of boundendness is more important than accuracy, the non-orthogonal correction has got to be limited or completely discarded. The non-orthogonality correction is further described in [5].

The final form of the discretised diffusion term is:

$$\mathbf{S} \cdot (\nabla\phi)_f = |\Delta| \frac{\phi_N - \phi_P}{\mathbf{d}} + \mathbf{k} \cdot (\nabla\phi)_f, \tag{3.1.12}$$

where $\mathbf{k} \cdot (\nabla\phi)_f$ is the non-orthogonal correction.

Source Terms

All terms that cannot be written as convection, diffusion or temporal terms are treated as sources. The source term, $q_v(\phi)$ can be a general function of ϕ. The linearized form of the source term is:

$$q_v(\phi) = q_u + q_p\phi, \tag{3.1.13}$$

where q_u and q_p can also depend on ϕ. The volume integral is:

$$\int_{V_P} q_\phi(\phi) dV = q_u V_P + q_p V_P \phi_P. \tag{3.1.14}$$

3.1.2 Temporal Discretization

In the previous section, the dicretization of the spatial terms has been presented. Temporal discretization is of great importance for transient simulations. Since only steady state simulations were performed in this thesis, temporal discretization will not be shown, however more details can be found in [5].

3.2 Solving the Discretized Equations

After discretizing the governing partial differential equations, a system of linear equations of the form

$$[A][\phi] = [b], \tag{3.2.1}$$

is to be solved [3]. Here, $[A]$ is a matrix constituted of the coefficients of the unknown variables. $[\phi]$ are the unknowns, located at the centroids of the mesh elements and are the values which are sought after. The vector $[b]$ contains all sources, constants, boundary conditions and non-linearizable coefficients. In an extended form, the system of equations is:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N-1} & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N-1} & a_{2N} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN-1} & a_{NN} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \vdots \\ \phi_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_N \end{bmatrix}, \tag{3.2.2}$$

where a_{ij}, ϕ_i and b_i are components of matrix $[A]$ and vectors $[\phi]$ and $[b]$, respectively. Generally, each row in the above equation represents an equation defined over one element of the computational domain, and the non-zero coefficients a_{ij} are those which describe the relation of the element with the neighbouring elements, or to the values present in the element itself.

The methods for solving linear systems of equations are numerous, and are generally divided into direct and iterative. Direct methods are rarely used in CFD applications. Iterative techniques are preffered for their lower memory and computational cost requirements.

In a direct method, $[A]$ is inverted and the solution $[\phi]$ is computed as $[\phi] = [A]^{-1}[b]$. If $[A]$ is large, finding its inverse is computationally very challenging. Meshes for CFD applications tend to have large numbers of cells, which leads to a large matrix $[A]$. Also, $[A]$ is usually a sparse matrix, which additionally complicates the procedure of computing its inverse.

When using iterative linear solvers the solution algorithm is iteratively applied until a required level of convergence is reached. The strategy of all iterative linear solvers is similar: a series of solutions $[\phi]^{(n)}$ is computed, which ideally converges to the exact solution $[\phi]$. To start the solving process, an initial value is chosen $[\phi]^{(0)}$ and an iterative procedure that computes $[\phi]^{(n)}$ in each iteration from the previously calculated solution $[\phi]^{(n-1)}$ is performed.

3.3 The SIMPLE algorithm

The SIMPLE (Semi-Implicit Method for Pressure Linked Equations) algorithm is a pressure-based segregated iterative method used for solving steady state flow problems. It was first mentioned in [6] and has been widely used since. The SIMPLE algorithm is used in the simulations in this thesis, in the simpleFoam and the MRFSimpleFoam solvers which are available in OpenFOAM.

The steps for solving the discretized governing equations are:

- Guess the initial conditions for all field variables.
- Solve the momentum equation for velocity (momentum predictor step):

$$\mathbf{u}_P = (a_P^{\mathbf{u}})^{-1} \left[H(\mathbf{u}) - \nabla p \right], \tag{3.3.1}$$

where

$$H(\mathbf{u}) = \mathbf{r} - \sum_N a_N^{\mathbf{u}} \mathbf{u}_N, \tag{3.3.2}$$

where $\mathbf{r}$ is the right-hand side contribution in the momentum equation.

- Using the obtained velocity, solve the incompressible continuity equation for pressure (pressure correction step):

$$\nabla \cdot \left[(a_P^{\mathbf{u}})^{-1} \nabla p \right] = \nabla \cdot \left[(a_P^{\mathbf{u}})^{-1} H(\mathbf{u}) \right]. \tag{3.3.3}$$

- Based on the calculated pressure field, assemble the conservative face flux F:

$$F = \mathbf{S} \cdot H(\mathbf{u}) - a_N^p (p_N - p_P). \tag{3.3.4}$$

- Solve other transport equations, if there are any (turbulence, *etc*.)

- Repeat until convergence the criterion is met

Under-relaxation is needed in this procedure, with usual under-relaxations factors of $\alpha_p = 0.2$ and $\alpha_{\mathbf{u}} = 0.8$, which can be different depending on the case.

3.4 Additional Features

CFD is a very broad field in a sense that it is used to solve problems which can be of quite differing nature (by criteria of compressibility, time-dependency, existence of combustion *etc*.). Apart from numerous techniques for solving dicretized governing equations, there exist some case-specific modeling and numerical techniques which can be applied to some problems. Some of those additional features are used in this thesis and will be described in this section.

General Grid Interface (GGI)

General Grid Interface (GGI) is a coupling interface used for joining multiple non-conformal regions where the patch nodes on each side of the interface do not match [7]. Examples of coupling interfaces present in OpenFOAM that are built to join conformal mesh regions are processor, regionCouple and cyclic. In the case of conformal mesh regions, the patch nodes on each sides of the interface coincide with each other.

Making multiple non-conformal regions has certain advantages over a completely conformal multi-region mesh. In the case of complicated 3-D geometries, the attention that needs to

be given in the meshing process in order to preserve the conformity can be substantial. Hydroturbines are a good example and one of them, a Francis-type turbine, will be analyzed as the third validation case in this thesis. In the case of the Francis turbine, the mesh was split into three regions which were meshed seperately. After the meshing process, the regions are coupled with GGI patches, allowing the rotor region to rotate relative to the static upstream and downstream regions. Some specialized versions of GGI exist, which are needed in order to simplify the mesh complexity of turbomachinery simulations and reduce the computational time needed to run the simulations, *i.e.* cyclicGGI, which was also used in the Francis turbine case.

GGI is using weighted interpolation to evaluate and send a value across a pair of conformal or non-conformal coupled patches. No re-meshing of the neighbouring cells of the interface is required. The equations that control the flow values between the GGI master patch and GGI shadow patch are derived in line with the basic FVM discretization reasoning. They state that consistent and conservative discretization across the interface is achieved using weighted interpolation of the following form:

- For the flow values or variables from the master patch to the slave patch

$$\phi_{Si} = \sum_n W_{M_n toS_i} \cdot \phi_{Mn}. \tag{3.4.1}$$

- For the flow values or variables from the slave patch to the master patch

$$\phi_{Mj} = \sum_m W_{S_m toM_j} \cdot \phi_{Sm}. \tag{3.4.2}$$

- In order for the interface discretisation to remain conservative, we have the following three constraints:

$$\sum_n W_{M_n toS_i} = 1.0, \tag{3.4.3}$$

$$\sum_n W_{S_m toM_j} = 1.0, \tag{3.4.4}$$

$$W_{M_n toS_i} \cdot |S_{M_n}| = W_{S_m toM_j} \cdot |S_{S_m}| = |S_{\cap_{MtoS}}|, \tag{3.4.5}$$

with the additional symmetry constraint:

$$\text{if } W_{M_n toS_i} > 0, \text{ then } W_{S_m toM_j} > 0, \tag{3.4.6}$$

but in general:

$$W_{M_n toS_i} \neq W_{S_m toM_j}. \tag{3.4.7}$$

ϕ_S is a shadow patch variable, ϕ_M is a master patch variable, i is an ith shadow patch face, j is a jth master patch face, n is the number of master face neighbours for shadow patch i, m is the number of shadow face neighbours for master patch j, W_{MtoS} is the master facet to shadow facets weighting factor, W_{StoM} is the shadow facet to master facets weighting factor, $|S_M|$ is the surface area of the master facet, $|S_M|$ is the surface area of the shadow facet and $|S_{\cap_{MtoS}}|$ is the intersection area between master and shadow facets.

The value of the GGI weighting factors can be expressed from equation 3.4.5 which is based on facet area values and polygonal intersection.

For the master-to-shadow patch faces:

$$W_{MtoS_i} = \frac{|S_{\cap_{MtoS_i}}|}{S_{M_n}}, \;\; with \;\; W_{MtoS_i} \;\; \in \;\; [0.0,\; 1.0]. \tag{3.4.8}$$

For the shadow-to-master patch faces:

$$W_{StoM_j} = \frac{|S_{\cap_{StoM_j}}|}{S_{S_m}}, \;\; with \;\; W_{StoM_j} \;\; \in \;\; [0.0,\; 1.0]. \tag{3.4.9}$$

where $|S_{\cap_{MtoS}}|$ and $|S_{\cap_{StoM}}|$ are surface intersection areas between a master and shadow patch faces, S_M and S_S are surface areas of a master and shadow patch faces, i is the ith shadow patch face for a given master patch face and j is the jth master patch face for a given shadow patch face.

The GGI weighting factors are essentially percentages of surface intersections between two overlapping faces.

Multiple Reference Frame

Multiple Reference Frame (MRF) [8] is an approach for handling rotation in turbomachinery simulations. In MRF simulations, the mesh is static and the rotation is accounted for in a region-wise manner. Momentum equation may be formulated either in terms of absolute or relative velocity; the latter requires a transformation of the velocity field at the rotor-stator interface, which complicates the model. This is also called the *frozen rotor* technique, where a section of a stator and a rotor are simulated in a prescribed relative position and in steady state. For increased fidelity, a frozen rotor simulation may be repeated several times, varying the relative position of components. In simulations of steady flow in single passages at high mesh resolution, cyclic boundaries are used. Imposing cyclicity of a mesh structure is both tedious and limiting in terms of quality. A cyclic GGI interface will be used in this thesis to establish accurate and implicit handling of non-matching periodic boundaries.

For every rotating cell zone, the incompressible Navier-Stokes equations have to be modifed to take into account the rotation of the zone.

The following can be expressed [14] for the position vector $\mathbf{r}$ and velocity $\mathbf{u}$:

$$\left[\frac{d\mathbf{r}}{dt}\right]_I = \left[\frac{d\mathbf{r}}{dt}\right]_R + \omega_r \times \mathbf{r}, \tag{3.4.10}$$

$$\mathbf{u} = \mathbf{u}_R + \omega_r \times \mathbf{r}, \tag{3.4.11}$$

where ω_r is the angular velocity vector. The expression for acceleration in an inertial frame of reference is:

$$\left[\frac{d\mathbf{u}_I}{dt}\right]_I = \left[\frac{d\mathbf{u}_R}{dt}\right]_R + \frac{d\omega_r}{dt} \times \mathbf{r} + 2\omega_r \times \mathbf{u}_R + \omega_r \times \omega_r \times \mathbf{r}, \tag{3.4.12}$$

where the second term on the right-hand side is the tangential acceleration, the third term is the Coriolis acceleration and the last term is the centrifugal acceleration. When equations 2.2.6 and 2.2.7 (which are in an inertial frame of reference) are adjusted for the relative reference frame, with relative velocity, the following equations are obtained:

$$\frac{\partial \mathbf{u}_R}{\partial t} + \frac{d\omega_r}{dt} \times \mathbf{r} + \nabla \cdot (\mathbf{u}_R \times \mathbf{u}_R) + 2\omega_r \times \mathbf{u}_R + \omega_r \times \omega_r \times \mathbf{r} = -\nabla \left(\frac{p}{\rho}\right) + \nabla \cdot (\nu \nabla \mathbf{u}_R), \tag{3.4.13}$$

$$\nabla \cdot \mathbf{u}_R = 0. \tag{3.4.14}$$

In the relative reference frame with absolute velocity:

$$\frac{\partial \mathbf{u}_R}{\partial t} + \frac{d\omega_r}{dt} \times \mathbf{r} + \nabla \cdot (\mathbf{u}_R \times \mathbf{u}_I) + \omega_r \times \mathbf{u}_I = -\nabla \left(\frac{p}{\rho} \right) + \nabla \cdot (\nu \nabla \mathbf{u}_I), \tag{3.4.15}$$

$$\nabla \cdot \mathbf{u}_I = 0. \tag{3.4.16}$$

The incompressible governing equations for steady-state flow, for multiple frames of reference, can be written as:

- Inertial reference frame with absolute velocity

$$\nabla \cdot (\mathbf{u}_I \times \mathbf{u}_I) = -\nabla p + \nu \nabla \cdot \nabla (\mathbf{u}_I), \tag{3.4.17}$$

$$\nabla \cdot \mathbf{u}_I = 0. \tag{3.4.18}$$

- Rotating reference frame with relative velocity

$$\nabla \cdot (\mathbf{u}_R \times \mathbf{u}_R) + 2\omega_r \times \mathbf{u}_R + \omega_r \times \omega_r \times \mathbf{r} = -\nabla p + \nabla \cdot (\nu \nabla \mathbf{u}_R), \tag{3.4.19}$$

$$\nabla \cdot \mathbf{u}_I = 0. \tag{3.4.20}$$

- Rotating reference frame with absolute velocity

$$\nabla \cdot (\mathbf{u}_R \times \mathbf{u}_I) + \omega_r \times \mathbf{u}_I = -\nabla p + \nabla \cdot (\nu \nabla \mathbf{u}_I), \tag{3.4.21}$$

$$\nabla \cdot \mathbf{u}_I = 0. \tag{3.4.22}$$

Equations 3.4.17 - 3.4.22 are the basic set of equations used in MRF models [14].

Chapter 4

Immersed Boundary Method

In the previous chapter, basic principles of the Finite Volume Method were described. In the following chapter, the Immersed Boundary Method (IBM) will be laid out.

4.1 Introduction

IBM is a method of mesh generation used in the field of CFD. The obvious aim of IBM is to simplify the process of meshing complicated geometries. Another one is to decrease the computational time in dynamic mesh simulations.

4.2 Previous Studies

A review of various versions of IBM was presented in [9] and will be laid out here.

The idea of IBM was first proposed by Peskin in [12] to simulate cardiac mechanics and associated blood flow. A novel procedure was formulated for imposing the effect of the immersed boundary on the flow. Later, many new versions were proposed but in this thesis, the focus is on liquid-solid immersed boundary, whereas applications of IBM to problems with liquid-liquid and liquid-gas immersed boundaries will not be covered.

We shall use the example uf fluid flow past a solid body, shown in figures 4.2.1 and 4.2.2 to explain the difference between the conventional body-fitted approach (4.2.1) and the immersed boundary approach (4.2.2).

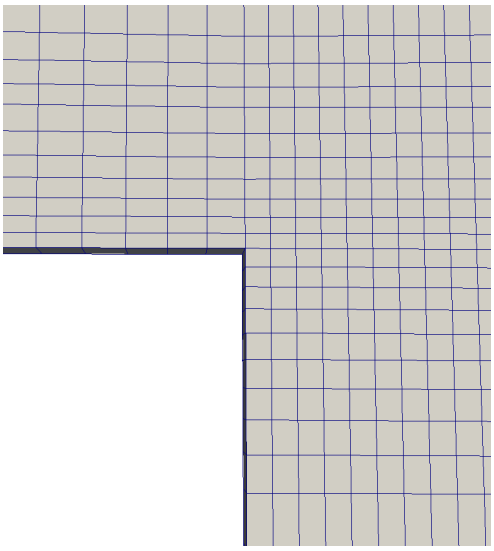


Figure 4.2.1: Body-fitted mesh detail

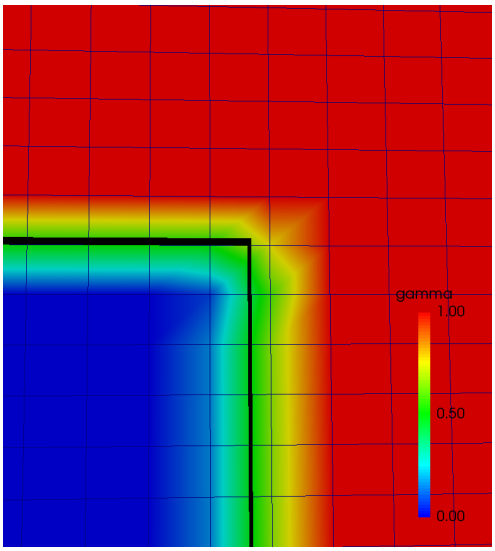


Figure 4.2.2: IBM mesh detail

In the body-fitted approach, a surface grid which covers the boundaries of the solid, is generated. Then it is used as a geometrical boundary condition to generate a mesh in the region occupied by the fluid, either a structured or an unstrucutred one.

When using the IBM, a so-called background mesh is created first. The background mesh represents the computational domain, *i.e.* a piece of real space which we intend to simulate, without any solid objects in it.

Then, surface of the solid object is inserted into the background mesh and the background mesh is divided into active (fluid) region and non-active (solid) region.

An example is shown in 4.2.2: the red part of the mesh corresponds to the fluid region, the blue part corresponds to the solid region and the black line corresponds to the surface of the solid.

Since the mesh does not conform to the solid boundary, incorporating boundary conditions requires modification of the equations in the vicinity of the boundary. Different versions of the IBM are distinguished based on the implementation of these modifications.

4.2.1 Continuous Forcing Approach

In this approach, modifications of boundary conditions definition are performed for the original governing equations and the modified equations are afterwards dicretized. This approach

requires a distinction of the ways in which elastic and rigid boundaries are treated.

Elastic Boundaries

In the first IBM version by Peskin [12], cardiac flow was simulated. This is an example of a flow with elastic bounderies, where a mutual interaction between the fluid and the solid exists.

The immersed boundary is represented by a set of elastic fibers and the location of these fibers is tracked in a Lagrangian fashion by a collection of massless points that move with the local fluid velocity. The coordinate $\mathbf{x}_k$ of the kth Lagrangian point is governed by the equation:

$$\frac{\partial \mathbf{x}_k}{\partial t} = \mathbf{u}\left(\mathbf{x}_k, t\right). \tag{4.2.1}$$

The stress $\mathbf{F}$ and deformation of these fibers is related by a constitutive law similar to Hooke's law. The effect of the immersed boundary on the surrounding fluid is essentially captured by transmitting the fiber stress to the fluid through a localized forcing term in the momentum equations, which is given by:

$$\mathbf{f}_m(\mathbf{x}, t) = \sum_k \mathbf{F}_k(t)\delta(|\mathbf{x} - \mathbf{x}_k|), \tag{4.2.2}$$

where δ is the Dirac delta function. The location of fibers does not generally coincide with the nodal points of the Cartesian grid so the forcing is distributed over a band of cells around each Lagrangian point and this distributed force is imposed on the momentum equation of the surrounding nodes. Thus, the sharp delta function is replaced by a smoother distribution function, which is suitable for use on a discrete mesh.

The choice of the distribution function is a key ingredient in this method.

Rigid Boundaries

The previously described method is well suited for elastic boundaries but is problematic for solids with rigid boundaries. A way a rigid boundary can be incorporated into the described model is to consider the boundary as an elastic but an extremely stiff one.

A second approach is to consider the structure attached to an equilibrium location by a spring with a restoring force $\mathbf{F}_k$ given by:

$$\mathbf{F}_k(t) = -\kappa\left(\mathbf{x}_k - \mathbf{x}_k^e(t)\right), \tag{4.2.3}$$

where κ is a positive spring constant and $\mathbf{x}_k^e$ is the equilibrium location of the kth Lagrangian point. Imposing the boundary condition on a rigid immersed boundary requires large values of κ. However, this results in a stiff system of equations that is subject to severe stability constraints. On the other hand, lower values of κ can lead to unwanted elastic effects such as excessive deviation from the equilibrium location.

A general model, of which the approach presented in 4.2.3 is a specific version, is:

$$\mathbf{F} = \alpha \int_0^T \mathbf{u}(t)dt + \beta\mathbf{u}(T), \tag{4.2.4}$$

where α and β are coefficients selected to best enforce the boundary condition at the immersed solid boundary. This method requires large values of α and β, which can lead to stability problems.

Another method in this class is one where the entire flow is assumed to occur in a porous medium. In this formulation, an extra force term is contained of the form

$$\mathbf{F} = \frac{\mu}{K}\mathbf{u}, \tag{4.2.5}$$

where K is the permeability of the medium and is defined as infinity for the fluid and as 0 for the solid region. The force therefore activates only in the solid region, driving the velocity to zero. In practice, K is large in the fluid region and small in the solid region, which, along with the smoothing of the variation of K at the fluid-solid interface, leads to an error in the imposition of the correct velocity on the solid surface. This method is also subject to stiffness problems associted with large variations in the values of K.

4.2.2 Discrete Forcing Approach

In this approach, modifications of boundary conditions definition are performed for the discretized form of governing equations. In this approach, methods are further categorized into those that are formulated to impose the boundary conditions on the immersed boundary through indirect means, and those that directly impose the boundary conditions on the immersed boundary.

Indirect Boundary Conditions Imposition

For a simple, analytically integrable, one-dimensional model problem, it is possible to formally derive a forcing term that enforces a specific condition on a boundary inside the computational domain. The same is usually not feasible for the governing equations because the equations can not be integrated analytically to determine the forcing function. Consequently, all the approaches in the previous section employ what are essentially simplified models of the required forcing. To avoid this issue, a method was developed that extracts the forcing directly from the numerical solution for which an *a priori* estimate can be determined.

The major advantage of the discrete forcing concept is the absence of user specified parameters in the forcing and the elimination of associated stability constraints. However, the forcing still extends into the fluid region due to the use of a distribution function and the details of the implementation depend strongly on the numerical algorithm used to discretize the governing equations.

Direct Boundary conditions Imposition

Although the application of IBM to low and moderate Reynolds number shows success, its extension to higher Reynolds numbers is challenging due to the need to accurately resolve the boundary layers on surfaces not aligned with the grid lines. In such cases the local accuracy of the solution assumes greater importance, and the spreading of the effect of the immersed boundary introduced by the smooth force distribution function is less desireable. For this reason, other approaches can be considered where the immersed boundary is retained as a sharp interface with no spreading and where greater emphasis is put on the local accuracy near the immersed boundary. This can usually be accomplished by modifying the computational stencil near the immersed boundary to directly impose the boundary condition on the immersed boundary. Two methods that belong to this category will be described.

Ghost-Cell Finite-Difference Approach

The boundary condition on the immersed boundary is enforced here through the use of "ghost cells". Ghost cells are defined as cells in the solid that have at least one neighbour in the fluid. For instance cell *G* in figure 4.2.3 is a ghost cell. For each ghost cell, an interpolation

scheme that implicitly incorporates the boundary condition on the immersed boundary is then devised. There is a number of available options for constructing the interpolation scheme. One simple option is bilinear (trilinear in 3-D) interpolation where a generic flow variable ϕ can be expressed as:

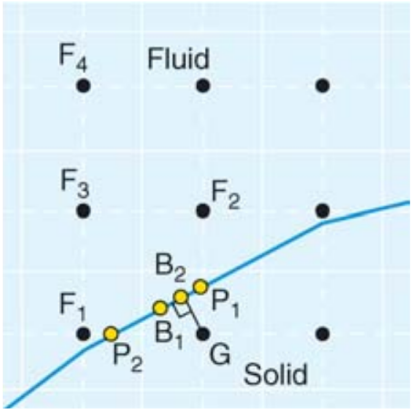


Figure 4.2.3: Representation of the points in the vicinity of an immersed boundary used in the ghost-cell approach. F_i are fluid points, G is the ghost point, and B_i and P_i are locations where the boundary condition can be enforced [9]

$$\phi = C_1 x_1 x_2 + C_2 x_1 + C_3 x_2 + C_4. \tag{4.2.6}$$

The coefficients in the above equation can be evaluated in terms of the values of ϕ at fluid nodes F_1, F_2 and F_3, and the boundary point B_2, which is the normal intercept from the ghost node to the immersed boundary. Boundary point B_1, which is midway between points P_1 and P_2, can also be used instead of B_2. Note that P_1 and P_2 are the intercepts with the y and x lines passing through the ghost node.

Applying a linear reconstruction is acceptable for laminar flows or for high Reynolds number flows when the first grid point is located in the viscous sublayer. At high Reynolds numbers when the resolution is marginal, linear reconstruction could lead to erroneous predictions. For such cases higher-order interpolation can be used. For instance, one could employ an interpolant which is linear in the tangential direction and quadratic in the normal direction, such as:

$$\phi = C_1 n_c^2 + C_2 n_c t_c + C_3 n_c + c_4 t_c + C_5, \tag{4.2.7}$$

where n_c and t_c are local coordinates, normal and tangent, respectively, to the immersed boundary. The coefficients can be determined by using the four fluid points values F_1 to F_4 and the boundary condition at point B_2 where the selection of point F_4 depends on the surface normal. Alternatively, the points F_1 to F_3 and the two boundary points P_1 and P_2 could be used without losing generality.

Irrespective of the particular interpolation scheme used, the value of the variable at the ghost-cell node, ϕ_G, can be expressed as:

$$\sum \omega_i \phi = \phi_G, \tag{4.2.8}$$

where the summation extends over all points in the stencil, including one or more boundary points, and ω_i are known geometry dependent coefficients.

The ghost-cell implementation is used in $foam-extend$ 4.0 and is described in [15].

Cut-Cell Finite-Volume Approach

None of the IBM versions described so far are designed to satisfy the underlying conservation laws for the cells in the vicinity of the immersed boundary. Strict global and local conservation of mass and momentum can only be guaranteed by resorting to a finite-volume approach and this is the primary motivation for the cut-cell methodology. Figure 4.2.4 shows a schematic of a Cartesian grid with an immersed boundary that which separates solid from fluid. In this method, cells in the Cartesian grid that are cut by the immersed boundary are identified, and the intersection of the boundary with the sides of these cut cells is determined. Next, cells cut by the immersed boundary, whose cell centers lie in the fluid, are reshaped by discarding the portion of these cells that lies in the solid. Pieces of cut cells whose centers lie in the solid are absorbed by neighbouring cells. This results in the formation of control volumes, which are trapezoidal in shape.

Finite-volume discretization of the governing equations requires the estimation of mass, convective and diffusive flux integrals, and pressure gradients on the faces of each cell and the issue is to evaluate these on the cell faces of the trapezoidal cells. One approach is to express

a given flow variable ϕ in terms of a two-dimensional polynomial interpolating function in an appropriate region and evaluate the fluxes f based on this interpolating function. For instance, to approximate the flux on the southwest face, f_{sw}, ϕ (in the trapezoidal region shown in figure 4.2.5) is expressed in terms of a function that is linear in x_1 and quadratic in x_2:

$$\phi = C_1 x_1 x_2^2 + C_2 x_2^2 + C_3 x_1 x_2 + C_4 x_1 + C_5 x_2 + C_6, \tag{4.2.9}$$

where C_1 to C_6 are six unknown coefficients that can be expressed in terms of values of ϕ at the six stencil points shown in 4.2.5 and an expression similar to equation 4.2.8 is developed for f_{sw}. Equation 4.2.9 represents the most compact function that allows at least a second-order accurate evaluation of ϕ or its derivative at the *sw* location. A similar approach can be employed to evaluate the flux on the east-face f_e as well as the interface flux f_i. This approach results in a discretization scheme that is globally, as well as locally, second-order accurate and also satisfies conservation of mass and momentum exactly irrespective of the grid resolution.

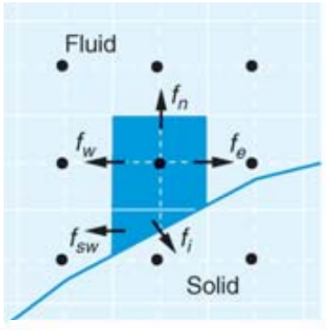


Figure 4.2.4: Trapezoidal finite volume formed near the immersed boundary for which f denotes the face-flux of a generic variable [9]

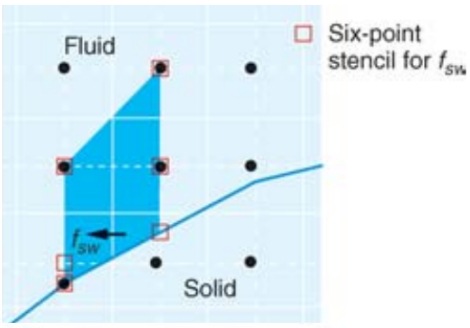


Figure 4.2.5: Region of interpolation and stencil employed for approximating the flux f_{sw} on the southwest face of the trapezoidal finite volume [9]

4.3 foam-extend 4.1 Implementation

A new version of IBM is implemented in $foam-extend$ 4.1 and was presented in [10]. Usage guidelines for IBM in $foam-extend$ 4.1 are presented in the appendix of this thesis. The version implemented in $foam-extend$ 4.0 performed satisfactorily in single-phase flows, but

carried some drawbacks: handling of Neumann boundary conditions and wall functions implementation. Also, in free surface flows, the solver showed strong instability due to matching of gradients next to the immersed boundary. Further improvement in robustness and accuracy was sought regarding the precision of discretization at the immersed boundary.

The previous implementation had problems with losing the information in the cell which is cut, which leads to loss of precision at the intersection. The objective of the new implementation was to implement the influence of the presence of the boundary within the mesh as if the mesh is body-fitted. That includes introducing the "new" immersed boundary face in the cut cell, accounting for the partial cell volume, accounting for partial face areas without loss of accuracy and calculating face and cell centers, all without changing the background geometric mesh.

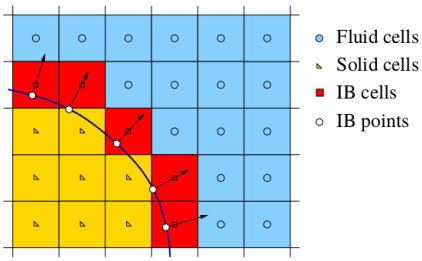


Figure 4.3.1: $foam-extend$ 4.0 IBM scheme [10]

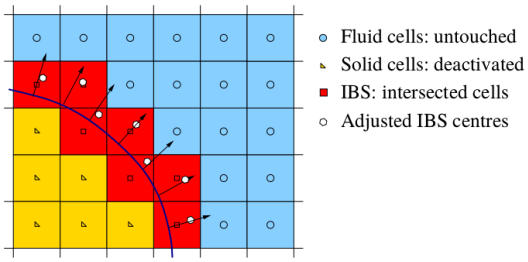


Figure 4.3.2: $foam-extend$ 4.1 IBM scheme [10]

In the new implementation, the immersed boundary patch is included into the mesh via the distance function: all cells that straddle the immersed boundary remain active. As opposed to the old implementation, the STL resolution is now not important.

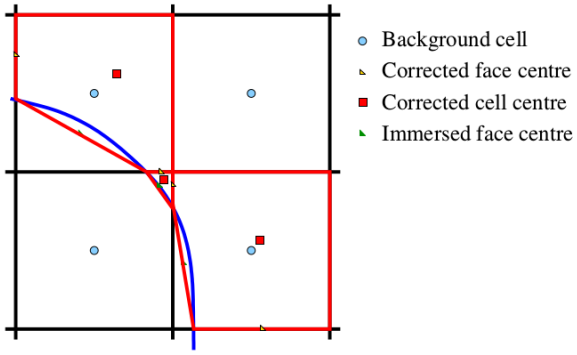


Figure 4.3.3: $foam-extend$ 4.1 cell cutting scheme [10]

In figure 4.3.3, the cell cutting scheme of the new implementation is shown. The nearest distance of the immersed boundary and the vertices of all affected cells is calculated. Intersection is calculated for all faces and cells and original data is replaced with new data of the "active part" (face and cell centers, face areas and cell volumes. Near-wall distance of new cells is calculated from active cell centers to the immersed faces. Delta coefficients and interpolation factors are corrected to take into account the new center positions.

Boundary Conditions

The immersed boundary is represented in the mesh by the intersection of the STL file and the cells. Therefore, conventional boundary conditions implementation suffice on the immersed patch for a static mesh and immersed boundary. For a moving immersed boundary, change in intersection is involved, which means that the number of immersed boundary faces changes. Evaluation of immersed boundary properties is performed without interpolation or simplification. Also, the STL surface is automatically refined or coarsened to comply with the background mesh. Due to finite accuracy, the STL can often coincide with faces or not provide accurate intersection. In case of a direct face intersection on a existing face, the face becomes the immersed boundary face. Inaccurate STL intersection may yield a geometrically open cell, with possible robustness issues. Using the so-called Marooney Maneouvre [10] guarantees a closed cell after cutting:

$$\sum_C S_f = 0 \tag{4.3.1}$$

is valid for a regular cell, while

$$\sum_C \gamma_f S_f + S_{fIB} = 0 \tag{4.3.2}$$

is valid for an intersected cell, where γ_f is the face area correction, obtained by cell cutting. The corrected immersed boundary face area is:

$$S_{fIB} = -\sum_C \gamma_f S_f. \tag{4.3.3}$$

4.4 Closure

In this chapter, previous and related studies of IBM have been laid out and the underlying theory of the $foam-extend$ 4.1 IBM version has been presented. In the following chapter, the $foam-extend$ 4.1 version of IBM will be validated on a series of cases.

Chapter 5

Validation Cases

In the previous chapter, IBM was laid out. In the following chapter, the results obtained using IBM will be compared to the results obtained by using body-fitted meshes, on the same validation cases, with equivalent geometries and boundary conditions.

First, some OpenFOAM syntax which is used while presenting the cases in this thesis should be described.

blockMesh is a OpenFOAM utility used for creating block-structured meshes in OpenFOAM. The zeroGradient boundary condition corresponds to the Neumann boundary condition where the gradient on the boundary patch is set to 0. The fixedValue boundary condition corresponds to the Dirichlet boundary condition where the value on the boundary patch has to be defined (e.g. zero velocity on walls or a fixed velocity on inlet patches). The inletOutlet boundary condition is a combination of zeroGradient and fixedValue, where if the mass flux points out of the domain on the boundary patch, the zeroGradient condition is active, and if the mass flux points into the domain, the inletValue is used. The kqRWallFunction is a turbulence kinetic energy k boundary condition used on walls. The omegaWallFunction is an eddy turnover time omega (ω) boundary condition used on walls. The nutkWallFunction is an eddy viscosity nut (ν_t) boundary condition used on walls, if wall functions are used. The empty boundary condition is used on the front and back patches of 2-D cases. Immersed boundary equivalents of kqRWallFunction, omegaWallFunction and nutkWallFunction are immersedBoundaryKqRWallFunction, immersedBoundaryOmegaWallFunction and immersedBoundaryNutWallFunction. With IBM boundary conditions, the triValue and triGradient key words are used, where triValue corresponds to a Dirichlet boundary condition and triGradient corresponds to a Neu-

mann boundary condition. The setDeadValue key word defines if the value of the field in the ”dead” (inactive) cells is set to a specified value or not. gamma is a field variable which describes the presence of the immersed boundary in the mesh, where gamma=1 coresponds to the fluid (active) region, gamma=0 corresponds to the solid (inactive) region and 0 <gamma< 1 corresponds to cells which are cut by the immersed boundary.

5.1 Backward Facing Step

The backward facing step case is a geometrically very simple case often used for validation of new software and methods in CFD. In this thesis, it was used as the most simple validation case. The setup and results of four variants of the case will be presented; laminar body-fitted, laminar IBM, turbulent body-fitted and turbulent IBM.

5.1.1 Case Setup

The backward facing step is a geometrically simple 2-D case, whose geometry is shown in Figure 5.1.1.

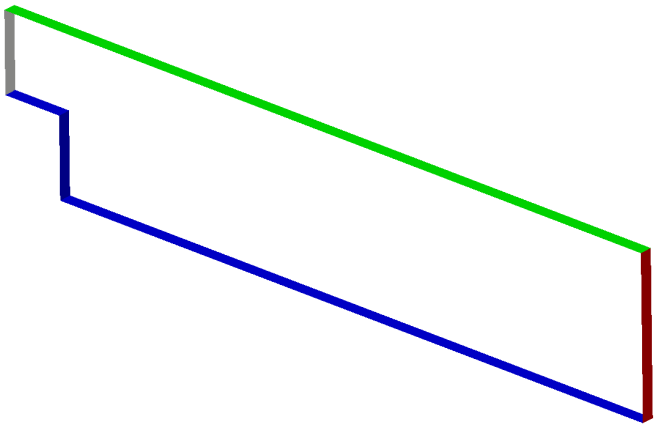


Figure 5.1.1: Backward facing step geometry

The length of the domain is 22.5 cm, the height is 5.08 cm, the height of the inlet is 2.54 cm and the step is 1.9 cm away from the inlet. In figure 5.1.1, the inlet is colored white, the outlet

red, the upper wall green, and the lower wall blue. Front and back patches are transparent.

The meshes were made using the *blockMesh* utility, both for the body-fitted and for the IBM mesh case. The STL used for definining the immersed boundary in the IBM cases was obtained from OpenFOAM tutorial cases. The body-fitted mesh which was used is shown in 5.1.2, the IBM mesh used is shown in 5.1.3 and the gamma field in the IBM mesh is shown in 5.1.4. The cell count is 3 240 for the body-fitted mesh and 4 032 for the IBM mesh.

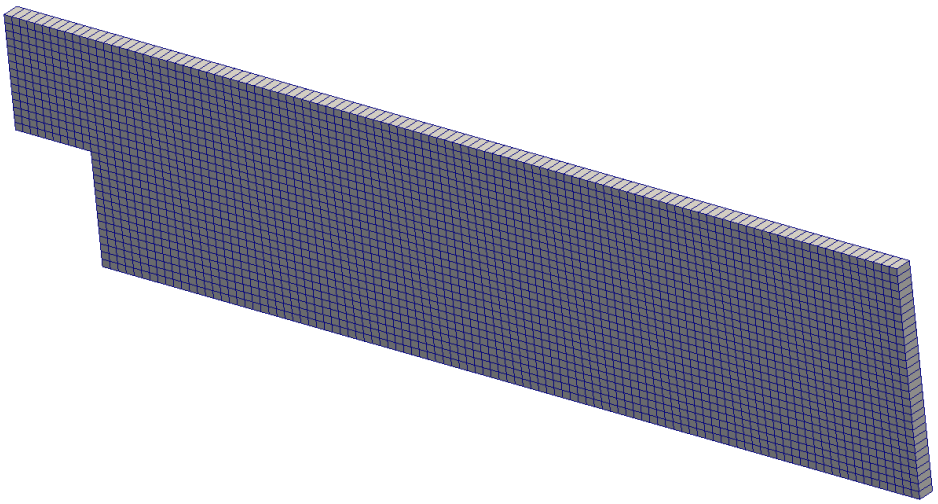


Figure 5.1.2: Backward facing step body-fitted mesh

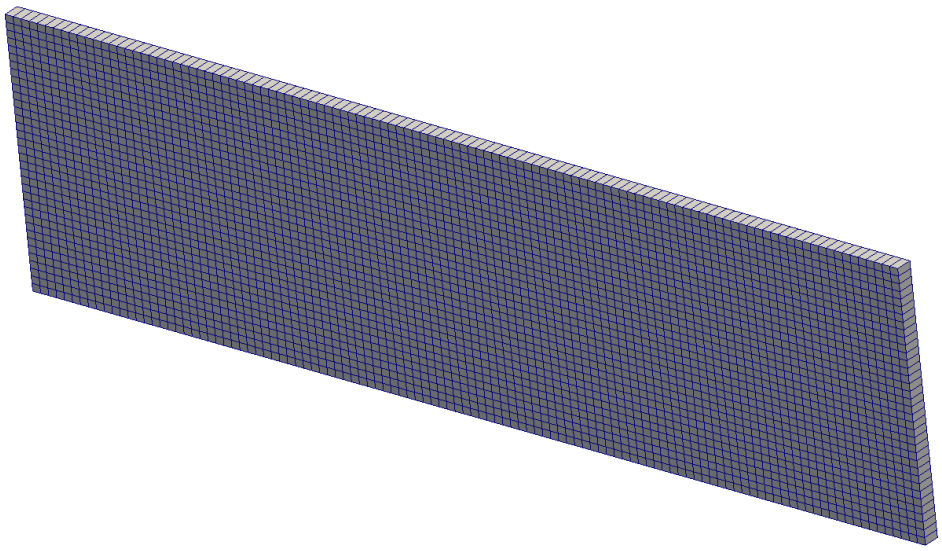


Figure 5.1.3: Backward facing step IBM background mesh

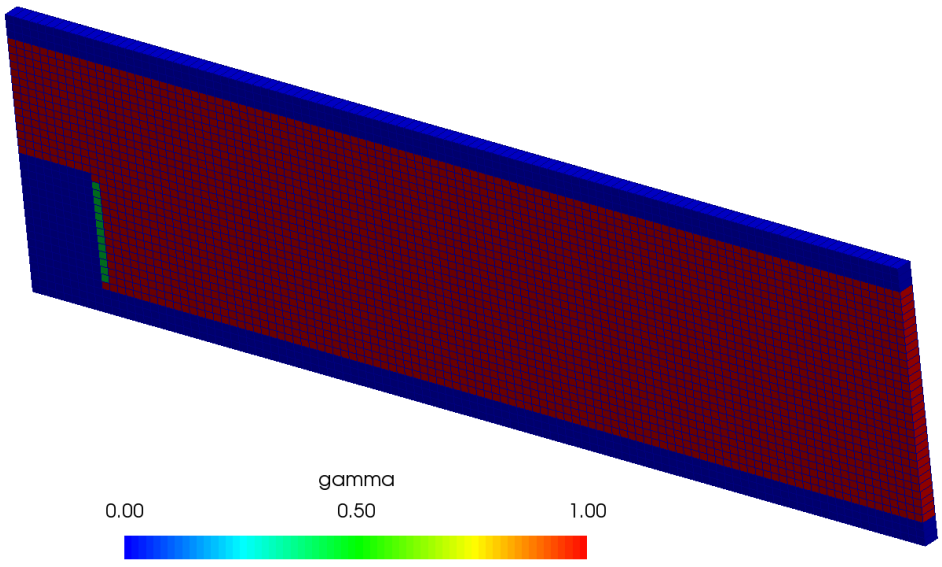


Figure 5.1.4: Backward facing step IBM mesh, red is the active (fluid) region, blue is the inactive region, the STL used to "cut" the background mesh is white

	field	
patch	**p** [m^2s^{-2}]	**U** [ms^{-1}]
Inlet	zeroGradient	fixedValue uniform (1 0 0)
Outlet	fixedValue uniform 0	inletOutlet inletValue uniform (0 0 0)
UpperWall	zeroGradient	fixedValue uniform (0 0 0)
LowerWall	zeroGradient	fixedValue uniform (0 0 0)
FrontAndBack	empty	empty

Table 5.1.1: Backward facing step BF boundary conditions for the laminar simulation

	field	
patch	**p** [m^2s^{-2}]	**U** [ms^{-1}]
Inlet	zeroGradient	fixedValue uniform (1 0 0)
Outlet	fixedValue uniform 0	inletOutlet inletValue uniform (0 0 0)
UpperWall	zeroGradient	fixedValue uniform (0 0 0)
LowerWall	zeroGradient	fixedValue uniform (0 0 0)
FrontAndBack	empty	empty
pitzDailyIB	mixedIb triGradient uniform 0 setDeadValue no	mixedIb triValue uniform (0 0 0) setDeadValue yes

Table 5.1.2: Backward facing step IBM boundary conditions for the laminar simulation

	field				
patch	**p** [m^2s^{-2}]	**U** [ms^{-1}]	**k** [m^2s^{-2}]	**omega** [s^{-1}]	**nut** [m^2s^{-1}]
Inlet	zeroGradient	fixedValue uniform (10 0 0)	fixedValue uniform 0.375	fixedValue uniform 375	zeroGradient
Outlet	fixedValue uniform 0	inletOutlet inletValue uniform (0 0 0)	zeroGradient	zeroGradient	zeroGradient
UpperWall	zeroGradient	fixedValue uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
LowerWall	zeroGradient	fixedValue uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
FrontAndBack	empty	empty	empty	empty	empty

Table 5.1.3: Backward facing step BF boundary conditions for the turbulent simulation

	field				
patch	**p** [m^2s^{-2}]	**U** [ms^{-1}]	**k** [m^2s^{-2}]	**omega** [s^{-1}]	**nut** [m^2s^{-1}]
Inlet	zeroGradient	fixedValue uniform (10 0 0)	fixedValue uniform 0.375	fixedValue uniform 375	zeroGradient
Outlet	fixedValue uniform 0	inletOutlet inletValue uniform (0 0 0)	zeroGradient	zeroGradient	zeroGradient
UpperWall	zeroGradient	fixedValue uniform (0 0 0)	zeroGradient	zeroGradient	zeroGradient
LowerWall	zeroGradient	fixedValue uniform (0 0 0)	zeroGradient	zeroGradient	zeroGradient
FrontAndBack	empty	empty	empty	empty	empty
pitzDailyIB	mixedIb triGradient uniform 0 setDeadValue no	mixedIb triValue uniform (0 0 0) setDeadValue yes	immersedBoundary-KqRWallFunction setDeadValue yes	immersedBoundary-OmegaWallFunction setDeadValue yes	immersedBoundary-NutWallFunction setDeadValue yes

Table 5.1.4: Backward facing step IBM boundary conditions for the turbulent simulation

Simulation Settings

The turbulent simulations were performed using the $k-\omega$ *SST* turbulence model. Fluid properties which were used in the simulation correspond to air. The potentialFoam solver was used to initialize the flow field. The simpleFoam solver was used to obtain a steady-state solution, in 2500 non-linear iterations, which was enough for the simulations to converge.

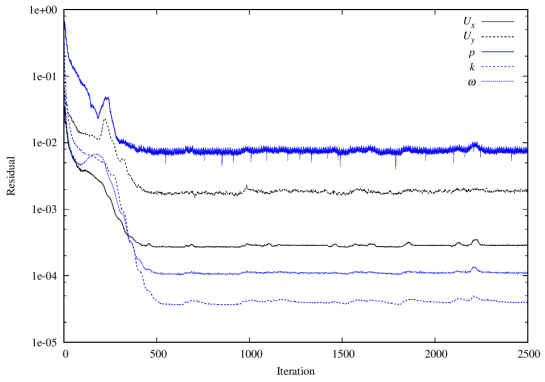


Figure 5.1.5: Backward facing step BF convergence history, turbulent simulation

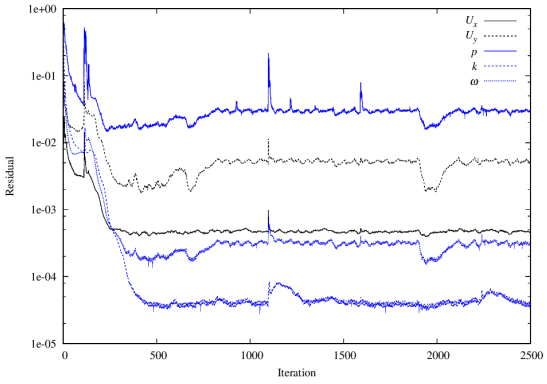


Figure 5.1.6: Backward facing step IBM convergence history, turbulent simulation

5.1.2 Results and Validation

The results will be presented qualitatively, using figures which portray the resulting scalar and vector fields of the simulations (figures 5.1.7 - 5.1.14), and quantitatively, by comparing values which are the usual sought after values of simulating internal flow: the pressure drop between the inlet and the outlet and the wall shear stress tensor magnitude on the upper wall of the domain. The figures which show the obtained results are from turbulent body-fitted and IBM simulations. The laminar simulations show even better results, in a sense that body-fitted mesh and IBM mesh simulations produce almost identical results for laminar simulations.

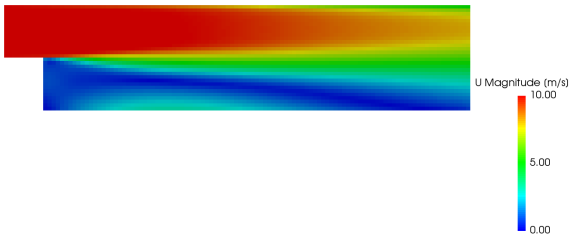


Figure 5.1.7: Backward facing step BF velocity field visualization, turbulent simulation

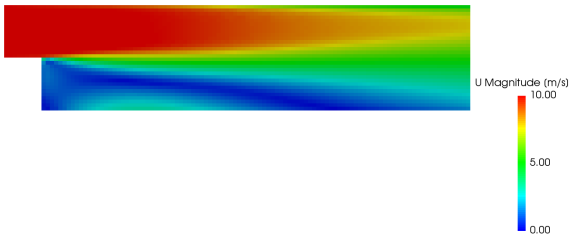


Figure 5.1.8: Backward facing step IBM velocity field visualization, turbulent simulation

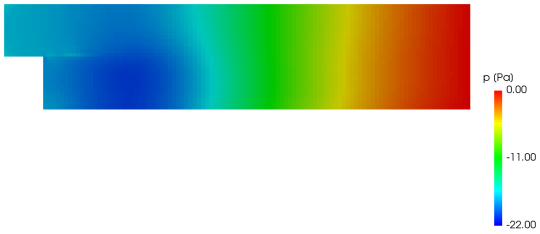


Figure 5.1.9: Backward facing step BF pressure field visualization, turbulent simulation

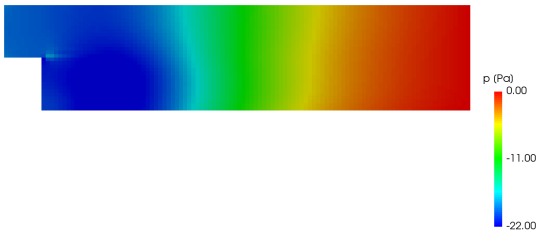


Figure 5.1.10: Backward facing step IBM pressure field visualization, turbulent simulation

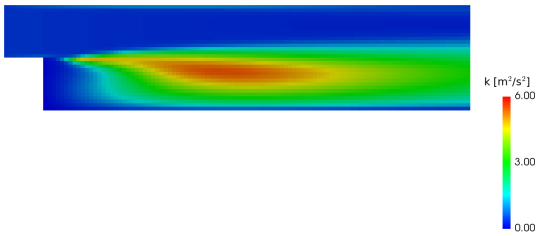


Figure 5.1.11: Backward facing step BF turbulence kinetic energy field visualization, turbulent simulation

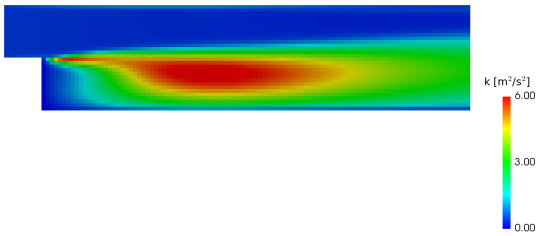


Figure 5.1.12: Backward facing step IBM turbulence kinetic energy field visualization, turbulent simulation

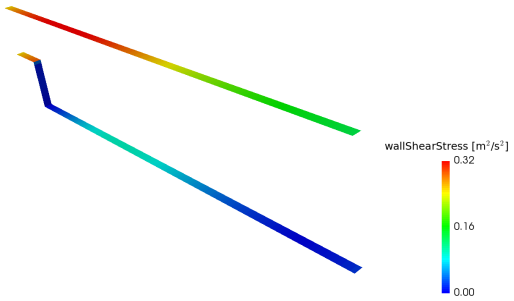


Figure 5.1.13: Backward facing step BF wall shear stress visualization, turbulent simulation

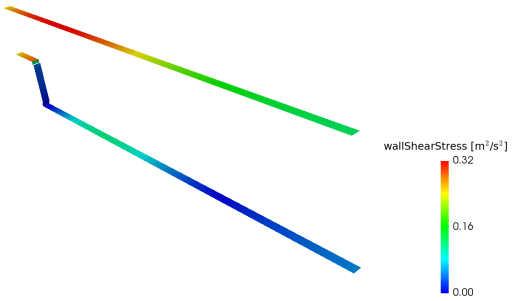


Figure 5.1.14: Backward facing step IBM wall shear stress field visualization, turbulent simulation

The comparison of the pressure drop for BF and IBM turbulent simulations is shown in figure 5.1.16 for turbulent simulations and in 5.1.22 for laminar simulations. It was sampled on a line whose starting point is at the center of the inlet patch and the end point is at the center of the outlet patch, figure 5.1.15. The comparison of resulting pressure and wall shear stress fields is shown in figures 5.1.16 and 5.1.17 for turbulent simulations and in figures 5.1.22 and 5.1.23 for laminar simulations.

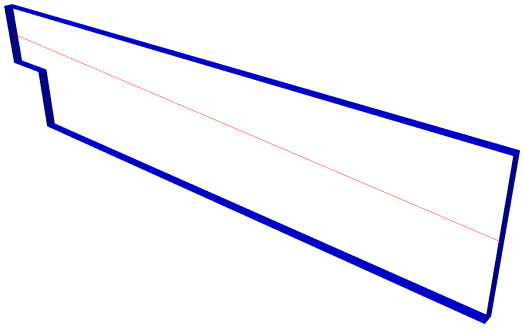


Figure 5.1.15: Backward facing step BF and IBM pressure drop through the channel comparison line visualization

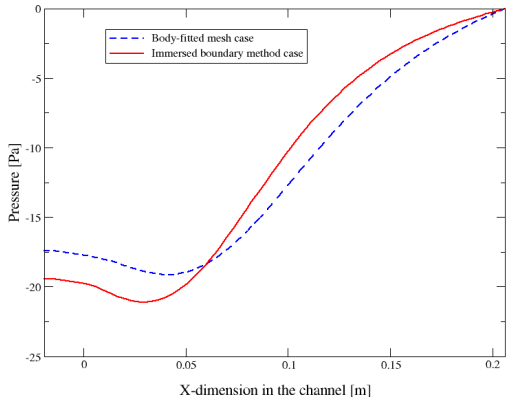


Figure 5.1.16: Backward facing step BF and IBM pressure drop through the channel comparison, turbulent simulations

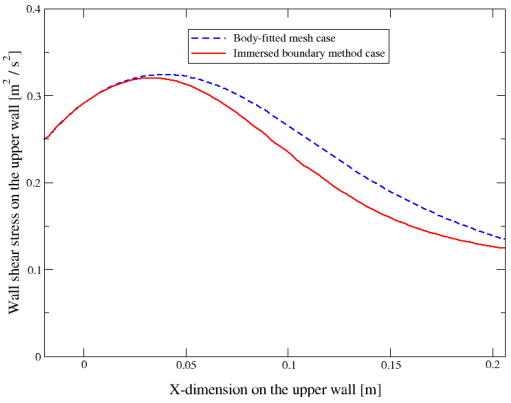


Figure 5.1.17: Backward facing step BF and IBM wall shear stress on the upper wall comparison, turbulent simulations

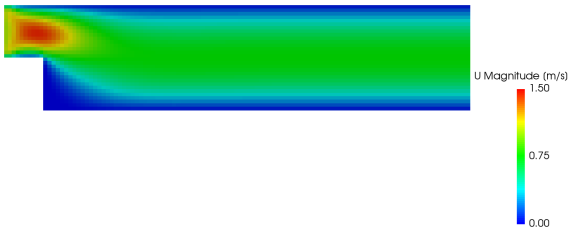


Figure 5.1.18: Backward facing step BF velocity field visualization, laminar simulation

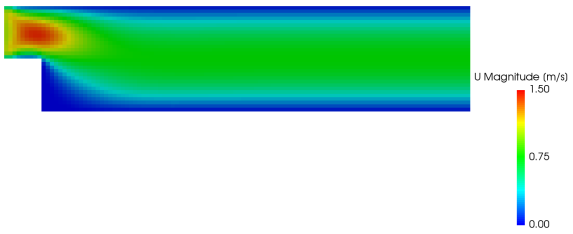


Figure 5.1.19: Backward facing step IBM velocity field visualization, laminar simulation

Figure 5.1.20: Backward facing step BF pressure field visualization, laminar simulation

Figure 5.1.21: Backward facing step IBM pressure field visualization, laminar simulation

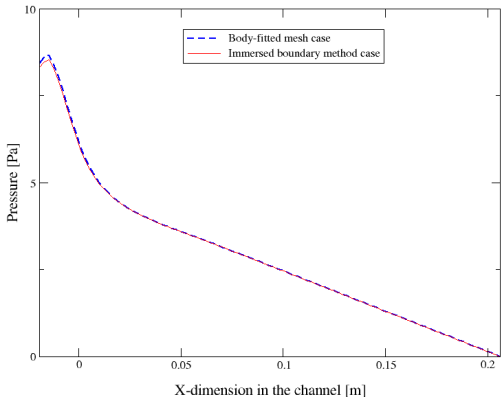


Figure 5.1.22: Backward facing step BF and IBM drop through the channel comparison line visualization, laminar simulation

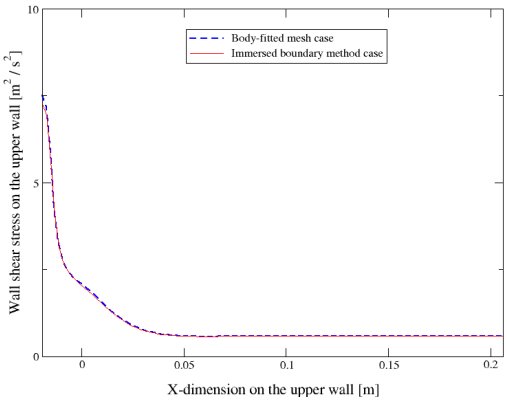


Figure 5.1.23: Backward facing step BF and IBM wall shear stress on the upper wall comparison, laminar simulation

The y^+ values of the turbulent simulations on the upper wall and the lower wall are shown in 5.1.24 and 5.1.25. The red parts are the cells where y^+ is larger than 30, which is the minimal required value when using wall functions. The blue parts are the ones where y^+ is smaller than 30. The y^+ values are additionally discussed in a following section.

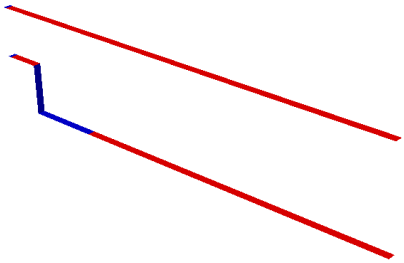


Figure 5.1.24: Backward facing step BF y^+ values visualization

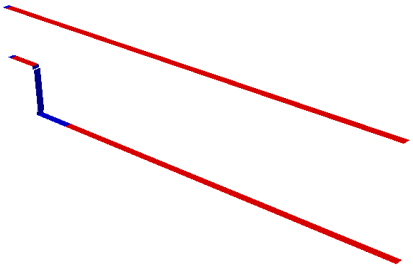


Figure 5.1.25: Backward facing step IBM y^+ values visualization

5.1.3 Closure

IBM simulations of the backward facing step show good results in comparison to BF simulations with the same input parameters. That is especially true for the laminar simulations, where the obtained results are almost identical to the BF simulations results.

For turbulent simulations, shown in figures 5.1.16 and 5.1.17, the resulting IBM simulation pressure and wall shear stress fields are not identical to the ones obtained in the body-fitted simulations, but show relatively good agreement. The difference of pressure drops between the inlet and outlet patches between the body-fitted and the IBM simulations is 10.4%. The agreement of resulting fields can also be observed in figures 5.1.7 - 5.1.14.

For laminar simulations, the results obtained by the IBM and BF simulations are almost identical, which can be seen in figures 5.1.22 and 5.1.23. The difference of pressure drops between the inlet and outlet patches between the body-fitted and the IBM simulations is only 1.3%.

5.2 Onera M6 Wing

Onera M6 wing is a case often used for validation of new software and methods used to simulate external flows. It has been used since 1979 when a detailed report of flow experiments was published in [16]. That enabled CFD researchers to compare results obtained with new simulation software and methods with the results of the experiments. The experiments were carried out with various Mach numbers (0.7, 0.84, 0.88 and 0.92), angles of attack (up to 6 degrees) and with Reynolds numbers of about $12 \cdot 10^6$.

In this thesis the experimental results were not used, as the experiments were performed for flow conditions which would require the usage of compressible solvers ($Ma > 0.3$). The aim of thesis is not to validate a certain solver but to validate IBM. For that reason it is not important to have a set of data of experimental origin to compare it with IBM simulation results. The IBM results will only be compared to the results of performed incompressible body-fitted simulations, with a much lower Mach number than in the original experiments, which simplifies the simulation process.

5.2.1 Case Setup

The geometry of the Onera M6 wing was taken from [17] and is shown in 5.2.1. The mean aerodynamic cord used for the Reynolds number formula is 0.65 m and the reference area for aerodynamic coefficients is 1.55 m^2. The height of the wing is 1.22 m.

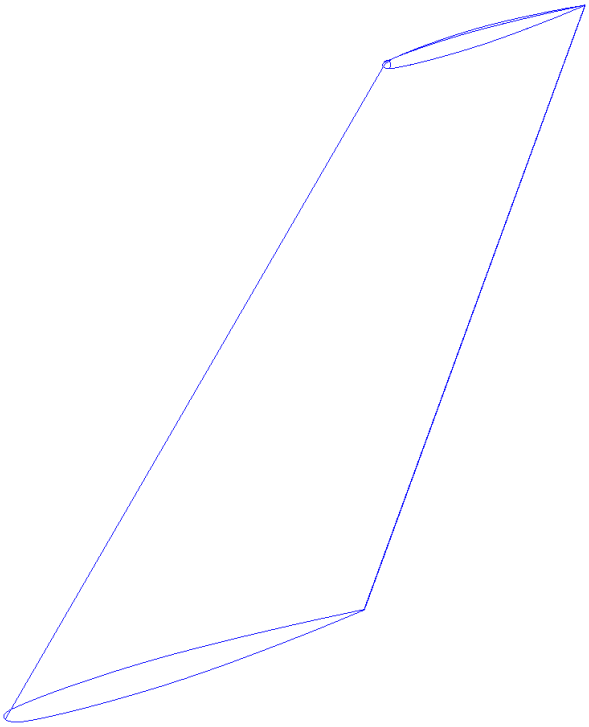


Figure 5.2.1: Onera M6 wing geometry

The meshes were made in Pointwise [18], which is a commercial mesh generation software. A section of the body-fitted case mesh geometry is shown in Figure 5.2.2 and the same for the IBM case is shown in Figure 5.2.3. The inlet is colored red and it is a C-type farfield. The outlet is colored green, the top patch yellow, bottom is white and the side patches (only one is visible in Figure 5.2.2 and Figure 5.2.3) are colored blue. In the body-fitted case the wing patch is colored pink and in the IBM mesh geometry there is no wing patch since only the background mesh is shown.

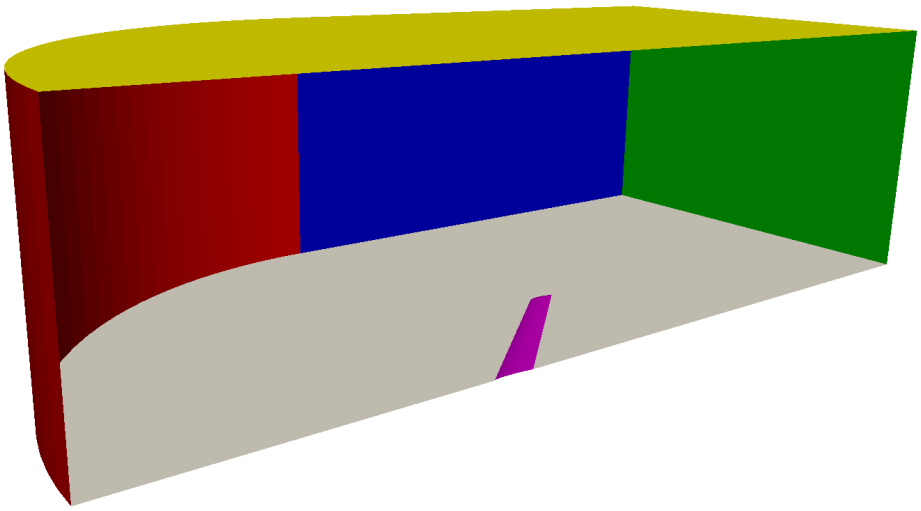


Figure 5.2.2: Onera M6 body-fitted mesh geometry section

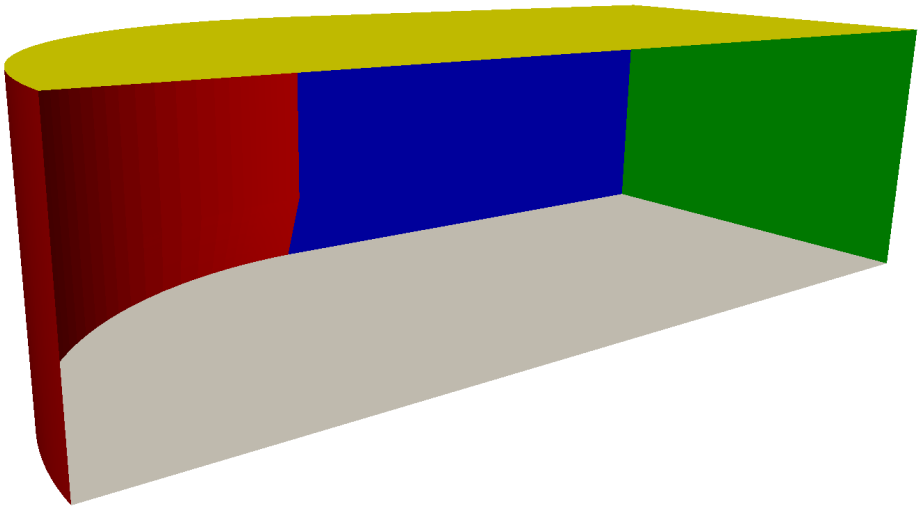


Figure 5.2.3: Onera M6 IBM mesh geometry section

Cell distribution in the meshes is shown in Figure 5.2.6 - Figure 5.2.8. Cell count for the body-fitted and IBM cases are 3 458 906 and 4 622 448, respectively.

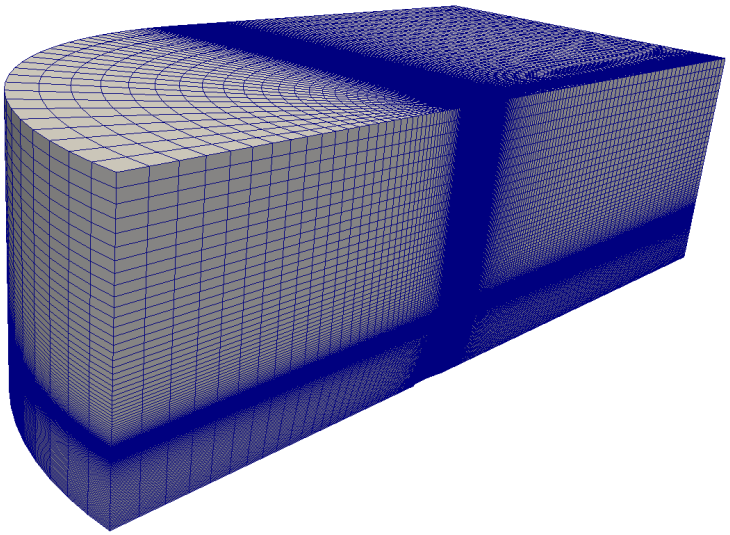


Figure 5.2.4: Onera M6 body-fitted mesh domain cell distribution

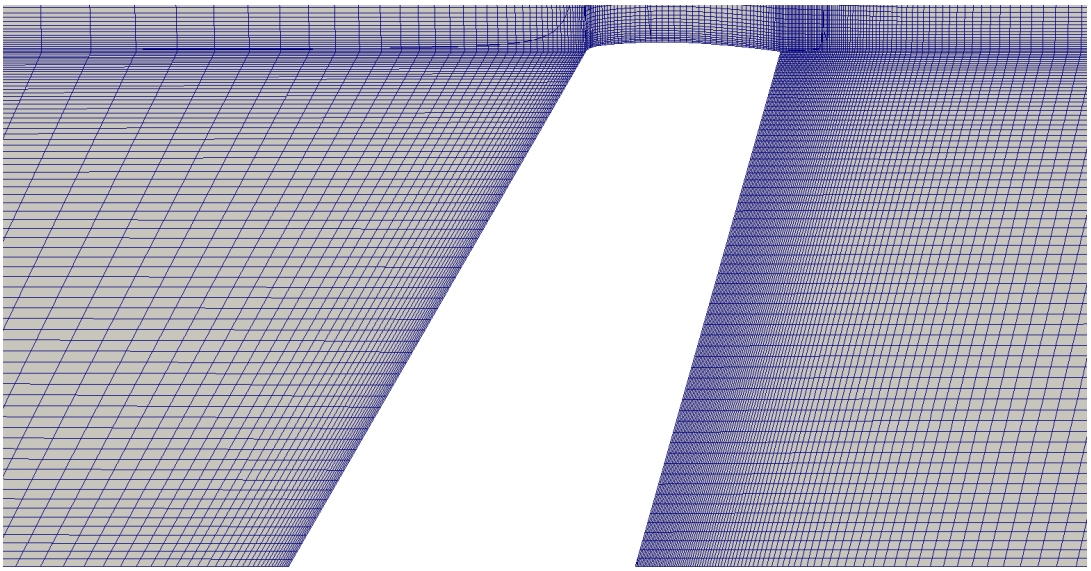


Figure 5.2.5: Onera M6 body-fitted mesh domain cell distribution near the wing

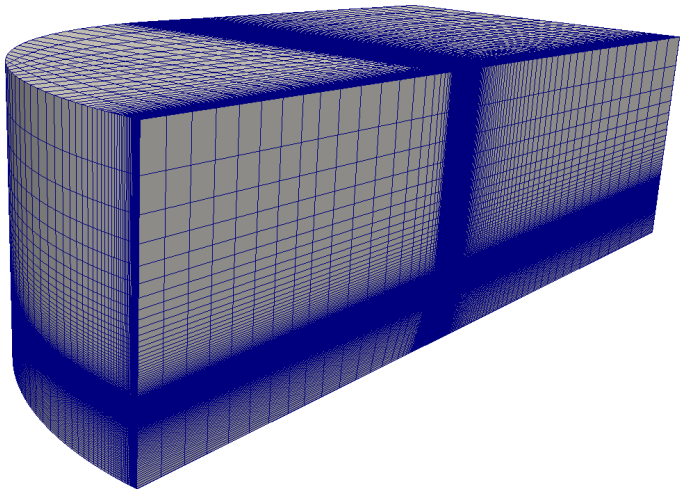


Figure 5.2.6: Onera M6 IBM mesh domain cell distribution

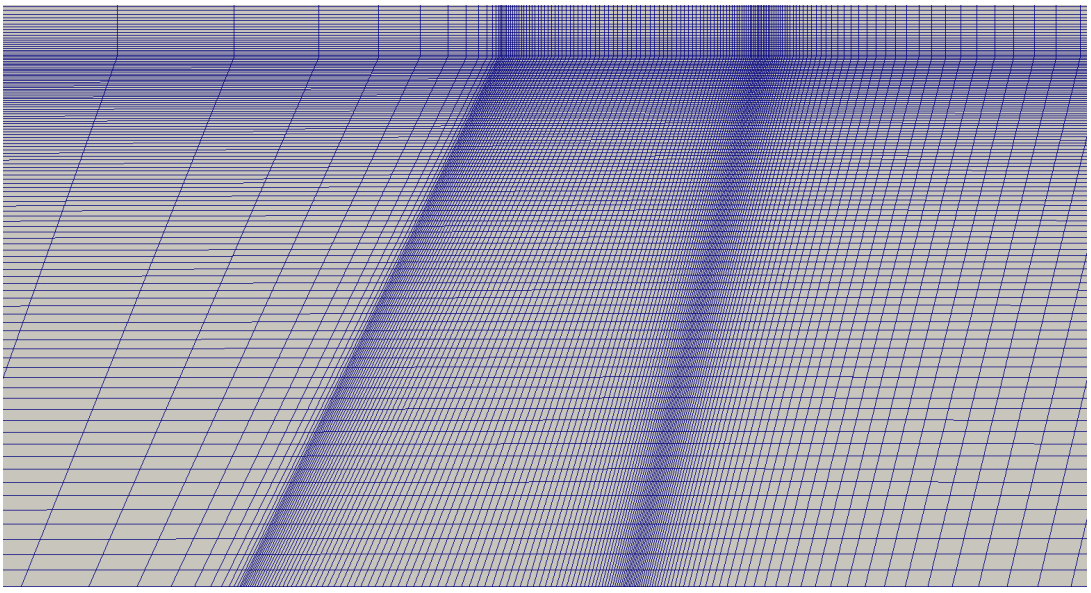


Figure 5.2.7: Onera M6 IBM mesh domain cell distribution near and through the wing

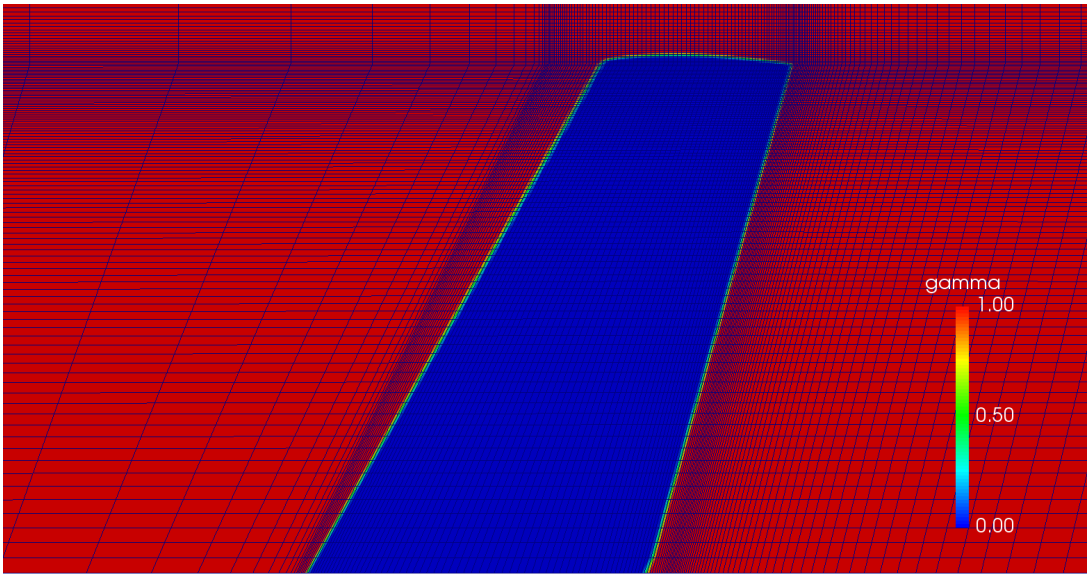


Figure 5.2.8: Onera M6 IBM mesh gamma field near and through the wing

The boundary conditions for individual patches, both for the body-fitted (Wing patch) and the IBM (WingIB patch) simulations, are shown in table 5.2.1.

	field				
patch	**p** [m^2s^{-2}]	**U** [ms^{-1}]	**k p** [m^2s^{-2}]	**omega p** [s^{-1}]	**nut p** [m^2s^{-1}]
Wing	zeroGradient	fixedValue uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
WingIB	mixedIb triGradient uniform 0 setDeadValue no	mixedIb triValue uniform (0 0 0) setDeadValue yes	immersedBoundary- KqRWallFunction setDeadValue yes	immersedBoundary- OmegaWallFunction setDeadValue yes	immersedBoundary- NutWallFunction setDeadValue yes;
Inlet	zeroGradient	inletOutlet inletValue uniform (20 0 0)	inletOutlet inletValue uniform 0.06	inletOutlet inletValue uniform 60	calculated value uniform 0
Outlet	fixedValue value uniform 0	inletOutlet inletValue uniform (20 0 0)	inletOutlet inletValue uniform 0.06	inletOutlet inletValue uniform 60	calculated value uniform 0
Top	slip	slip	slip	slip	calculated value uniform 0
Bottom	slip	slip	slip	slip	calculated value uniform 0
Side	zeroGradient	inletOutlet inletValue uniform (20 0 0)	inletOutlet inletValue uniform 0.06	inletOutlet inletValue uniform 60	calculated value uniform 0

Table 5.2.1: Onera M6 wing boundary conditions

The simulations were performed for different angles of attack (0 1, 3, 5, 8, 12 and 18

degrees). Changes in angles of attack were carried out by changing the inletOutlet boundary condition inletValue parameters. The velocity vectors in Table 5.2.1, (20 0 0) correspond to an angle of attack of 0 degrees. E. g. for an angle of attack of 5 degrees, the velocity vectors would be:

$$(20cos(5^\circ)\ 0\ 20sin(5^\circ)) = (19.924\ 0\ 1.743)\ [\mathrm{ms}^{-1}]. \tag{5.2.1}$$

Simulation Settings

The simulations were performed using the $k-\omega\ SST$ turbulence model. Fluid properties which were used in the simulation correspond to air. The potentialFoam solver was used to initialize the flow field. The simpleFoam solver was used to obtain a steady-state solution, in 1000 non-linear iterations, which was enough for the simulations to converge.

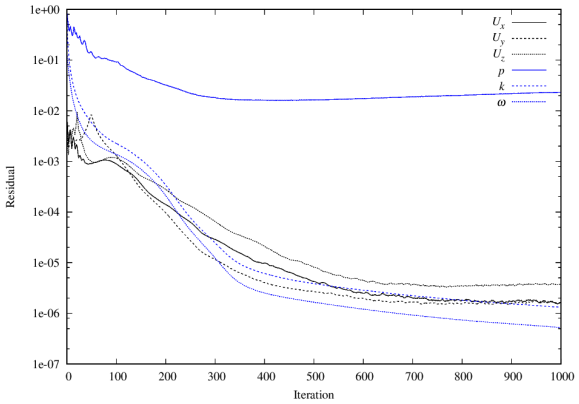


Figure 5.2.9: Onera M6 BF convergence history, angle of attack of 5 degrees

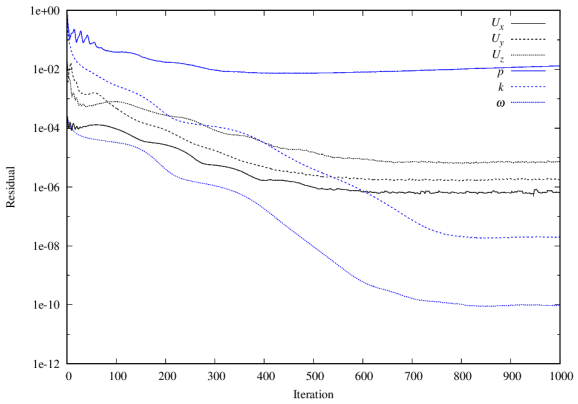


Figure 5.2.10: Onera M6 IBM convergence history, angle of attack of 5 degrees

5.2.2 Results and Validation

The results will be presented qualitatively, using figures which portray the resulting scalar and vector fields of the simulations, and quantitatively, by comparing values which are the usual sought after values of simulating flow around a wing: drag and lift coefficients and pressure coefficient distribution across the wing on a height (Y coordinate) of 0.5 meters. The figures which show the obtained results are from a simulation which was performed for 5° angle of

attack. The simulations with different angles of attack exhibit comparable results, in a sense that body-fitted mesh and IBM mesh simulations give similar values.

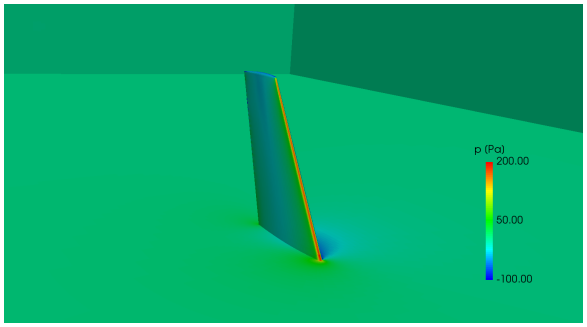


Figure 5.2.11: Onera M6 BF pressure field visualization with an angle of attack of 5 degrees

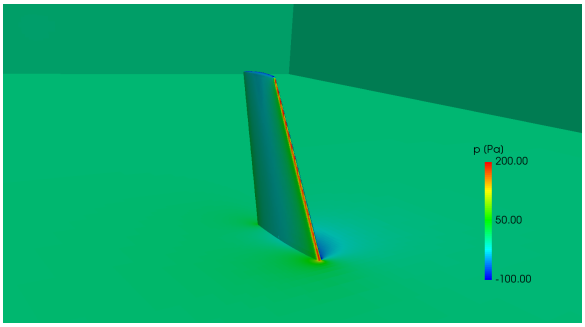


Figure 5.2.12: Onera M6 IBM pressure field visualization with an angle of attack of 5 degrees

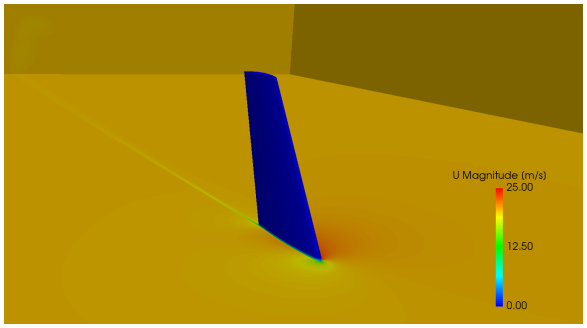


Figure 5.2.13: Onera M6 BF velocity field visualization with an angle of attack of 5 degrees

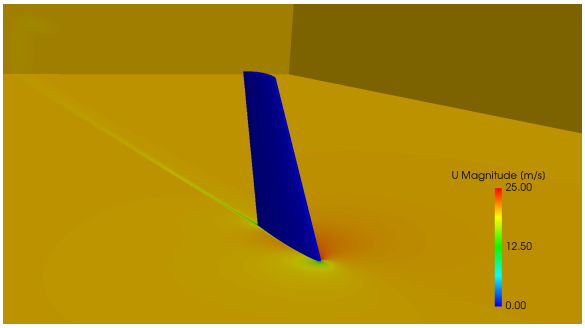


Figure 5.2.14: Onera M6 IBM velocity field visualization with an angle of attack of 5 degrees

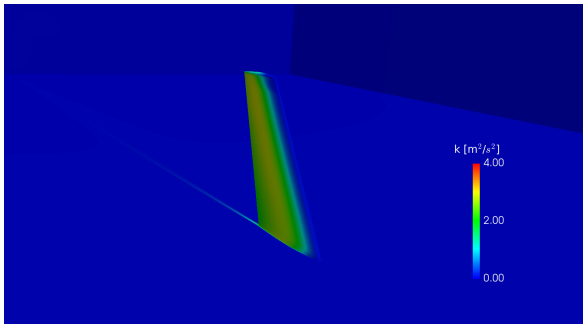


Figure 5.2.15: Onera M6 BF turbulence kinetic energy field visualization with an angle of attack of 5 degrees

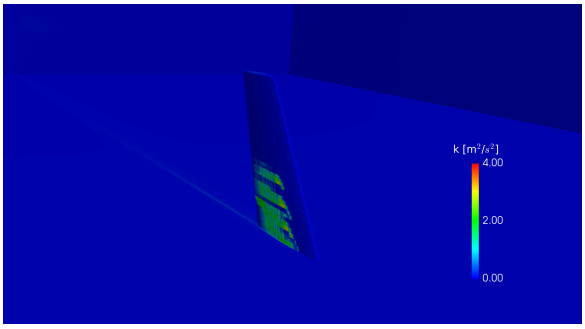


Figure 5.2.16: Onera M6 IBM turbulence kinetic energy field visualization with an angle of attack of 5 degrees

Figure 5.2.17: Onera M6 pressure = 15 *Pa* contour with an angle of attack of 5 degrees, yellow contour - IBM result, white contour - BF result

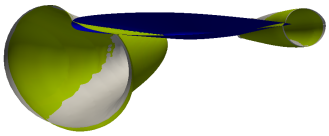


Figure 5.2.18: Onera M6 pressure = 15 *Pa* contour with an angle of attack of 5 degrees, alternate angle, yellow contour - IBM result, white contour - BF result

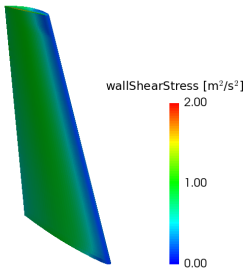


Figure 5.2.19: Onera M6 BF wall shear stress field visualization with an angle of attack of 5 degrees

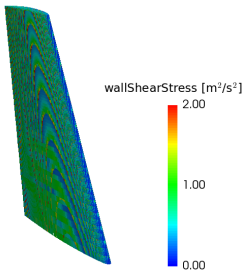


Figure 5.2.20: Onera M6 IBM wall shear stress field visualization with an angle of attack of 5 degrees

The wall shear stress visualization in 5.2.19 and 5.2.20 shows a discrepancy between BF and IBM results. The resulting wall shear stress field discrepancy is one of the causes of the force coefficients differences between BF and IBM simulations. Wall shear stress accuracy is one of the issues associated with IBM, which is confirmed here.

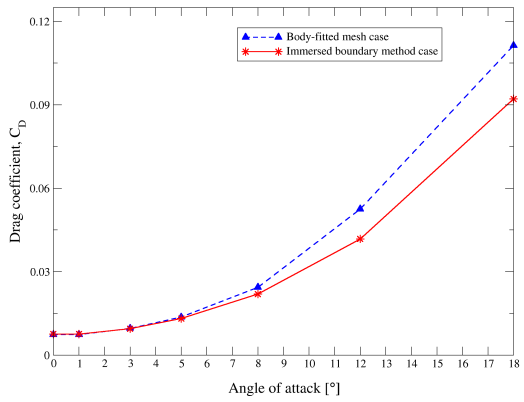


Figure 5.2.21: Onera M6 drag coefficients for different angles of attack

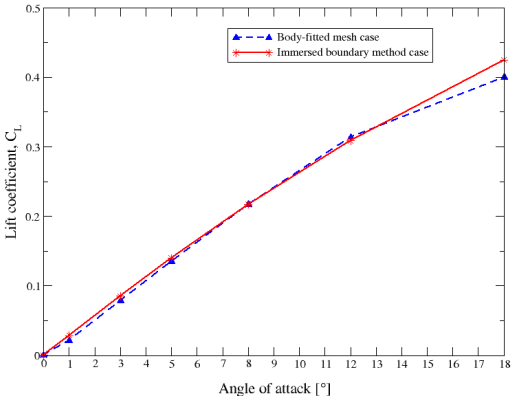


Figure 5.2.22: Onera M6 lift coefficients for different angles of attack

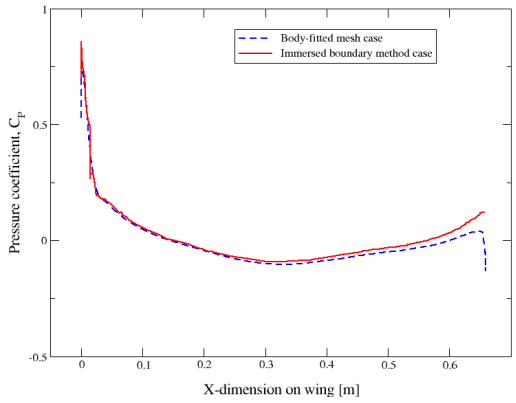


Figure 5.2.23: Onera M6 frontside pressure coefficients at wing height of 0.5 m

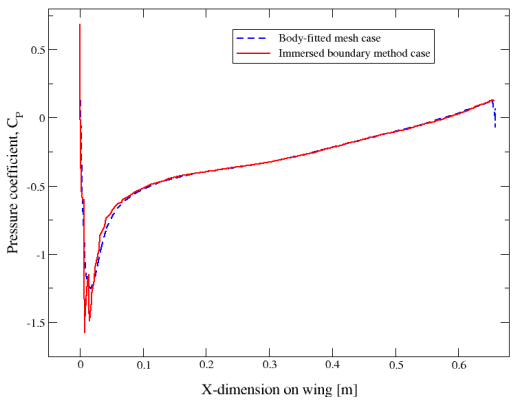


Figure 5.2.24: Onera M6 backside pressure coefficients at wing height of 0.5 m

The Onera M6 case is the most challenging one concerning the y^+ values and will be discussed in a following section.

5.2.3 Closure

IBM simulations of the Onera M6 wing show good results if compared to body-fitted simulations with the same input parameters. That is especially true for simulations with smaller angles of attack, where the obtained results are practically identical to the BF simulations results. This can be seen in figures 5.2.22 and 5.2.21. For even larger angles of attack (over 8 degrees), the drag and lift coefficients on the wing are relatively satisfactory when compared to BF simulations, with approximately 15 - 20% difference for the drag coefficient and approximately 5% for the lift coefficient. The agreement of resulting fields can also be observed in figures 5.2.11 - 5.2.18. The agreement of resulting pressure coefficients on the wing can be observed in figures 5.2.23 and 5.2.24.

5.3 Francis-99 Model Turbine

Francis-99 [19] is a project whose aim is to provide an open platform to hydropower researchers and to give the possibility to explore their capabilities and enhance their skills. Due to confidentiality, almost no modern turbine designs are available in public domains. This makes it

difficult for academic researchers to evaluate turbine design. Here, an open access to the complete design and operating data of a model Francis turbine is provided. The geometry of the model turbine has been used for IBM validation in this thesis. The test rig of the model Francis turbine is shown in 5.3.1 and its cut view is shown in 5.3.2.

Figure 5.3.1: Test rig of the model Francis turbine/pump-turbine [19]

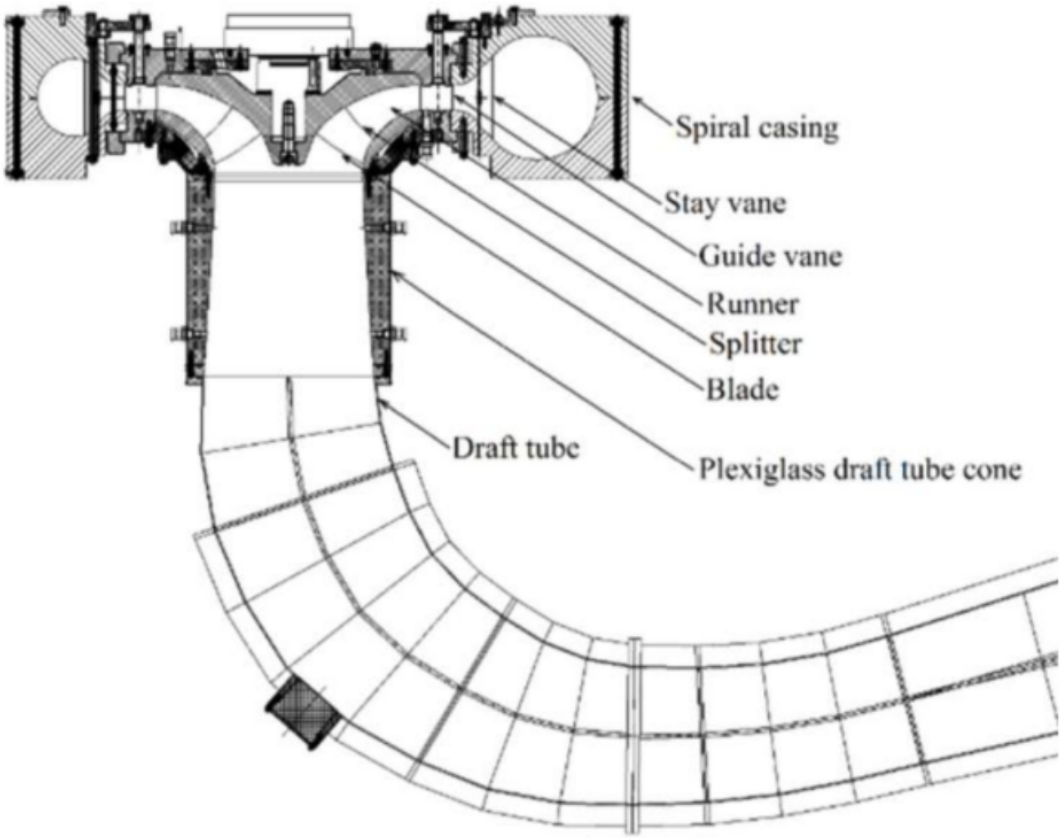


Figure 5.3.2: Cut view of the Francis 99 turbine model [20]

The model Francis turbine utilized for Francis-99 is a scaled model of the turbines operating at Tokke power plant in Norway. The Francis turbine has a splitter blade runner, which includes 30 blades. The runner outlet diameter is 0.349 m. The obtained maximum hydraulic efficiency of the turbine is 93.4% at the best efficiency point and the uncertainty was +/- 0.16% [1]. The test rig is extensively used for model testing and for specific investigations such as rotor stator interaction, vortex rope, rotating stall with pump-turbine runner, water hammer, cavitation, *etc*. The open loop hydraulic system is used to perform transient measurements such as load variation, start-stop, and total load rejection. Moreover, strict guidelines are used for the calibration of each instrument utilized during the measurements. Complete history of the calibrated instruments is maintained for reference study in order to observe any deviation of the characteristics over time.

As in the Onera M6 validation case, although experimental data exists, it will not be used

for the validation of IBM simulations results. Only the results obtained using a body-fitted mesh simulation will be used to validate IBM.

5.3.1 Case Setup

The simulations were performed for several different guide vanes positions. One of the roles of the guide vanes is to regulate the flow of water through the turbine, and in that way, the power of the turbine and all the associated parameters (efficiency, hydraulic head *etc*.). The body-fitted mesh for a certain guide vane position is shown in figures 5.3.3 - 5.3.5. The white parts of the mesh represent the mesh surrounding the guiding vanes, the blue parts of the mesh represent the rotor part and the red-colored parts represent the draft tube. The mesh cell distribution can be seen in figures 5.3.6 - 5.3.8.

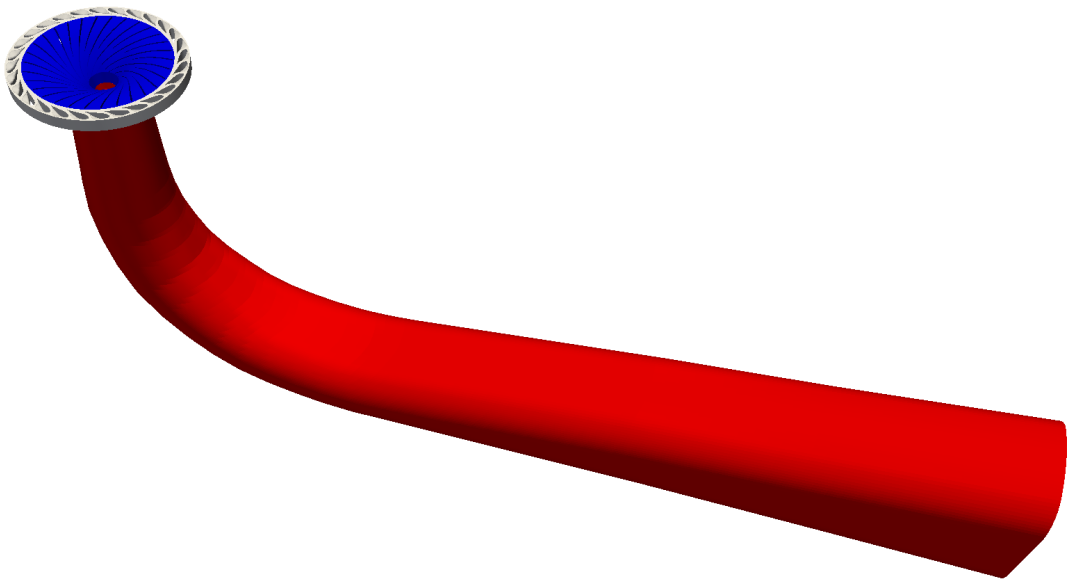


Figure 5.3.3: Full geometry Francis turbine

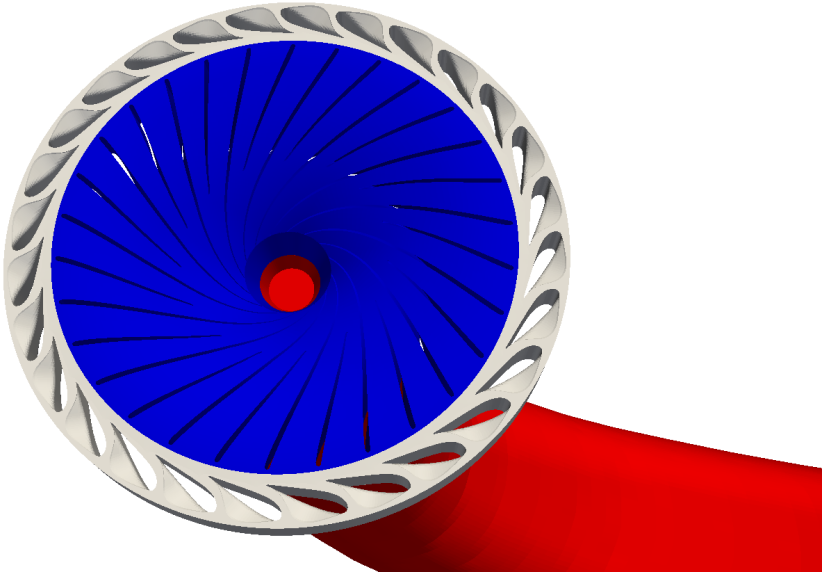


Figure 5.3.4: Full geometry Francis turbine close up

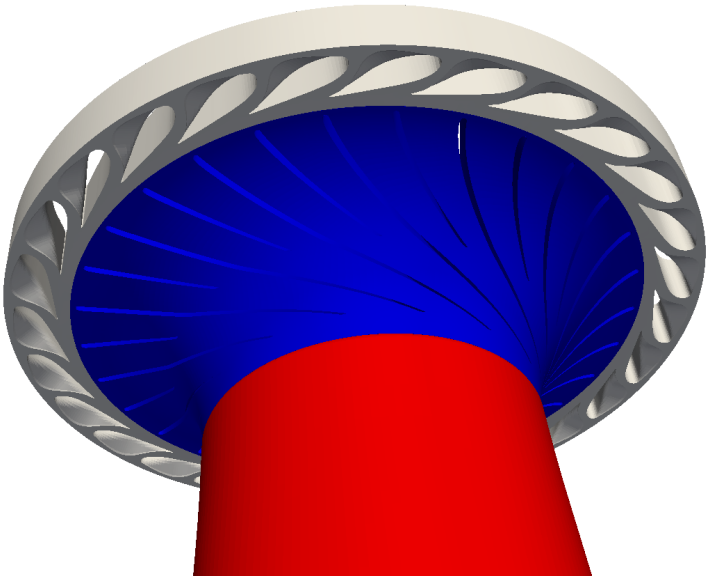


Figure 5.3.5: Full geometry Francis turbine close up, alternate angle

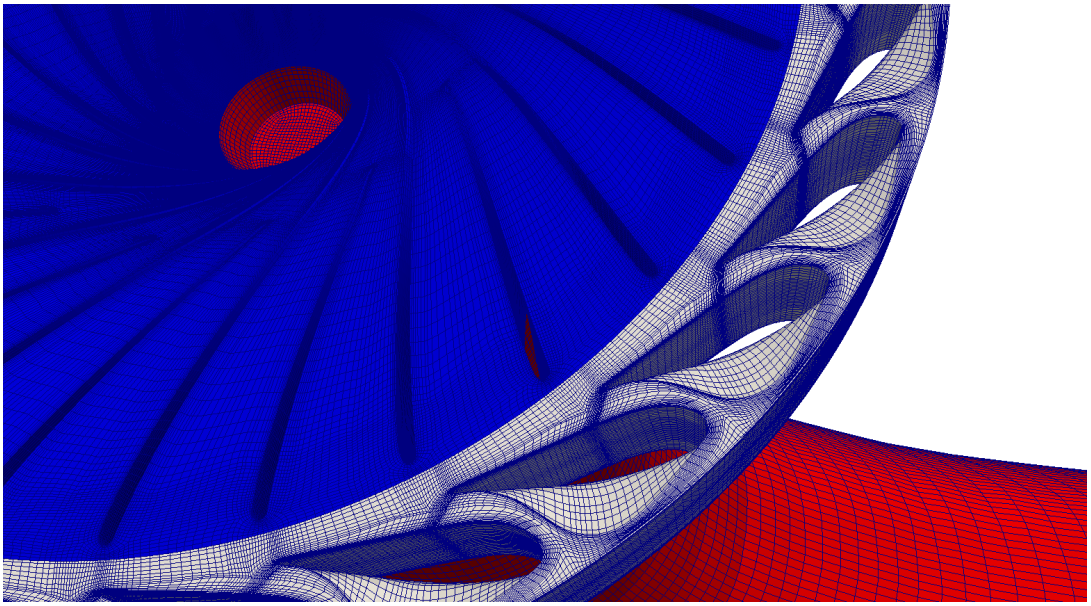


Figure 5.3.6: Full geometry Francis turbine body-fitted mesh close up

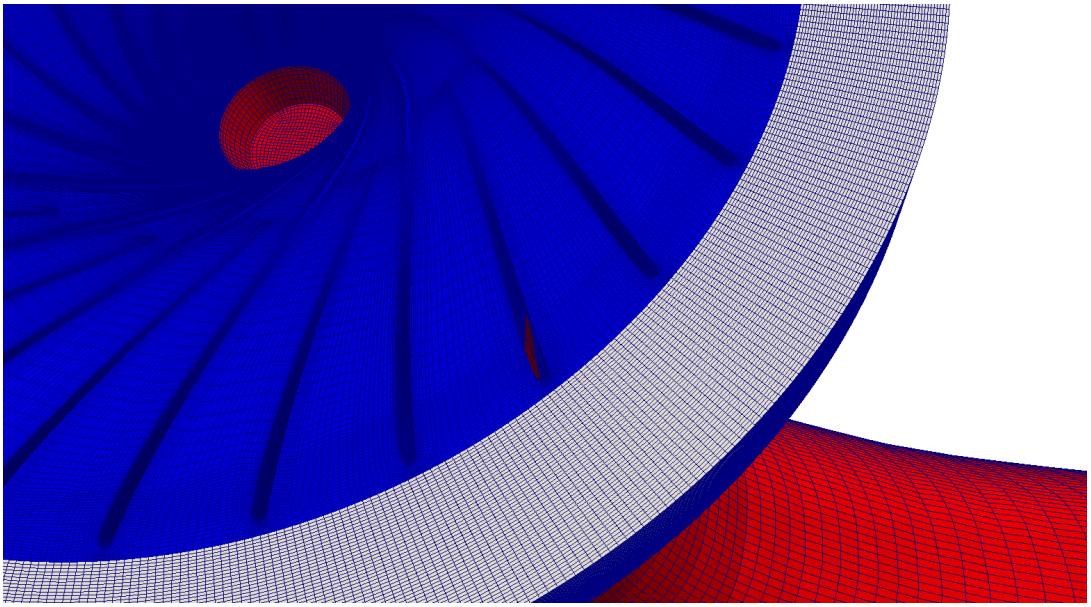


Figure 5.3.7: Full geometry Francis turbine IBM mesh close up

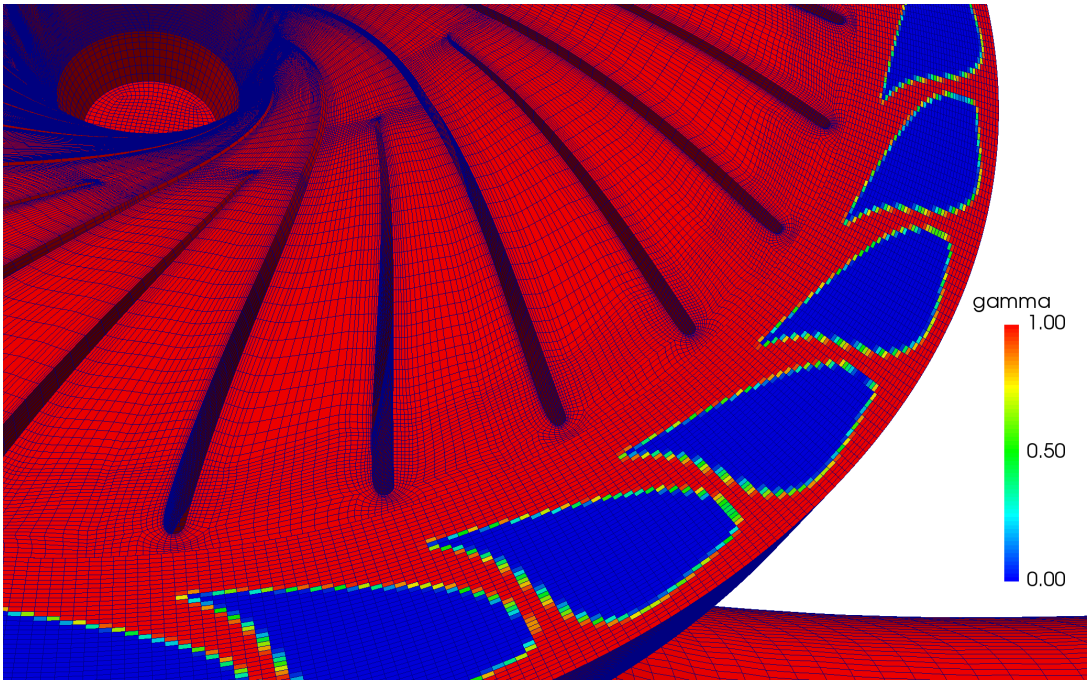


Figure 5.3.8: Full geometry Francis turbine IBM mesh close up, gamma field visualization

Several guide vane positions were simulated. For one of them (the simulation with the maximum flow of 0.1 m^3/s), the full geometry of the turbine was used. For the other three guide vanes positions (flows of 0.011 m^3/s, 0.055 m^3/s and 0.091 m^3/s), only a part of the geometry was used, specifically, 1/14 of the mesh, with a cyclicGGI boundary condition. The part geometry case is shown in figures 5.3.9 - 5.3.11. The white parts represent the mesh surrounding the guiding vanes, the blue part of the mesh is the rotor part and the red-colored parts represent the draft tube.

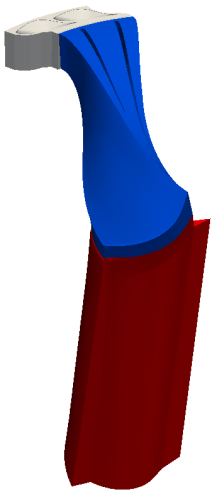


Figure 5.3.9: Part geometry Francis turbine

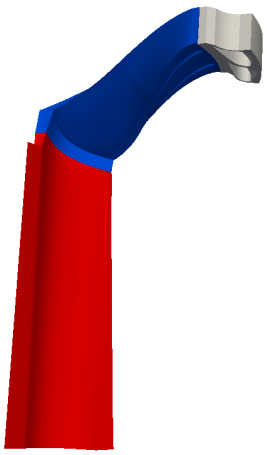


Figure 5.3.10: Part geometry Francis turbine, alternate angle

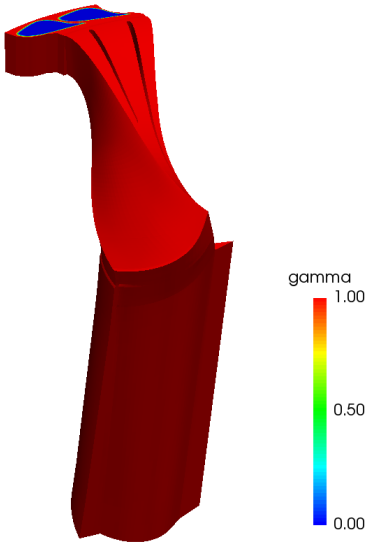


Figure 5.3.11: Part geometry Francis turbine IBM gamma field visualization

The body-fitted meshes were obtained from a previous master's thesis student's work [21]. For the IBM mesh simulations, a part of the body-fitted mesh was used (rotor and draft tube), while a part (guide vanes region) was meshed seperately using Pointwise and additionally merged to the rest of the mesh with the mergeMeshes utility. The interaction between different

mesh regions is accomplished using GGI, both in the body-fitted and immersed boundary mesh simulations.

	field				
patch	**p** [m^2s^{-2}]	**U** [ms^{-1}]	**k** [m^2s^{-2}]	**omega** [s^{-1}]	**nut** [m^2s^{-1}]
GVinlet	zeroGradient	cylindrical- NormalVelocity refValue- uniform (-1.4123 -2.1185 0)	fixedValue value uniform 0.0521	fixedValue value uniform 52.1	calculated
GVsupport	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
GV	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
GVIB	mixedIb triGradient uniform 0 setDeadValue no	mixedIb triValue uniform (0 0 0) setDeadValue yes	immersedBoundary- KqRWallFunction setDeadValue yes	immersedBoundary- OmegaWallFunction setDeadValue yes	immersedBoundary- NutWallFunction setDeadValue yes
GVoutlet	ggi	ggi	ggi	ggi	ggi
RUinlet	ggi	ggi	ggi	ggi	ggi
RUhub	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
RUsupport	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
RU	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
RUoutlet	ggi	ggi	ggi	ggi	ggi
DTinlet	ggi	ggi	ggi	ggi	ggi
DTwall	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
DToutlet	fixedValue value uniform 0	zeroGradient	zeroGradient	zeroGradient	calculated

Table 5.3.1: Francis full geometry boundary conditions

	field				
patch	**p** [m^2s^{-2}]	**U** [ms^{-1}]	**k** [m^2s^{-2}]	**omega** [s^{-1}]	**nut** [m^2s^{-1}]
GVinlet	zeroGradient	cylindrical- NormalVelocity refValue- uniform (-1.4123 -2.1185 0)	fixedValue value uniform 0.0521	fixedValue value uniform 52.1	calculated
GVsupport	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
GV	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
GVIB	mixedIb triGradient uniform 0 setDeadValue no	mixedIb triValue uniform (0 0 0) setDeadValue yes	immersedBoundary- KqRWallFunction setDeadValue yes	immersedBoundary- OmegaWallFunction setDeadValue yes	immersedBoundary- NutWallFunction setDeadValue yes
GVperiodic0	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI
GVperiodic1	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI
GVoutlet	ggi	ggi	ggi	ggi	ggi
RUinlet	ggi	ggi	ggi	ggi	ggi
RUhub	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
RUsupport	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
RU	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
RUperiodic0	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI
RUperiodic1	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI
RUoutlet	ggi	ggi	ggi	ggi	ggi
DTinlet	ggi	ggi	ggi	ggi	ggi
DTwall	zeroGradient	fixedValue value uniform (0 0 0)	kqRWallFunction	omegaWallFunction	nutkWallFunction
DTperiodic0	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI
DTperiodic1	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI	cyclicGGI
DToutlet	fixedValue value uniform 0	zeroGradient	zeroGradient	zeroGradient	calculated

Francis Turbine IBM Specifics

The Francis turbine case is especially interesting for usage of IBM. For the guide vanes position which corresponds to the smallest turbine flow of 0.011 m^3/s, the guide vanes are so close to each other that they are almost touching one another. This can result in certain problems related to mesh quality. This is shown in figures 5.3.12 - 5.3.19. If we compare figures 5.3.14 and 5.3.18, we can see that, by bringing the guide vanes closer together, the quality of the cells in the space between the blades is decreasing in quality (skewness, aspect ratio, *etc*.). Not only

that, but in some cases of similar nature, cells could even overlap if the walls were brought too close to one another.

This problem can be avoided by using IBM, as the IBM background mesh, even for moving cases, remains unchanged for all positions of the simulated object, as is seen in 5.3.15 and 5.3.19.

In cases like this, the aformentioned problem could even make it impossible to solve cases with significant geometry variations. $E.g.$, here, the solution to that problem while using BF meshing could be to set the inlet of the simulation domain between the guide vanes and the rotor region, as opposed to setting it upstream of the guide vanes (which was done in the simulations in this thesis). That way, the pressure drop in the guide vanes region would not be simulated, which has a big effect on the total simulated pressure drop in the turbine. In the simulations in this thesis, the stay vanes region was also not simulated. However, that has a smaller effect on the final results than not using the guide vanes region, as the stay vanes are fixed and the stay vanes pressure drop varies only slightly with change of turbine flow, while the variation of guide vanes pressure drop is not so small with change of turbine flow.

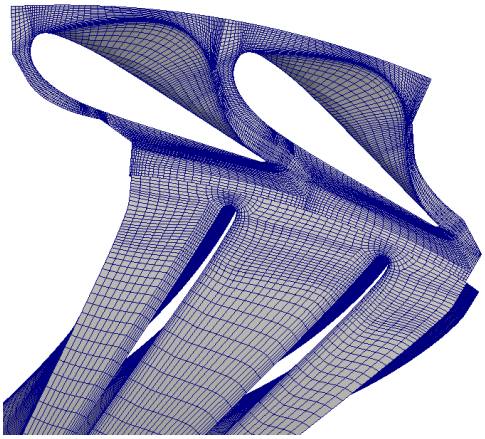


Figure 5.3.12: Francis turbine BF partial geometry guide vanes region mesh, 0.091 m^3/s flow simulation

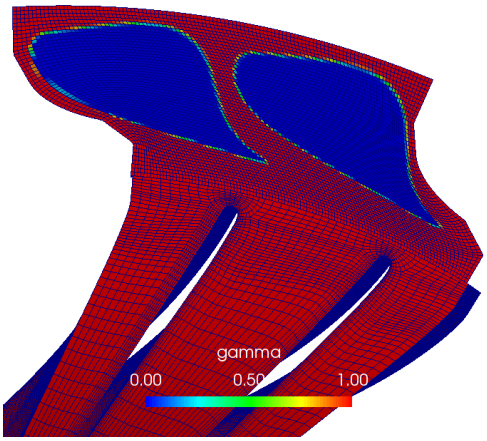


Figure 5.3.13: Francis turbine IBM partial geometry guide vanes region mesh, 0.091 m^3/s flow simulation

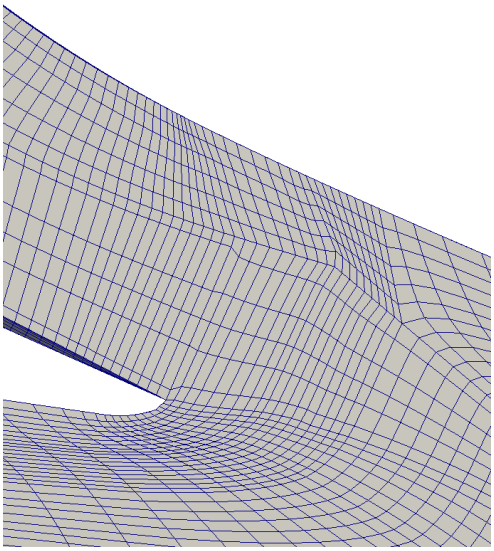


Figure 5.3.14: Francis turbine BF partial geometry guide vanes region mesh, 0.091 m^3/s flow simulation, close up of blade tip

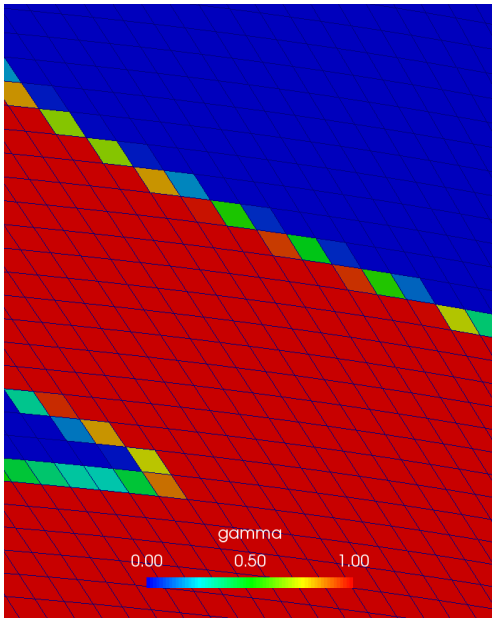


Figure 5.3.15: Francis turbine IBM partial geometry guide vanes region mesh, 0.091 m^3/s flow simulation, close up of blade tip

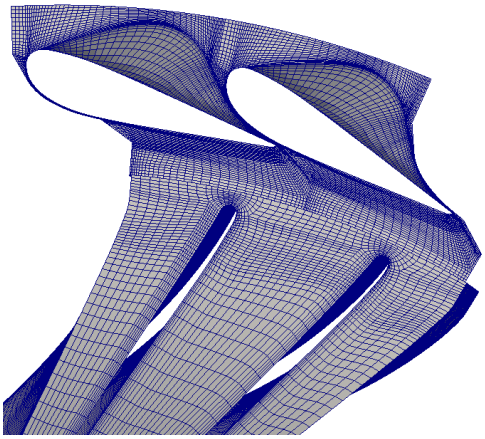


Figure 5.3.16: Francis turbine BF partial geometry guide vanes region mesh, 0.011 m^3/s flow simulation

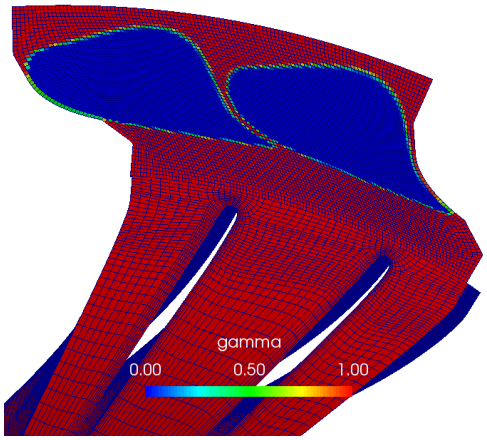


Figure 5.3.17: Francis turbine IBM partial geometry guide vanes region mesh, 0.011 m^3/s flow simulation

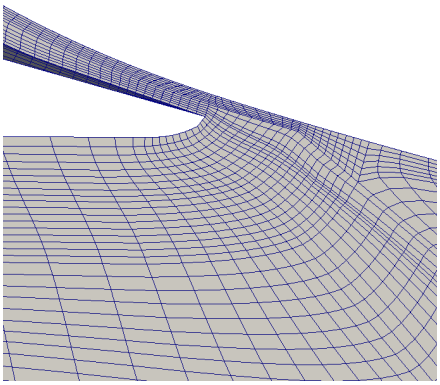


Figure 5.3.18: Francis turbine BF partial geometry guide vanes region mesh, 0.011 m^3/s flow simulation, close up of blade tip

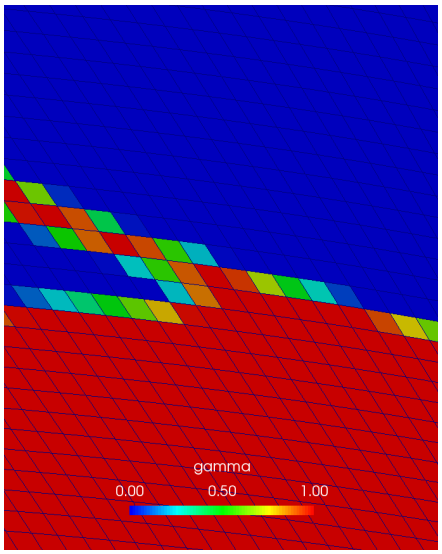


Figure 5.3.19: Francis turbine IBM partial geometry guide vanes region mesh, 0.011 m^3/s flow simulation, close up of blade tip

Simulation Settings

The simulations were performed using the $k-\omega$ *SST* turbulence model. Fluid properties which were used in the simulation correspond to water. potentialFoam solver was used to initialize the flow field. MRFsimpleFoam solver (with one rotating region, the rotor region) was used to obtain a steady-state solution, in 8000 non-linear iterations, which was enough for the simulations to converge.

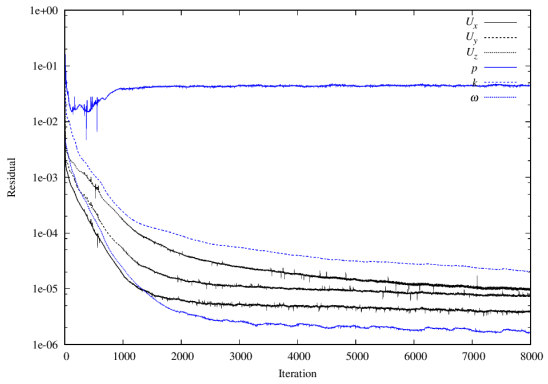


Figure 5.3.20: Francis turbine BF convergence history, 0.055 m^3/s flow partial geometry simulation

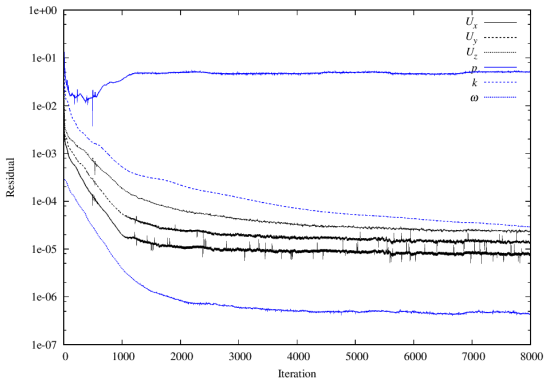


Figure 5.3.21: Francis turbine IBM convergence history, 0.055 m^3/s flow partial geometry simulation

5.3.2 Results and Validation

The results will be presented qualitatively, using figures which portray the resulting scalar and vector fields of the simulations (figures 5.3.22 - 5.3.35), and quantitatively, by comparing values which are the usual sought after values of simulating flow in a turbine: efficiency, power and hydraulic head of the turbine (figures 5.3.36 - 5.3.38). The efficiencies, powers and the hydraulic heads in the simulations were calculated using the turboPerformance library for OpenFOAM. The figures which show the obtained results from the full geometry simulation were performed with a certain guide vanes position, with an inflow of 0.1 m^3/s. The figures which show the obtained results from a part geometry simulation are from a simulation which was performed with a certain guide vanes position, with an inflow of 0.055 m^3/s. The simulations with different guide vanes positions and inflows show comparable results, in a sense that body-fitted mesh and IBM mesh simulations give similar values.

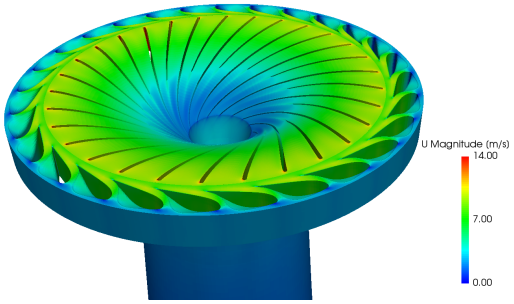


Figure 5.3.22: Full geometry Francis turbine BF velocity field visualization, 0.1 m^3/s turbine flow simulation

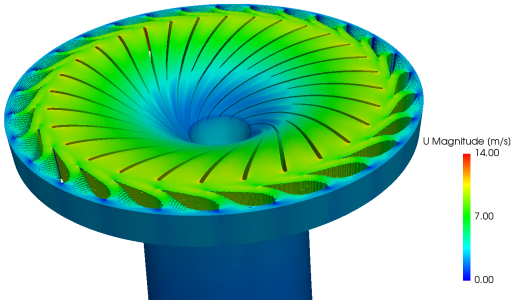


Figure 5.3.23: Full geometry Francis turbine IBM velocity field visualization, 0.1 m^3/s turbine flow simulation

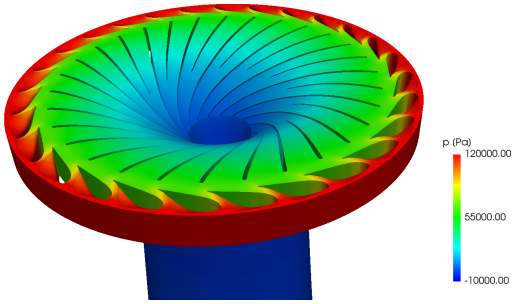


Figure 5.3.24: Full geometry Francis turbine BF pressure field visualization, 0.1 m^3/s turbine flow simulation

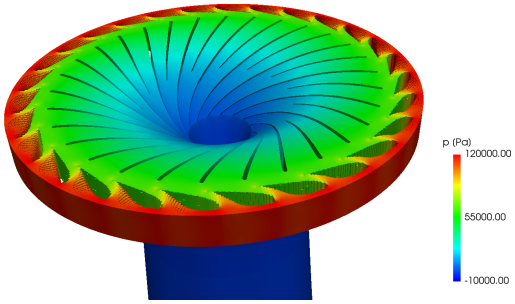


Figure 5.3.25: Full geometry Francis turbine IBM pressure field visualization, 0.1 m^3/s turbine flow simulation

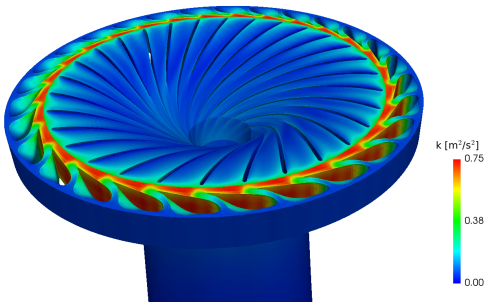


Figure 5.3.26: Full geometry Francis turbine BF turbulence kinetic energy field visualization, 0.1 m^3/s turbine flow simulation

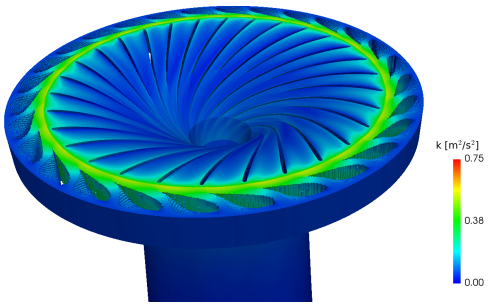


Figure 5.3.27: Full geometry Francis turbine IBM turbulence kinetic energy field visualization, 0.1 m^3/s turbine flow simulation

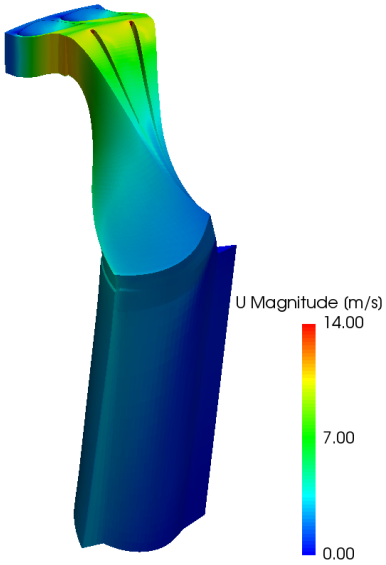


Figure 5.3.28: Part geometry Francis turbine BF velocity field visualization, 0.055 m^3/s turbine flow simulation

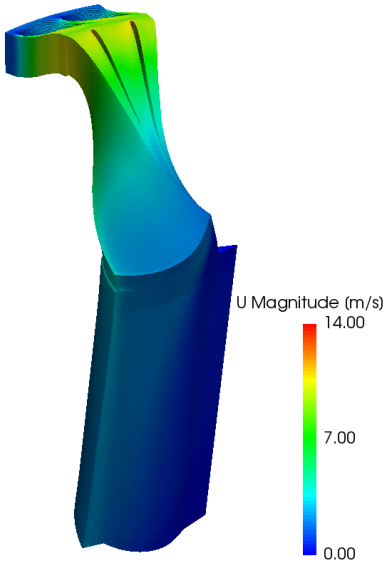


Figure 5.3.29: Part geometry Francis turbine IBM velocity field visualization, 0.055 m^3/s turbine flow simulation

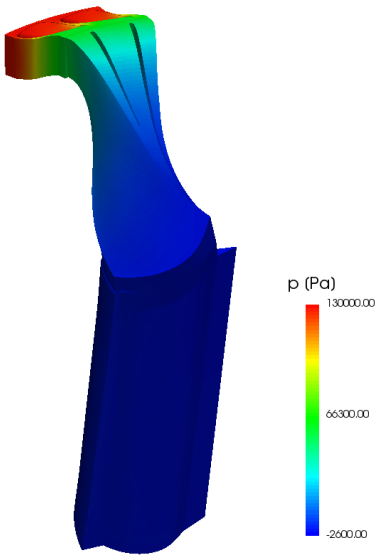


Figure 5.3.30: Part geometry Francis turbine BF pressure field visualization, 0.055 m^3/s turbine flow simulation

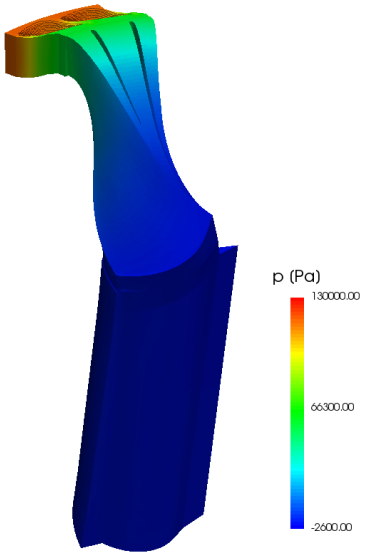


Figure 5.3.31: Part geometry Francis turbine IBM pressure field visualization, 0.055 m^3/s turbine flow simulation

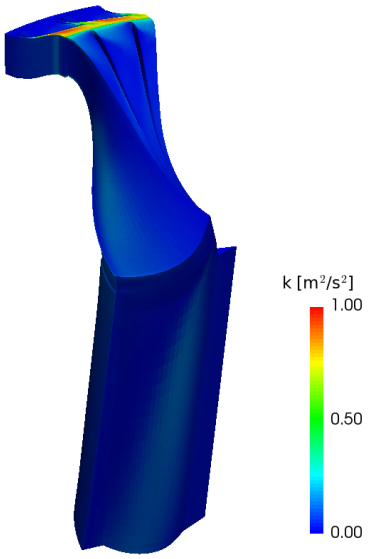


Figure 5.3.32: Part geometry Francis turbine BF turbulence kinetic energy field visualization, 0.055 m^3/s turbine flow simulation

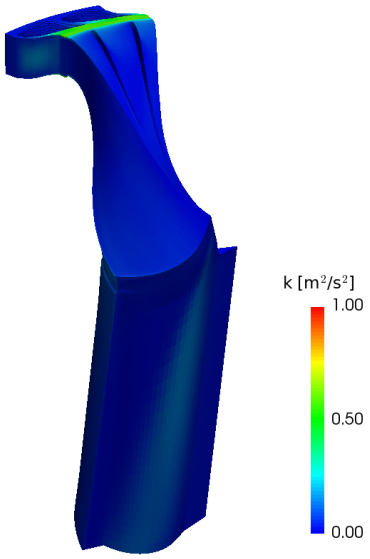


Figure 5.3.33: Part geometry Francis turbine IBM turbulence kinetic energy field visualization, 0.055 m^3/s turbine flow simulation

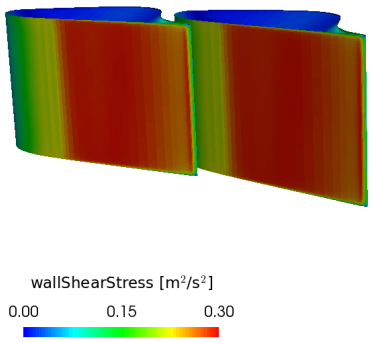


Figure 5.3.34: Part geometry Francis turbine BF wall shear stress field visualization, 0.055 m^3/s turbine flow simulation

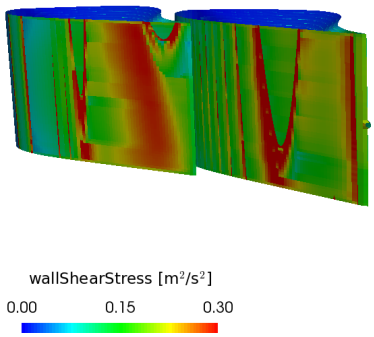


Figure 5.3.35: Part geometry Francis turbine IBM wall shear stress field visualization, 0.055 m^3/s turbine flow simulation

The wall shear stress visualization in 5.3.34 and 5.3.35 shows the same kind of discrepancy between BF and IBM results as in the Onera M6 wing simulations. The resulting wall shear stress field discrepancy is one of the causes of the force coefficients differences between BF and IBM simulations.

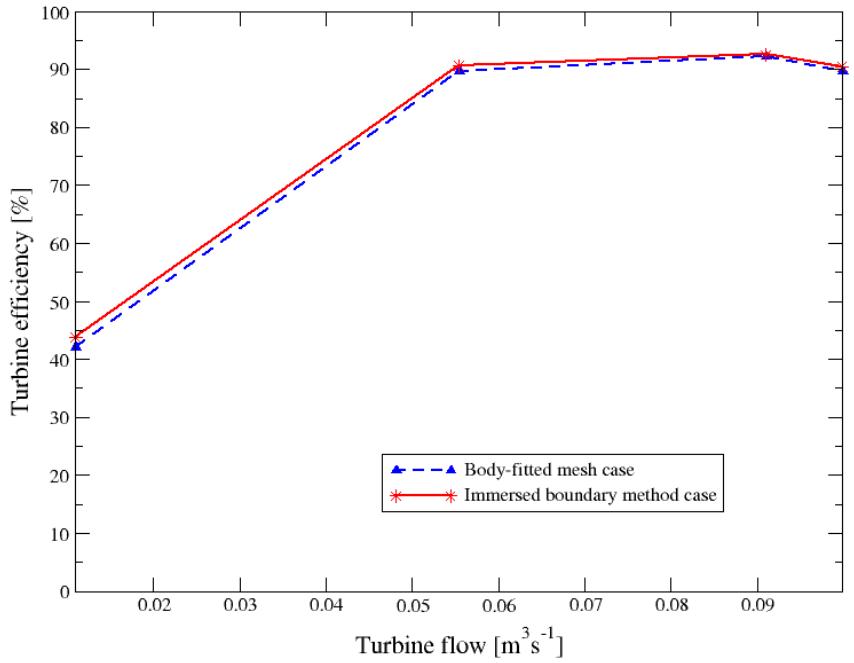


Figure 5.3.36: Francis turbine efficiency - flow curve

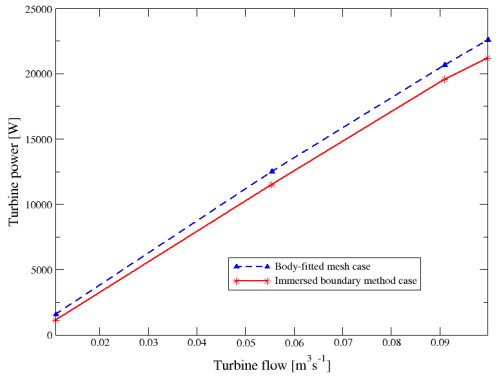


Figure 5.3.37: Francis turbine power - flow curve

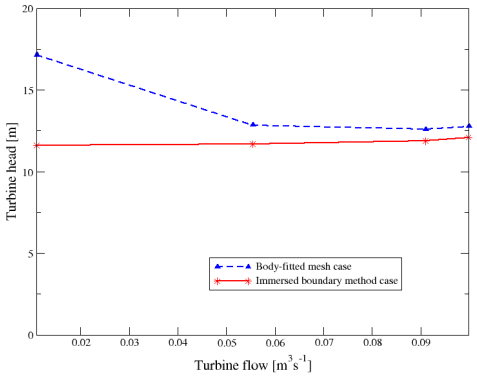


Figure 5.3.38: Francis turbine head - flow curve

The y^+ values on the guide vanes area are shown in 5.3.39 and 5.3.40. The red parts are the cells where y^+ is larger than 30, which is the minimal required value when using wall functions. The blue parts are the ones where y^+ is smaller than 30. The y^+ values are additionally discussed in a following section.

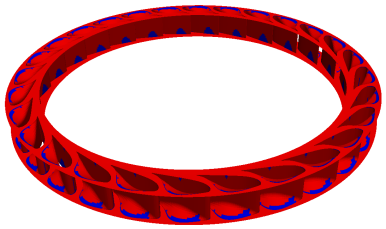


Figure 5.3.39: Francis turbine BF y^+ values visualization

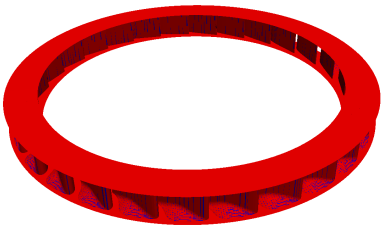


Figure 5.3.40: Francis turbine IBM y^+ values visualization

5.3.3 Closure

IBM simulations of the Francis turbine show good results in comparison to BF simulations with the same input parameters. That is generally true, except for the hydraulic head comparison of the 0.011 m^3/s flow simulations, where the guide vanes are in the most closed position. The agreement of the resulting fields can be observed in figures 5.3.22 - 5.3.33. The agreement of resulting turbine efficiencies, powers and hydraulic heads can be observed in figures 5.3.36 - 5.3.38, where it is shown that body-fitted and IBM simulations produce similar results. The turbine efficiencies are generally in very good agreement (a difference of about 4% for the 0.011 m^3/s simulations and of about 0.5 - 1% for the rest), as well as turbine powers (a difference of about 30% for the 0.011 m^3/s simulations and of about 5-8% for the rest) and hydraulic heads (a difference of about 32% for the 0.01 m^3/s simulation and of about 5-9% for the rest).

5.4 y^+ Considerations

During the process of setting up and performing the validation cases, y^+ was taken into consideration while creating the meshes. The turbulence boundary conditions which were used correspond to the wall function boundary layer approach. That means that y^+ values in the cell centers adjacent to the walls are supposed to be in the range of 30 - 300. That potentially represents a problem with IBM because, compared to body-fitted meshing, it is harder to control the distance between the walls and the first cell centers. That problem was most accentuated with the Onera M6 wing case, due to a very thin trailing edge. That problem is shown in figures 5.4.1

and 5.4.2. In figure 5.4.1, the mesh is finer than in 5.4.2 and because of that can better capture the surface geometry of the wing, although that mesh results with bad y^+ values. Figure 5.4.2 shows a coarser mesh, which results in a better y^+ field on the wing, but can not describe the surface geometry with IBM as good as the first mesh. The y^+ field values are shown in 5.4.3 - 5.4.5. Red parts of the wing surface correspond to a y^+ in the range of 30 - 300, while the blue parts correspond to y^+ of less than 30. With the wall function approach, there is often a part of the domain, near a stagnation point, that does not fulfil the $y^+ > 30$ condition, but as long as most of the wall has a favourable y^+ value, the obtained results should be satisfactory. The y^+ field in 5.4.4 corresponds to the mesh shown in 5.4.1 and the one in 5.4.5 corresponds to the mesh shown in 5.4.2.

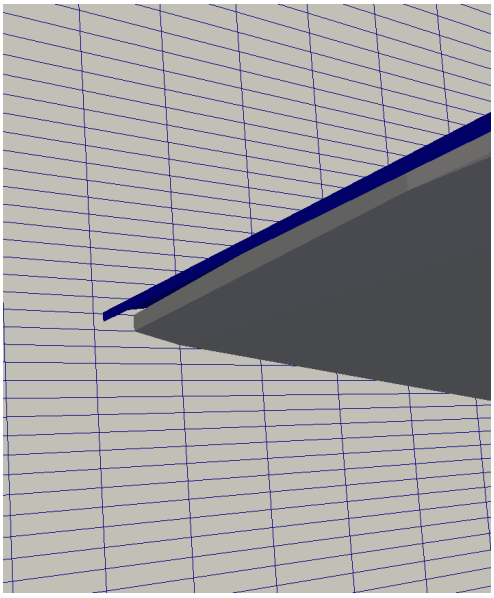


Figure 5.4.1: Onera bad y^+ trailing edge

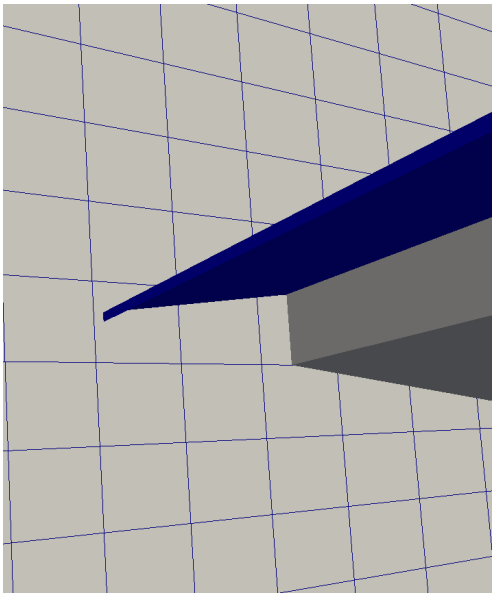


Figure 5.4.2: Onera good y^+ trailing edge

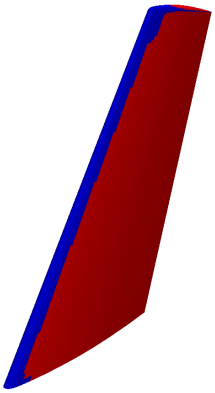


Figure 5.4.3: Body-fitted Onera M6 y^+ field on the wing

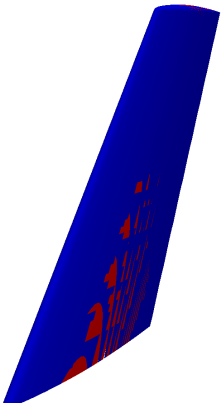


Figure 5.4.4: IBM Onera M6 y^+ field on the wing, bad y^+

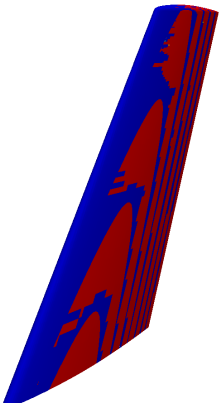


Figure 5.4.5: IBM Onera M6 y^+ field on the wing, good y^+

Although the mesh shown in 5.4.2 results in a better y^+ on the wing, the results obtained with that mesh are not as good as the ones obtained with the mesh in 5.4.1. For example, with the finer mesh, the discrepancy between the body-fitted and IBM results for a 5 degree simulation, is 3.9% for the drag coefficient C_D and 3.8% for the lift coefficient C_L. With the coarser mesh, those discrepancies are 11.9% and 7.0%, respectively. For that reason, the finer mesh was used for simulations in section 5.2.

For the backward facing step and Francis turbine cases, there were no major problems with the y^+ values on the walls. The backward facing step is a very simple and rectangular geometry, which makes the task of estimating the correct wall-adjacent cell size simple. The Francis turbine case is geometrically a very complex case, but has no edges which are as sharp as the trailing edge of the Onera M6 wing, which also makes it possible to estimate the wall-adjacent cell size correctly.

It should also be mentioned that, although the wall function approach was used in this thesis, an alternative exits. The boundary layer can be fully resolved, and then the wall-adjacent cell centers have to be in the y^+ range of less than 5, in the viscous sub-layer. This is a more expensive approach, as this can result in meshes with large cell counts. However, this could be beneficial for IBM, as the background meshes could be made very fine, with a large cell count in the region where the immersed boundary is inserted and that way the y^+ values could be guaranteed to be smaller than a certain value.

Chapter 6

Conclusion

In this thesis, the theory of the IBM version implemented in $foam-extend$ 4.1 was presented and it was validated on three different cases. The results are shown in Chapter 5 and are generally satisfactory.

The backward facing step internal flow simulations show satisfactory agreement of results of BF and IBM simulations, especially for laminar flow simulations. Due to its' simple geometry, it is also a case where y^+ restrictions are easily enforced when wall functions are used in turbulent flow simulations.

Simulations of external flow around the Onera M6 wing show very good agreement of BF and IBM simulation results for simulations with angles of attack of 0 - 8 degrees. For larger angles of attack (up to 18 degrees), the discrepancy of drag coefficients is in the range of 15 - 20 % and the dicrepancy of lift coefficients is approximately 5 %. This case proved to be problemating in terms of y^+ restrictions enforcement while using wall functions, due to its' sharp trailing edge.

Simulations of flow inside a Francis-type model turbine also show generally good agreement of BF and IBM simulation results (except in the simulation where the turbine flow was very small, with almost completely closed guide vanes). The discrepancy between resulting efficiencies is 1 % or less, while the discrepancies of resulting turbine powers and hydraulic heads was 5 - 9 %. This case, although geometrically quite complex, was not as problematic as the Onera M6 case in terms of y^+ restrictions enforcement while using wall functions, since no edges as sharp as the Onera M6's trailing edge are present.

However, the simulations which were performed are only a fraction of what can be done

with CFD. For that reason, further studies should be made to validate this IBM implementation for compressible flow, heat transfer, multiphase flow, *etc*. Some IBM validation work done at the Faculty of Mechanical Engineering and Naval Architecture will also be mentioned in the following section.

6.1 Other IBM Validation Work

There is a strong focus on development of new naval hydrodynamics CFD simulation methods at the Faculty of Mechanical Engineering and Naval Architecture. The $foam-extend$ 4.1 IBM version was also tested on cases of such nature. One of them will be presented here [11].

The mesh and the gamma field are shown in 6.1.1 - 6.1.2.

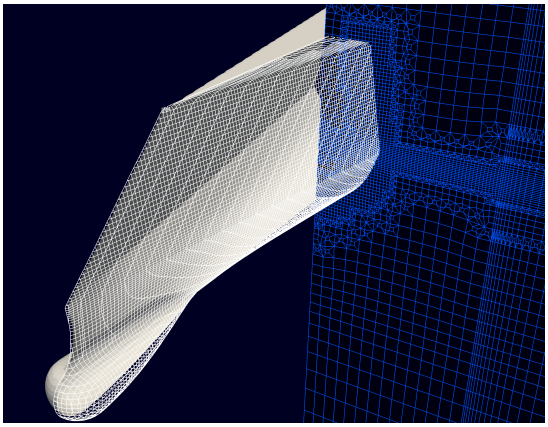


Figure 6.1.1: IBM mesh of the ship [11]

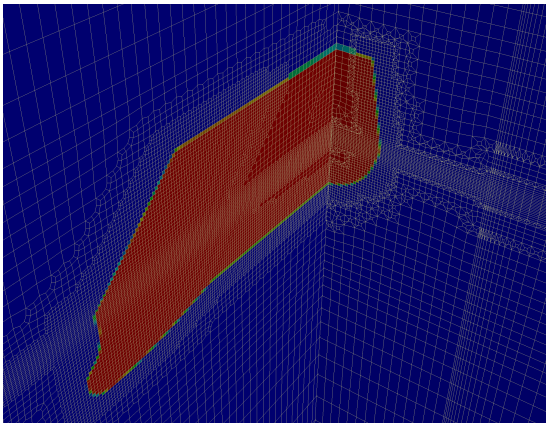


Figure 6.1.2: Ship gamma field [11]

The resulting dynamic pressure field on the ship surface is shown in 6.1.3 and the surrounding water surface elevation comparison is shown in 6.1.4. The difference of resulting ship resistance forces is 6.1% (the BF simulation results with a force of 98 kN and the IBM simulation results with a force of 92 kN).

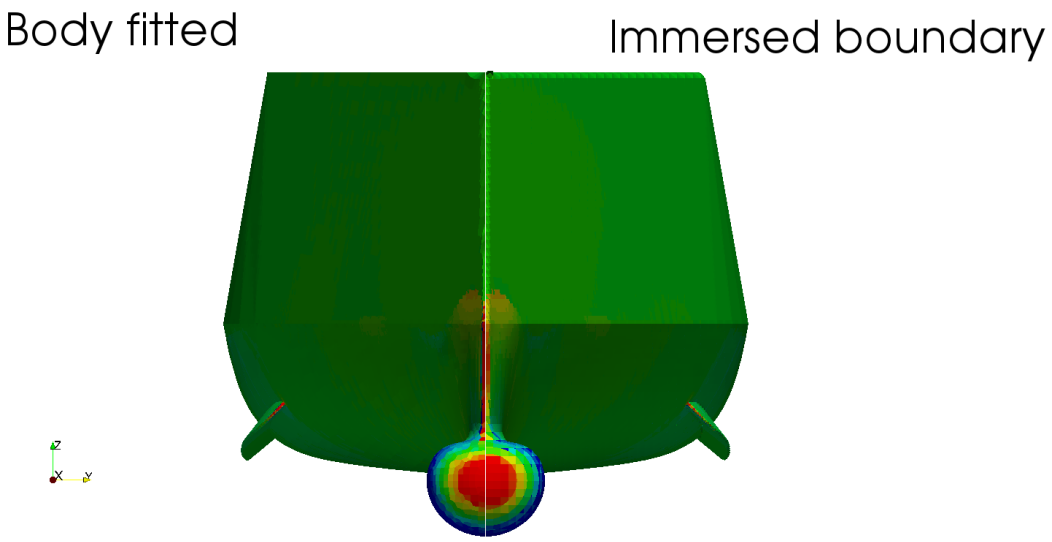


Figure 6.1.3: Dynamic pressure field on the surface of the ship [11]

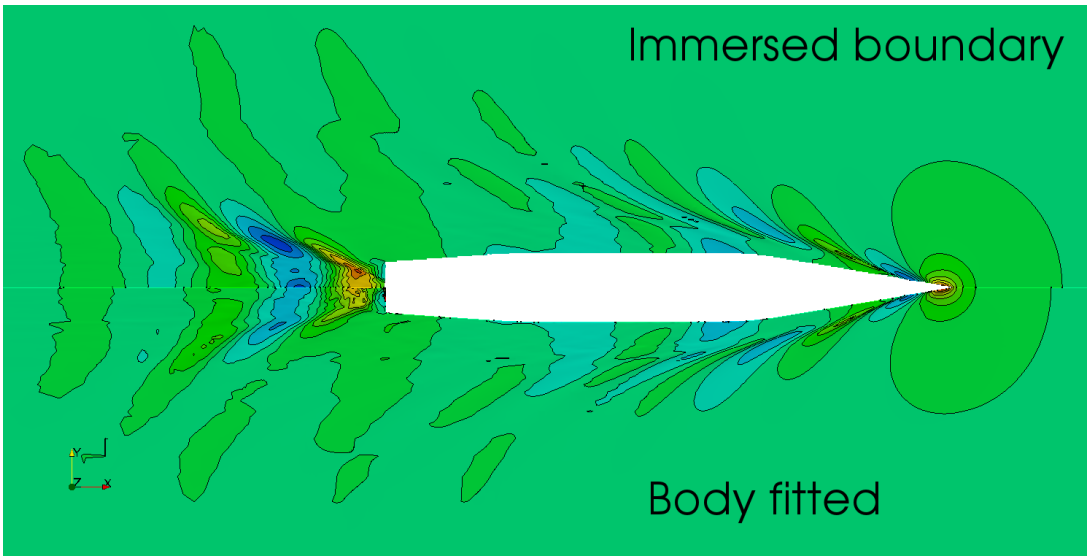


Figure 6.1.4: Surrounding water surface elevation comparison [11]

6.2 Closure

The validation work done in this thesis, as well as other work done to validate the *foam – extend* 4.1 IBM version [11] shows promising results, as it is implemented in a way that substantially simplifies the preprocessing of CFD simulations, while retaining a good enough level

of accuracy in most cases. Further work should be done to validate the IBM if it is to be expected for it to find its' place as a common tool for academic and industrial purposes.

Bibliography

[1] OpenCFD Ltd, "About OpenFOAM." `https://www.openfoam.com/`. Online; accessed 11 March 2019.

[2] H. Jasak, H. Rusche, B. Gschaider, H. Nilsson, M. Beaudoin, and V. Škurić, "foam-extend." `https://sourceforge.net/projects/foam-extend/`. Online; accessed 11 March 2019.

[3] F. Moukalled, L. Mangani, and M. Darwish, *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 2016.

[4] F. Menter and T. Esch, "Elements of industrial heat transfer prediction," 16th Brazilian Congress of Mechanical Engineering (COBEM), 2001.

[5] H. Jasak, *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. PhD thesis, Imperial College, London, 1996.

[6] S. V. Patankar and D. B. Spalding, "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows," *Int. J. Heat Mass Transfer*, vol. 15, pp. 1787–1806, 1972.

[7] M. Beaudoin and H. Jasak, "Development of a general grid interface for turbomachinery simulations with openfoam," Proceedings of Open Source CFD International Conference, Berlin, 2008.

[8] H. Jasak and M. Beaudoin, "Openfoam turbo tools: From general purpose cfd to turbomachinery simulations," vol. 1, 2011.

[9] R. Mittal and G. Iaccarino, "Immersed boundary methods," *Annu. Rev. Fluid Mech.*, vol. 37, pp. 239–261, 2005.

[10] H. Jasak, "New immersed boundary in foam-extend," Wikki Ltd., Faculty of Mechanical Engineering and Naval Architecture, 2018. 13th OpenFOAM Workshop, Shangai.

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

[12] C. S. Peskin, "Flow patterns around heart valves: A numerical method," *Journal of Computational Physics*, vol. 10, pp. 252–271, 1972.

[13] F. R. Menter, M. Kuntz, and R. Langtry, "Ten years of industrial experience with the sst turbulence model," in *Turbulence, Heat and Mass Transfer 4*, 2003.

[14] NASA, Langley Research Center, "3D ONERA M6 Wing Validation Case." `https://turbmodels.larc.nasa.gov/onerawingnumerics_val.html`. Online; accessed 18 January 2019.

[15] U. Şentürk, D. Brunner, H. Jasak, N. Herzog, C. Rowley, and A. Smits, "Benchmark simulations of flow past rigid bodies using an open-source, sharp interface immersed boundary method," *Progress in Computational Fluid Dynamics An International Journal*, vol. 1, 03 2018.

[16] V. Schmitt and F. Charpin, "Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers," *Experimental ata Base for Computer Program Assessment. Report of the Fluid Dynamics Panel Working Group 04*, 1979.

[17] OpenFOAM wiki, "See the MRF development." `http://openfoamwiki.net/index.php/See_the_MRF_development`. Online; accessed 31 January 2019.

[18] Pointwise, Inc., "Pointwise." `https://www.pointwise.com/`. Online; accessed 13 February 2019.

[19] Norwegian Hydropower Centre, "Francis-99." `https://www.ntnu.edu/nvks/francis-99`. Online; accessed 29 January 2019.

[20] C. Trivedi, M. Cervantes, B. Gandhi, and O. Dahlhaug, "Experimental and numerical studies for a high head francis turbine at several operating points," *Journal of Fluids Engineering*, vol. 135, p. 111102, 2013.

[21] L. Ćulić, "Validation of the harmonic balance solver for turbomachinery start-up and shut-down simulations," Master's thesis, Faculty of Mechanical engineering and Naval Architecture, University of Zagreb, Zagreb, 2018.

Appendix

Appendix 1: Usage Guidelines for IBM in $foam-extend$ 4.1

The basic case structure in OpenFOAM is composed of the following folders:

0 Folder

0 folder contains boundary and initial conditions for all the field variables. $E.g.$ the p file is composed of inputs which define the initial field values for pressure, as well as boundary conditions for all of the patches of the mesh.

For IBM, the structure is identical. To properly define the boundary and initial conditions on an immersed boundary, specific IBM boundary conditions are input here.

- Pressure p:

  To define a zero gradient boundary condition on the immersed boundary, the following syntax is used:

```
{
    type mixedIb;
    patchType immersedBoundary;
    triValue uniform 0;
    triGradient  uniform 0;
    triValueFraction uniform 0;
    setDeadValue   no;
}
```

- Velocity U:

  To define a zero velocity boundary condition on the immersed boundary, the following syntax is used:

```
{
    type mixedIb;
    patchType immersedBoundary;
    triValue uniform (0 0 0);
    triGradient uniform (0 0 0);
    triValueFraction uniform 1;
    setDeadValue yes;
    deadValue (0 0 0);
}
```

- Turbulence kinetic energy k:

  To define a *kqRWallFunction* boundary condition on the immersed boundary, the following syntax is used:

```
{
    type immersedBoundaryKqRWallFunction;
    patchType immersedBoundary;
    setDeadValue yes;
    deadValue "...";
}
```

- Eddy turnover time *omega*:

  To define an *omegaWallFunction* boundary condition on the immersed boundary, the following syntax is used:

```
{
    type immersedBoundaryOmegaWallFunction;
```

```
        patchType immersedBoundary;
        setDeadValue yes;
        deadValue "...";
    }
```

- Eddy viscosity rate *nut*:

  To define a *nutkWallFunction* boundary condition on the immersed boundary, the following syntax is used:

```
    {
        type immersedBoundaryNutWallFunction;
        patchType immersedBoundary;
        setDeadValue    yes;
        deadValue       "...";
    }
```

constant Folder

In this folder, the properties of the simulation which are constant through the simulation are found: the mesh, definition of the turbulence and transport models, *etc*.

For IBM, additionally there has to be a $triSurface$ folder in which is the ftr file of the immersed boundary surface. The ftr file is easily obtained from an *stl* file by using the $surfaceConvert$ utility in OpenFOAM. It should be noted that the user should be careful with the surface normals of the surface file: the surface normals should be pointing into the fluid region of the mesh.

Also, the immersed boundary has to be defined in the *boundary* file in $constant/polyMesh$. The immersed boundary patch description has to be the first patch in the list of boundaries and has to be written using the following syntax:

```
"immersedBoundaryPatchName"
{
    type immersedBoundary;
    nFaces 0;
```

```
        startFace "here goes the startFace value of the next patch";
        internalFlow "yes or no";
        isWall yes;
    }
```

It should also be noted that the field values on the immersed boundary (pressure, velocity, turbulence kinetic energy, *etc*.) can be visualized using the automatically generated *vtk* files that are located in the time folders of the simulation.

system Folder

In this folder, three mandatory files are found: *controlDict*, $fvSchemes$ and $fvSolution$. Their contents define the general parameters of the simulation: simulation time step size, number of time steps, discretization schemes, solvers which are used to solve the discretized equations, *etc*. For IBM, additional libraries have to be sourced in the *controlDict* file: *libimmersedBoundary.so* and *libimmersedBoundaryTurbulence.so*.

Appendix 2: Backward Facing Step fvSchemes and fvSolution

fvSchemes Settings

```
ddtSchemes
{
    default          steadyState;
}
gradSchemes
{
    default          cellLimited Gauss linear 1;
}
divSchemes
{
    default             none;
    div(phi,U)          Gauss vanLeerDC;
```

```
    div(phi,k)          Gauss upwind;
    div(phi,omega)      Gauss upwind;
    div((nuEff*dev(T(grad(U)))))  Gauss linear;
}
laplacianSchemes
{
    default          Gauss linear limited 0.5;
}
interpolationSchemes
{
    default          linear;
}
snGradSchemes
{
    default          limited 0.5;
}
```

fvSolution Settings

```
solvers
{
    p
    {
        solver           CG;
        preconditioner   Cholesky;
        minIter          1;
        maxIter          1000;
        tolerance        1e-06;
        relTol           0.01;
    }
    U
    {
```

```
        solver           BiCGStab;
        preconditioner   ILU0;
        minIter          0;
        maxIter          1000;
        tolerance        1e-08;
        relTol           0;
    }
    k
    {
        solver           BiCGStab;
        preconditioner   ILU0;
        minIter          1;
        maxIter          1000;
        tolerance        1e-08;
        relTol           0;
    }
    omega
    {
        solver           BiCGStab;
        preconditioner   ILU0;
        minIter          1;
        maxIter          1000;
        tolerance        1e-08;
        relTol           0;
    }
}
SIMPLE
{
    nNonOrthogonalCorrectors 2;
}
relaxationFactors
```

```
{
    equations
    {
        U           0.6;
        k           0.6;
        omega       0.6;
    }
    fields
    {
        p           0.3;
    }
}
```

Appendix 3: Onera M6 fvSchemes and fvSolution

fvSchemes Settings

```
ddtSchemes
{
default  steadyState;
}
gradSchemes
{
default  cellLimited Gauss linear 1;
}
divSchemes
{
default none;
div(phi,U)  Gauss upwind;
div(phi,k)  Gauss upwind;
div(phi,omega) Gauss upwind;
```

```
div((nuEff*dev(T(grad(U))))  Gauss linear;
}
laplacianSchemes
{
default  Gauss linear limited 0.5;
}
interpolationSchemes
{
default  linear;
}
snGradSchemes
{
default  limited 0.5;
}
fluxRequired
{
default  no;
p  ;
}
```

fvSolution Settings

```
solvers
{
    p
    {
        solver           CG;
        preconditioner   Cholesky;
        minIter          1;
        maxIter          1000;
        tolerance        1e-06;
        relTol           0.01;
```

```
    }
    "(U|k|omega)"
    {
        solver  BiCGStab;
        preconditioner  DILU;
        minIter         1;
        maxIter         1000;
        tolerance    1e-8;
        relTol    0;
    }
}
SIMPLE
{
    nNonOrthogonalCorrectors  1;
    pRefValue  0;
}
relaxationFactors
{
    fields
    {
        p 0.1;
    }
    equations
    {
        U 0.5;
        k 0.5;
        omega 0.5;
    }
}
```

Appendix 4: Francis Turbine fvSchemes and fvSolution

fvSchemes Settings

```
ddtSchemes
{
    default steadyState;
}
gradSchemes
{
    default     cellLimited leastSquares 1;
}


divSchemes
{
    default         none;
    div(phi,U)      Gauss  upwind;
    div(phi,k)      Gauss upwind;
    div(phi,omega)  Gauss upwind;
    div(phi,R)      Gauss upwind;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss upwind;
    div((nuEff*dev(grad(U).T()))) Gauss linear;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}
laplacianSchemes
{
    default         Gauss linear limited 0.5;
}
interpolationSchemes
{
    default         linear;
```

```
    interpolate(U)  linear;
}
snGradSchemes
{
    default         limited 0.5;
}
fluxRequired
{
    default         no;
    p;
    pcorr;
}
```

fvSolution Settings

```
solvers
{
    p
    {
        solver           BiCGStab;
        preconditioner   DILU;
        tolerance        1e-06;
        relTol           0.01;
        minIter          1;
        maxIter          300;
    };
    U
    {
        solver           BiCGStab;
        preconditioner   DILU;
        tolerance        1e-06;
        relTol           0;
```

```
        minIter          1;
    };
    k
    {
        solver           BiCGStab;
        preconditioner   DILU;
        tolerance        1e-06;
        relTol           0;
        minIter          1;
    };
    "(omega)"
    {
        solver           BiCGStab;
        preconditioner   DILU;
        tolerance        1e-06;
        relTol           0;
        minIter          1;
    };
}
SIMPLE
{
    nNonOrthogonalCorrectors 1;
    pRefCell                 0;
    pRefValue                0;
}
relaxationFactors
{
    fields
    {
        p  1.0;
    }
```

```
    equations
    {
        U               0.2;
        nuTilda         0.2;
        k    0.2;
        omega    0.2;
    }
}
fieldBounds
{
    U 50;
    p -1e2 2e2;
}
```