National Technical University of Athens

Laboratory of Aerodynamics
Department of Fluid Mechanics
School of Mechanical Engineering

Implementation of the Immersed Boundary Method for 2-D external flows

Dimitris Vlastos

Pregraduate Thesis

Supervisor

Spyros Voutsinas

Professor, National Technical University of Athens

Athens, April 2021

Acknowledgements

First and foremost i would like to thank Professor Spyros Voutsinas for giving me the opportunity to study this subject but more importantly for his willingness to share his knowledge and experience, thus introducing me to the world of Computational Fluid Dynamics. I would also like to express my gratitude towards Dr. Konstantinos Diakakis for his invaluable support, guidance and the amount of technical knowledge he shared with me. Additionally, i would like to thank the members of the Advisory Commitee, Professor Ioannis Anagnostopoulos, Associate Professor Vasileios Riziotis and Professor Spyros Voutsinas. Finally, i would like to thank my family for all their efforts enabling me to pursue my goals.

Contents

List of Figures

1 Introduction

The progress in computer technology has allowed Computational Fluid Dynamics to simulate large scale problems and complex flows with excellent accuracy and efficiency. The solution of the Navier Stokes Equations (NSE) can be achieved through the use of one of two methods. In the case of Grid-based Solvers, a computational grid is used and the flow is described by calculating its corresponding values at grid nodes. On the contrary, non Grid-based solvers use fluid particles as a means to describe the flow through their trajectories. The use of a Lagrangian, mesh-free solver has the advantages of not needing a grid in order to simulate the flow, and of being self-adaptive having zero numerical diffusion. Their ability to satisfy the far-field conditions with great precision makes them suitable for wake aerodynamic simulations. However, the implementation of wall boundary conditions becomes a major challenge involving large numbers of particles and costly convolution operations. In Aerodynamics the vast majority of problems is solved using Eulerian, grid-based solvers. The plethora of different problems and applications requires the use of different types of grids and methods in order to accurately represent body-fluid interactions. Towards this goal, wall boundary conditions, which represent the effect of the solid body, must be put in place and, depending on the compexity of the problem, the grid must be adequately dense in the near-wall regions. A Eulerian solver, although effective, requires the use of a body conformal mesh. For simple geometries this is a rather straightforward task. However, for complex geometries or moving bodies the generation of a computational grid can be time consuming and complicated. In the case of moving meshes, the use of the Immersed Boundary Method can speed up the solution procedure and eliminate the issues associated with regridding at each time-step, in the case of a moving mesh. In this regard, many researchers turn towards grid-based methods that do not require a conformal grid and, at the same time, comply with the necessary boundary conditions. Thus, an accurate simulation of the flow can be achieved with much simpler and less detailed grids. In the case of moving boundaries a typical approach with body-fitted grids combines the difficulties associated with unsteadiness, mesh movement, is hard to perform and costly in CPU time. The use of immersed boundary methods diminishes these problems as the movement of the geometry is achieved by relocalizing the embedded surfaces in the fixed computational mesh.

1.1 Thesis scope

The main scope of this thesis is to study the Immersed Boundary Method, using simple Cartesian or cylindrical meshes. The Immersed Boundary Method is a non body-fitted method in which the computational grid is not needed to be conformed to the real physical boundary. CFD computations were possible with the use of the Navier-Stokes solver MaPFlow developed by G. Papadakis [2]. The final goal is to enable MaPFlow to simulate flows both with the classical approach of body-fitted grids as well as with non body-conformal meshes using the Immersed Boundary Method. In this regard additions were made to the MaPFlow source code in order to enable it to distinguish between external and immersed cells of a domain. Two geometries were used, an airfoil (NACA 0012) and a circular cylinder. A Cartesian grid and body-fitted grids without

physical solid boundaries were used. The domain is then split into the external flow domain and the immersed or internal one. The solution of the system of equations depends on where a cell lies. In this thesis, the results presented in chapter 5 refer to both unsteady and steady simulations with different Reynolds numbers and mesh discretizations. The flow around the airfoil was calculated when the body is placed at an angle of 1^o.

1.2 Thesis outline

The thesis is divided into 6 chapters.

1. Chapter 1 gives the basic overview of the current state of CFD and of certain aspects which will be studied in this thesis.

2. Chapter 2 describes the MaPFlow Solver, which is the computational tool used in this thesis.

3. Chapter 3 presents the Immersed Boundary Method in detail.

4. Chapter 4 contains the results from the simulation of the flow around a cylinder using a body-fitted grid without a physical boundary.

5. Chapter 5 contains the results for a Cartesian non-body conformal mesh for the circular cylinder and the airfoil.

6. Chapter 6 contains some concluding remarks.

2 MaPFlow CFD Solver

2.1 Governing Equations

2.1.1 Conservative Form

Let D denote a volume of fluid and ∂D its boundary. By integrating the governing equations over D, the following integral form is obtained:

$$\int_D \frac{\partial \vec{U}}{\partial t} dD + \oint_{\partial D} (\vec{F_c} - \vec{F_v}) dS = \int_D \vec{Q} dD \tag{1}$$

$\vec{U}$ is the vector of the Conservative Flow Variables,

$$\vec{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} \tag{2}$$

where ρ denotes the density, (u, v, w) the three components of the velocity field and E the total energy.

$\vec{F_c}$ and $\vec{F_v}$ denote the Convective and Viscous Fluxes respectively,

$$\vec{F_c} = \begin{pmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho (E + \frac{p}{\rho}) V \end{pmatrix} \tag{3}$$

$$\vec{F_v} = \begin{pmatrix} 0 \\ n_x \tau_x x + n_y \tau_x y + n_z \tau_x z \\ n_x \tau_y x + n_y \tau_y y + n_z \tau_y z \\ n_x \tau_z x + n_y \tau_z y + n_z \tau_z z \\ n_x \Theta_x + n_y \Theta_y + n_z \theta_z \end{pmatrix} \tag{4}$$

where V is the normal velocity,

$$V = \vec{u} \cdot \vec{n} \tag{5}$$

and

$$\begin{aligned} \Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k\frac{\partial T}{\partial x} \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + k\frac{\partial T}{\partial y} \\ \Theta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + k\frac{\partial T}{\partial z} \end{aligned} \tag{6}$$

The above system is completed by the constitutive equation of state for perfect gases:

$$p = (\gamma - 1)\rho \left[E - \frac{u^2 + v^2 + w^2}{2} \right] \tag{7}$$

2.1.2 Variable Transformations

CFD solvers are formulated using the governing equations (1) written in primitive ($\vec{V}$) or characteristic ($\vec{V}_{ch}$) variables,

$$\vec{V} = \begin{pmatrix} \rho \\ u \\ v \\ w \\ p \end{pmatrix} \tag{8}$$

Primitive Variables Starting from the differential form of the governing equations,

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}_c}{\partial \vec{x}} - \frac{\partial \vec{F}_v}{\partial \vec{x}} = \frac{\partial \vec{Q}}{\partial \vec{x}} \tag{9}$$

by neglecting the viscous terms and using the chain rule:

$$\begin{aligned} \frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}_c}{\partial \vec{U}} \frac{\partial \vec{U}}{\partial \vec{x}} &= \frac{\partial \vec{Q}}{\partial \vec{x}} \\ \frac{\partial \vec{U}}{\partial t} + A_c \frac{\partial \vec{U}}{\partial \vec{x}} &= \frac{\partial \vec{Q}}{\partial \vec{x}} \end{aligned} \tag{10}$$

the Jacobian of the convective fluxes $A_c = \partial \vec{F}_c / \partial \vec{U}$ is derived.

By introducing the transformation matrix $M = \partial \vec{U} / \partial \vec{V}$, the system of equations is written in Primitive Variables form [3]:

$$\begin{aligned} \frac{\partial \vec{U}}{\partial \vec{V}} \frac{\partial \vec{V}}{\partial t} + A_c \frac{\partial \vec{U}}{\partial \vec{V}} \frac{\partial \vec{V}}{\partial t} &= \frac{\partial \vec{Q}}{\partial \vec{x}} \\ M \frac{\partial \vec{V}}{\partial t} + A_c M \frac{\partial \vec{V}}{\partial t} &= \frac{\partial \vec{Q}}{\partial \vec{x}} \\ \frac{\partial \vec{V}}{\partial t} + M^{-1} A_c M \frac{\partial \vec{V}}{\partial t} &= M^{-1} \frac{\partial \vec{Q}}{\partial \vec{x}} \\ \frac{\partial \vec{V}}{\partial t} + A_p \frac{\partial \vec{V}}{\partial t} &= \frac{\partial \vec{Q}_v}{\partial \vec{x}} \end{aligned} \tag{11}$$

The transformation matrix M is defined as:

$$M = \frac{\partial \vec{U}}{\partial \vec{V}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ u & \rho & 0 & 0 & 0 \\ v & 0 & \rho & 0 & 0 \\ w & 0 & 0 & \rho & 0 \\ \frac{u^2+v^2+w^2}{2} & \rho u & \rho v & \rho w & \frac{1}{\gamma - 1} \end{bmatrix} \tag{12}$$

and its inverse as:

$$M^{-1} = \frac{\partial \vec{V}}{\partial \vec{U}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ \frac{\gamma - 1}{2}\left(u^2 + v^2 + w^2\right) & -u(\gamma - 1) & -v(\gamma - 1) & -w(\gamma - 1) & \gamma - 1 \end{bmatrix} \tag{13}$$

(11) has the same form as (10) but the Jacobian of the convective fluxes, A_p, is expressed in primitive variables.

Characteristic Variables Diagonalization of A_c ($A_c = R\Lambda L$, where R, L contain the right and left eigenvectors respectively and Λ the eigenvalues) enables the transformation of (10) in characteristic variables. By that, decoupling of the system of equations is achieved.

$$\begin{aligned} \frac{\partial \vec{U}}{\partial t} + L^{-1}\Lambda L \frac{\partial \vec{U}}{\partial \vec{x}} &= \frac{\partial \vec{Q}}{\partial \vec{x}} \\ L\frac{\partial \vec{U}}{\partial t} + \Lambda L \frac{\partial \vec{U}}{\partial \vec{x}} &= L\frac{\partial \vec{Q}}{\partial \vec{x}} \end{aligned} \tag{14}$$

By defining $M_{ch} \equiv \partial \vec{U}/\partial \vec{V}_{ch} = L$ the decoupled system is acquired:

$$\frac{\partial \vec{V}_{ch}}{\partial t} + \Lambda \frac{\partial \vec{V}_{ch}}{\partial \vec{x}} = \frac{\partial \vec{Q}_{ch}}{\partial \vec{x}} \tag{15}$$

with:

$$\Lambda = \begin{bmatrix} V & 0 & 0 & 0 & 0 \\ 0 & V & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & V+c & 0 \\ 0 & 0 & 0 & 0 & V-c \end{bmatrix} \tag{16}$$

Depending on the variables chosen the diagonalization of the Jacobian Matrix (A_c for conservative variables,A_p for primitive) will lead to different eigenvectors. However, the eigenvectors can be transformed to the desired variables using the appropriate transformation matrix. For example in between primitive and characteristic variables the following hold:

$$A_p = M^{-1}A_cM = \left(M^{-1}R\right)\Lambda\left(LM\right) = R_p\Lambda L_p \tag{17}$$

where $R_p = M^{-1}R$ and $L_p = LM$ are the right and left eigenvectors in primitive variables. Using the inverse transformation matrix enables the transformation from primitive eigenvectors to conservative eigenvectors. The right eigenvectors in primitive variables are :

$$R_p = \begin{bmatrix} n_x & 0 & n_z & -n_y & -\frac{n_x}{c^2} \\ n_y & -n_z & 0 & n_x & -\frac{n_y}{c^2} \\ n_z & n_y & -n_x & 0 & -\frac{n_z}{c^2} \\ 0 & n_x & n_y & n_z & -\frac{\lambda_1-\lambda_4}{\rho c^2} \\ 0 & -n_x & -n_y & -n_z & \frac{\lambda_1-\lambda_5}{\rho c^2} \end{bmatrix} \tag{18}$$

where $\vec{n} = (n_x, n_y, n_z)$ is the unit normal vector, $\lambda_1 = \lambda_2 = \lambda_3 = V$, $\lambda_4 = V + c$ and $\lambda_5 = V - c$.
The left eigenvectors in primitive variables are :

$$L_p = R_p^{-1} = \begin{bmatrix} n_x & n_y & n_z & \frac{\rho}{\lambda_4-\lambda_5} & \frac{\rho}{\lambda_4-\lambda_5} \\ 0 & -n_z & n_y & \frac{\lambda_1-\lambda_5}{\lambda_4-\lambda_5}n_x & \frac{\lambda_1-\lambda_4}{\lambda_4-\lambda_5}n_x \\ n_z & 0 & -n_x & \frac{\lambda_1-\lambda_5}{\lambda_4-\lambda_5}n_y & \frac{\lambda_1-\lambda_4}{\lambda_4-\lambda_5}n_y \\ -n_y & n_x & 0 & \frac{\lambda_1-\lambda_5}{\lambda_4-\lambda_5}n_z & \frac{\lambda_1-\lambda_4}{\lambda_4-\lambda_5}n_z \\ 0 & 0 & 0 & \frac{\rho c^2}{\lambda_4-\lambda_5} & \frac{\rho c^2}{\lambda_4-\lambda_5} \end{bmatrix} \tag{19}$$

2.1.3 Low Mach Number Preconditioning

In cases where the local Mach number approaches zero there is large disparity in the wave propagation speeds. The speed of sound (c) becomes very large compared to the flow velocity (V) and completely deteriorates the stability and convergence properties of the system. In such cases the equations should be modified. Low Mach Preconditioning is applied that acts on the time derivatives of the equations and basically modifies the speed of sound to make the two velocities comparable. By that, the convergence and stability characteristics of the system are improved.

Neglecting the viscous terms, and using the Preconditioning Matrix Γ, the system of equations (10) takes the following general form:

$$\begin{aligned} \Gamma^{-1}\frac{\partial \vec{U}}{\partial t} + A_c\frac{\partial \vec{U}}{\partial \vec{x}} &= \frac{\partial \vec{Q}}{\partial \vec{x}} \Rightarrow \\ \frac{\partial \vec{U}}{\partial t} + \Gamma A_c\frac{\partial \vec{U}}{\partial \vec{x}} &= \Gamma\frac{\partial \vec{Q}}{\partial \vec{x}} \Rightarrow \\ \frac{\partial \vec{U}}{\partial t} + A_\Gamma\frac{\partial \vec{U}}{\partial \vec{x}} &= \Gamma\frac{\partial \vec{Q}}{\partial \vec{x}} \end{aligned} \tag{20}$$

Various forms of the Preconditioning Matrix Γ have been proposed [4][**?**]. In the present work Eriksson's Preconditioning Matrix [5] has been implemented, based on its successful use in [6].

In primitive variables,

$$\Gamma_p = \begin{bmatrix} 1 & 0 & 0 & 0 & \beta \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \alpha \end{bmatrix} \tag{21}$$

where

$$\begin{aligned} \alpha &= min(1, M_{local}^2, \kappa_p M_\infty^2) \\ \beta &= (1-\alpha)/c^2 \end{aligned} \tag{22}$$

The amount of preconditioning is controlled by α in the sense that it follows the local Mach Number (M_{local}) variations. The parameter κ_p takes values from 3-5. On one hand, its role is to prevent α from approaching

zero in stagnant regions and on the other to assure constant α within the boundary layer [7]. It is noted that zero preconditioning and recovery of the original form of the equations is obtained for $\alpha = 1$.
In order to express the preconditioning matrix in conservative or characteristic variables, the appropriate transformation matrix is applied. For example,

$$\begin{aligned} \Gamma_c &= M^{-1}\Gamma_p M \\ M &= \frac{\partial \vec{U}}{\partial \vec{V}} \end{aligned} \tag{23}$$

When transforming the preconditioned system in characteristic variables, a new set of modified eigenvalues is obtained,

$$\Lambda_\Gamma = \begin{bmatrix} V & 0 & 0 & 0 & 0 \\ 0 & V & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & V'+c' & 0 \\ 0 & 0 & 0 & 0 & V'-c' \end{bmatrix} \tag{24}$$

in which the "modified" speed of sound c' and velocity V' are defined as follows:

$$\begin{aligned} V' &= \frac{1}{2}(1+\alpha)V \\ c' &= \frac{1}{2}\sqrt{\left[(1-\alpha)\,V\right]^2 + 4\alpha c^2} \end{aligned} \tag{25}$$

leading to the following expressions for the eigenvalues:

$$\begin{aligned} \lambda_1 &= \lambda_2 = \lambda_3 = V \\ \lambda_4 &= \frac{1}{2}\left[(1+\alpha)\,V - \sqrt{\left[(1-\alpha)\,V\right]^2 + 4\alpha c^2}\right] \\ \lambda_5 &= \frac{1}{2}\left[(1+\alpha)\,V + \sqrt{\left[(1-\alpha)\,V\right]^2 + 4\alpha c^2}\right] \end{aligned} \tag{26}$$

Since preconditioning changes the eigenvalues, the left and right eigenvectors will also change. The right eigenvectors in primitive variables become:

$$R_{\Gamma p} = \begin{bmatrix} n_x & 0 & n_z & -n_y & -\frac{n_x}{c^2} \\ n_y & -n_z & 0 & n_x & -\frac{n_y}{c^2} \\ n_z & n_y & -n_x & 0 & -\frac{n_z}{c^2} \\ 0 & n_x & n_y & n_z & -\frac{\lambda_1-\lambda_4}{\alpha\rho c^2} \\ 0 & -n_x & -n_y & -n_z & \frac{\lambda_1-\lambda_5}{\alpha\rho c^2} \end{bmatrix} \tag{27}$$

while the left eigenvectors in primitive variables become :

$$L_{\Gamma p} = R_{\Gamma p}^{-1} = \begin{bmatrix} n_x & n_y & n_z & \frac{\alpha\rho}{\lambda_4-\lambda_5} & \frac{\alpha\rho}{\lambda_4-\lambda_5} \\ 0 & -n_z & n_y & \frac{\lambda_1-\lambda_5}{\lambda_4-\lambda_5}n_x & \frac{\lambda_1-\lambda_4}{\lambda_4-\lambda_5}n_x \\ n_z & 0 & -n_x & \frac{\lambda_1-\lambda_5}{\lambda_4-\lambda_5}n_y & \frac{\lambda_1-\lambda_4}{\lambda_4-\lambda_5}n_y \\ -n_y & n_x & 0 & \frac{\lambda_1-\lambda_5}{\lambda_4-\lambda_5}n_z & \frac{\lambda_1-\lambda_4}{\lambda_4-\lambda_5}n_z \\ 0 & 0 & 0 & \frac{\alpha\rho c^2}{\lambda_4-\lambda_5} & \frac{\alpha\rho c^2}{\lambda_4-\lambda_5} \end{bmatrix} \tag{28}$$

2.2 Spatial discretization

In MaPFlow the flow variables are calculated and stored at cell centers. Assuming that the cell volume remains constant in time:

$$\frac{\partial}{\partial t}\int_D \vec{U} dD = D\frac{\partial \vec{U}}{\partial t} \tag{29}$$

where:

$$\vec{U} = \frac{1}{D}\int_D \vec{U}_{exact} dD \tag{30}$$

Thus equation (1) becomes:

$$\frac{\partial \vec{U}}{\partial t} = -\frac{1}{D}[\oint_{\partial D}(\vec{F}_c - \vec{F}_v)dS - \int_D \vec{Q} dD] \tag{31}$$

The surface integral is approximated by the sum of fluxes through the faces of each cell. The usual assumption is that the flux is evenly distributed over every face and its mean is computed at its center. For cell I, (31) takes the form:

$$\frac{d\vec{U}_I}{dt} = -\frac{1}{D_I}\underbrace{[\sum_{m=1}^{N_f}(\vec{F}_c - \vec{F}_v)_m \Delta S_m) - (\vec{Q}D)_I]}_{R_I} = -\frac{1}{D_I}\vec{R}_I \tag{32}$$

where N_f is the number of faces the cell has and ΔS_m is the area of face "m". The terms $(\vec{F}_c)_m, (\vec{F}_v)_m$ represent the convective and viscous fluxes through face m.

2.2.1 Reconstruction of variables

In order to calculate the fluxes appearing in the right hand side of (32), the values of all flow variables at the face centers are needed. This information is absent, since all flow variables are defined at the cell centers. Passing the flow information from the cell centers to the faces is carried out by means of *variable reconstruction*.

Consider two cells *I,J* being in contact over face *f*. Variable reconstruction on *f* can be defined either starting from cell *I* or cell *J*. For compressible solvers it is assumed that across the face the flow experiences a jump defined by the left L and right R states. The L/R specification depends on the normal to f which directs from L to R. These states are used because the interpolation of variables on a specific face is done twice; once for the left side and once for the right side of the face. Afterwards, the flux on the face is computed.

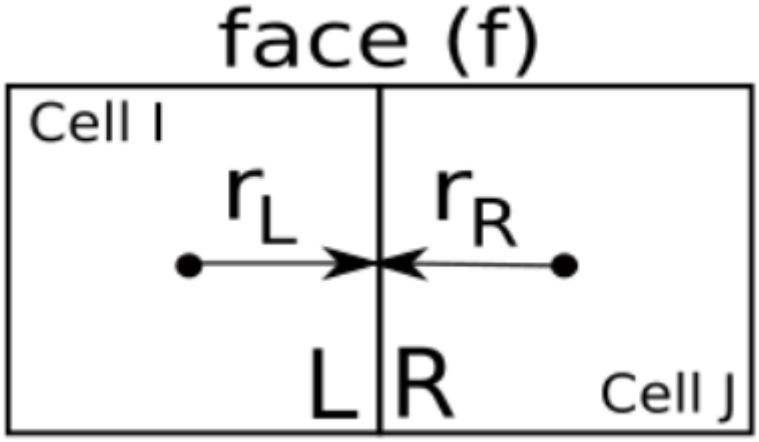


Figure 1: Two cell stencil for the reconstruction of variables on a face (f) using the PLR scheme.

PLR PLR is a 2^{nd} order accurate scheme which utilizes Venkatakrishnan's limiter [8]. The utilized stencil consists of two cells (one cell in both L and R directions), as shown in 1. The assumption that the solution is partially and linearly distributed in the finite volume is made. Left and right states are computed as follows:

$$\vec{W}_L = \vec{W}_I + \Psi_I(\nabla\vec{W}_I \cdot \vec{r}_L) \tag{33}$$

$$\vec{W}_R = \vec{W}_J - \Psi_J(\nabla\vec{W}_J \cdot \vec{r}_R) \tag{34}$$

where $\vec{W}$ is the primitive variables vector:

$$\vec{W} = \begin{pmatrix} \rho \\ u \\ v \\ w \\ E \end{pmatrix} \tag{35}$$

and $\vec{r_L}, \vec{r_R}$ is the corresponding distance of each cell center from the shared cell face between these two cells.

An important factor in the described scheme is the computation of $\nabla(\vec{W_I})$ in (33),(34). The derivative computation utilizes the Green-Gauss approximation, thus:

$$\nabla\vec{W} \approx \frac{1}{\Omega}\int_{\partial\Omega} \vec{W}\vec{n}dS \tag{36}$$

In cell centered schemes, (36) is discretized as follows:

$$\nabla\vec{W}_I \approx \frac{1}{\Omega}\sum_{J=1}^{N_f}\frac{1}{2}(\vec{W}_I + \vec{W}_J)\vec{n}_{IJ}\Delta S_{IJ} \tag{37}$$

Ψ is the limiter function that prevents variables from reaching extreme values in regions with discontinuities. Venkatakrishnan's limiter [8] is a limiter applied to ∇U. The implementation is as follows:

$$\Psi_i = min_j \begin{cases} \frac{1}{\Delta_2}[\frac{\Delta_{1,max}^2+\epsilon^2)\Delta_2+2\Delta_2^2\Delta_{1,max}}{\Delta_{1,max}^2+2\Delta_2^2+\Delta_{1,max}\Delta_2+\epsilon^2}] & \hat{\mathrm{I}}\pm\hat{\mathrm{I}}_{\frac{1}{2}}\ \Delta_2 > 0 \\ \frac{1}{\Delta_2}[\frac{\Delta_{1,min}^2+\epsilon^2)\Delta_2+2\Delta_2^2\Delta_{1,min}}{\Delta_{1,min}^2+2\Delta_2^2+\Delta_{1,min}\Delta_2+\epsilon^2}] & \hat{\mathrm{I}}\pm\hat{\mathrm{I}}_{\frac{1}{2}}\ \Delta_2 < 0 \\ 1 & \hat{\mathrm{I}}\pm\hat{\mathrm{I}}_{\frac{1}{2}}\ \Delta_2 = 0 \end{cases} \tag{38}$$

where

$$\Delta_{1,max} = U_{max} - U_i \tag{39}$$

$$\Delta_{1,min} = U_{min} - U_i \tag{40}$$

The purpose of the parameter ϵ^2 is to define limiter strictness. If it has zero value the limiter is very strict whereas for large values of the parameter the limiter reaches values close to unity and eventually does not have a significant effect on variable values. Parameter ϵ is practically proportional to mesh length scale:

$$\epsilon^2 = (K\Delta h)^3 \tag{41}$$

where K is a free parameter. Small values of K make the limiter strict rendering the PLR first order, while $K = \infty$ leads to an unlimited scheme. Typically the value of $K = 5$ is used.

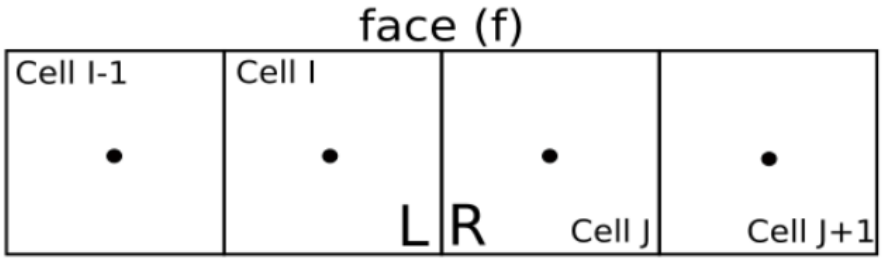


Figure 2: Four cell stencil for the reconstruction of variables on a face (f) using the MUSCL and QUICK schemes. In contrast to the PLR scheme, MUSCL and QUICK schemes do not utilize distance vectors.

MUSCL The MUSCL scheme was first introduced by B. van Leer in [9]. Starting from the scheme of Godunov [10], the piecewise constant approximation is replaced by reconstructed states, which are derived from cell-averaged states from the previous timestep. For every face, slope limited, reconstructed left and right states are obtained and used to calculate fluxes at the cell boundaries. MUSCL takes into account the second order derivative whereas the third order derivative is ignored, thus the scheme is considered 3^{rd} order accurate.

For the equations presented below, the following subscript notation is adopted. (see 2).

- The 1st Left cell is noted with "I"
- The 2nd Left cell is noted with "I-1"
- The 1st Right cell is noted with "J"
- The 2nd Right cell is noted with "J+1"

The left and right states are computed as follows:

$$\vec{W}_L = \vec{W}_I + \frac{1}{4}((1-\kappa)(\vec{W}_I - \vec{W}_{I-1})\phi(r_{I-1}) + (1+\kappa)(\vec{W}_J - \vec{W}_I)\phi(r_I)) \tag{42}$$

$$\vec{W}_R = \vec{W}_J + \frac{1}{4}((1-\kappa)(\vec{W}_{J+1} - \vec{W}_J)\phi(r_J) + (1+\kappa)(\vec{W}_J - \vec{W}_I)\phi(r_{J+1})) \tag{43}$$

where $\kappa = 1/3$.
In contrast to the PLR scheme, r_I is not a distance vector. It is instead calculated as:

$$r_I = \frac{\vec{W}_I - \vec{W}_{I-1}}{\vec{W}_J - \vec{W}_I} \tag{44}$$

If the denominator is lower than 0, $r_I = 1$. Finally, $\phi(r)$ is the limiter function which can either be the superbee limiter [11] or the limiter of van Albada [12].
The superbee limiter reads:

$$\phi(r_I) = max(0, min(2r_I, 1), min(r_I, 2)) \tag{45}$$

whereas that of van Albada reads:

$$\phi(r_I) = \frac{r_I^2 + r_I}{r_I^2 + 1} \tag{46}$$

QUICK QUICK is a higher-order differencing scheme that considers a three-point upstream weighted quadratic interpolation for the cell face values. The scheme was first introduced by B. P. Leonard in 1979 [13]. A quadratic function passing through two bracketing or surrounding nodes and one node in the upstream side is used in order to find the cell face value. In central differencing schemes and second order upwind schemes the first order derivative is included whereas the second order derivative is ignored. QUICK takes the second order derivative into account, but ignores the third order derivative and thus is considered 3rd order accurate.
For the equations presented below, the following subscript notation is adopted (see 2) :

- The 1st Left cell is noted with "I"
- The 2nd Left cell is noted with "I-1"

- The 1st Right cell is noted with "J"
- The 2nd Right cell is noted with "J+1"

The implemented algorithm is based on [14] and is as follows:

Left state:

If $|\vec{W}_J - \vec{W}_{I-1}| \leq 10^{-8}$, then

$$\vec{W}_L = \frac{1}{2}(\vec{W}_I + \vec{W}_J) - \frac{1}{8}(\vec{W}_J - 2\vec{W}_I + \vec{W}_{I-1}) \tag{47}$$

Else if $|\vec{W}_J - 2\vec{W}_I + \vec{W}_{I-1}| \leq 0.3|\vec{W}_J - \vec{W}_{I-1}|$, then

$$\vec{W}_L = \frac{1}{2}(\vec{W}_I + \vec{W}_J) - \frac{1}{8}(\vec{W}_J - 2\vec{W}_I + \vec{W}_{I-1}) \tag{48}$$

Else

$$\vec{W}_{Invd} = \frac{\vec{W}_I - \vec{W}_{I-1}}{\vec{W}_J - \vec{W}_{I-1}} \tag{49}$$

$$\begin{aligned}
&\text{If} \quad \vec{W}_{Invd} \geq 1.5 \text{ or } \vec{W}_{Invd} \leq -1 \text{ or } 0.35 \leq \vec{W}_{Invd} \leq 0.65 \\
&\vec{W}_{face} = 0.75 + 0.75(\vec{W}_{Invd} - 0.5), \\
&\text{else if} \quad -1 < \vec{W}_{Invd} \leq 0, \\
&\vec{W}_{face} = 0.375\vec{W}_{Invd} \\
&\text{else if} \quad 0 < \vec{W}_{Invd} \leq 0.35 \text{ or } 0.65 < \vec{W}_{Invd} \leq 1, \qquad (50) \\
&\vec{W}_{face} = \frac{\sqrt{\vec{W}_{Invd}(1 - \vec{W}_{Invd})^3} - \vec{W}_{Invd}^{\,2}}{1 - 2\vec{W}_{Invd}} \\
&\text{else if} \quad 1 < \vec{W}_{Invd} \leq 1.5, \\
&\vec{W}_{face} = \vec{W}_{Invd}
\end{aligned}$$

$$\vec{W}_L = \vec{W}_{I-1} + (\vec{W}_J - \vec{W}_{I-1})\vec{W}_{face} \tag{51}$$

Right state:

If $|\vec{W}_J - \vec{W}_{J+1}| \leq 10^{-8}$, then

$$\vec{W}_R = \frac{1}{2}(\vec{W}_J + \vec{W}_I) - \frac{1}{8}(\vec{W}_I - 2\vec{W}_J + \vec{W}_{J+1}) \tag{52}$$

Else if $|\vec{W}_I - 2\vec{W}_J + \vec{W}_{J+1}| \leq 0.3|\vec{W}_I - \vec{W}_{J+1}|$, then

$$\vec{W}_R = \frac{1}{2}(\vec{W}_J + \vec{W}_I) - \frac{1}{8}(\vec{W}_I - 2\vec{W}_J + \vec{W}_{J+1}) \tag{53}$$

Else

$$\vec{W}_{Jnvd} = \frac{\vec{W}_J - \vec{W}_{J+1}}{\vec{W}_I - \vec{W}_{J+1}} \tag{54}$$

$$\begin{aligned}
&\text{If} \quad \vec{W}_{Jnvd} \geq 1.5 \text{ or } \vec{W}_{Jnvd} \leq -1 \text{ or } 0.35 \leq \vec{W}_{Jnvd} \leq 0.65, \\
&\vec{W}_{face} = 0.75 + 0.75(\vec{W}_{Jnvd} - 0.5) \\
&\text{else if} \quad -1 < \vec{W}_{Jnvd} \leq 0, \\
&\vec{W}_{face} = 0.375\vec{W}_{Jnvd} \\
&\text{else if} \quad 0 < \vec{W}_{Jnvd} \leq 0.35 \text{ or } 0.65 < \vec{W}_{Jnvd} \leq 1 \\
&\vec{W}_{face} = \frac{\sqrt{\vec{W}_{Jnvd}(1 - \vec{W}_{Jnvd})^3} - \vec{W}_{Jnvd}^{\,2}}{1 - 2\vec{W}_{Jnvd}} \\
&\text{else if} \quad 1 < \vec{W}_{Jnvd} \leq 1.5 \\
&\vec{W}_{face} = \vec{W}_{Jnvd}
\end{aligned} \tag{55}$$

$$\vec{W}_R = \vec{W}_{J+1} + (\vec{W}_I - \vec{W}_{J+1})\vec{W}_{face} \tag{56}$$

2.2.2 Convective Fluxes

The discretization of the convective fluxes can be based on central, flux-vector or flux-difference schemes. Central schemes calculate the convective fluxes across faces as the arithmetic average of the values obtained at the two sides of the face plus an artificial dissipation term added to enhance stability [15]. Flux-vector schemes are based on *upwinding* which respects the direction of propagation of waves [16][17]. Finally, flux-difference schemes calculate convective fluxes at cell faces by solving the Riemann problem for the Left and Right states defined on the face [18].

The present work uses Roe's approximate Riemann solver [18], which is a flux-difference scheme. Roe's scheme consists of constructing the convective flux as a sum of wave contributions:

$$(\vec{F}_c)_{I+\frac{1}{2}} = \frac{1}{2}[\vec{F}_c(\vec{V}_R) + \vec{F}_c(\vec{V}_L) - |A_{Roe}|_{I+\frac{1}{2}}(\vec{V}_R - \vec{V}_L)] \tag{57}$$

where the Left and Right states $(\vec{V}_L, \vec{V}_R)$ are calculated using one of the available numerical schemes. The Roe matrix A_{Roe} has the same form as the convective flux Jacobian but instead of formally averaged values, the following Roe-averaged variables are used:

$$
\begin{aligned}
\tilde{\rho} &= \sqrt{\rho_L \rho_R} \\
\tilde{u} &= \frac{u_L\sqrt{\rho_L} + u_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{v} &= \frac{v_L\sqrt{\rho_L} + v_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{w} &= \frac{w_L\sqrt{\rho_L} + w_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{H} &= \frac{H_L\sqrt{\rho_L} + H_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{c} &= \sqrt{(\gamma - 1)(\tilde{H} - \tilde{q}^2/2)} \\
\tilde{q} &= \tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2
\end{aligned}
$$

In (57), $|A_{Roe}|$ is constructed using the absolute values of the eigenvalues and the the right eigenvector matrix R:

$$
|A_{Roe}| = R^{-1}|\Lambda|R \tag{58}
$$

In case the preconditioned system of equation is used, the eigenvalues and the eigenvectors of the preconditioned system must be used $(\Lambda_\Gamma, R_\Gamma)$. Thus, according to [19], $|A_{Roe}|$ is changed to:

$$
\begin{aligned}
|A_{\Gamma Roe}| &= |\Gamma^{-1}\Gamma A_{Roe}| \\
&\simeq \Gamma^{-1}|\Gamma A_{Roe}| \\
&\simeq \Gamma^{-1}R_\Gamma^{-1}|\Lambda_\Gamma|R_\Gamma
\end{aligned} \tag{59}
$$

2.2.3 Viscous Fluxes

For the calculation of Viscous Fluxes, variable values and space derivatives are needed. For the face in between cells I and J, variable values are obtained from simple averaging:

$$
\vec{V}_{IJ} = \frac{1}{2}\left(\vec{V}_I + \vec{V}_J\right) \tag{60}
$$

while for the gradients, the Green-Gauss formula is applied using the face averaged values $\vec{V}_{IJ}$ as defined in (60) but supplemented with a directional derivative[20]:

$$
\nabla\vec{V}_{IJ} = \overline{\nabla\vec{V}_{IJ}} - \left[\overline{\nabla\vec{V}_{IJ}} \cdot \vec{t_{IJ}} - \left(\frac{\partial\vec{V}}{\partial l}\right)_{IJ}\right] \cdot \vec{t_{IJ}} \tag{61}
$$

where

$$
\overline{\nabla\vec{V}_{IJ}} = \frac{1}{2}\left(\nabla\vec{V}_I + \nabla\vec{V}_J\right) \tag{62}
$$

is the mean gradient,

$$\left(\frac{\partial \vec{V}}{\partial l}\right)_{IJ} \approx \frac{\vec{V}_J - \vec{V}_I}{l_{IJ}} \tag{63}$$

and l_{IJ} is the distance between cell centers I and J and $\vec{t_{IJ}}$ is the unit vector pointing from cell center I to cell center J.

2.3 Temporal discretization

For the temporal discretization the method of lines is used. This means that temporal and spatial discretization are done separately leading for every control volume to the following equation:

$$\frac{d\left(D_I\vec{U}_I\right)}{dt} = -R_I \tag{64}$$

In comparison to (31) the form of equation (64) is more general in the sense that the control volume can vary with time.

Temporal discretization can be either explicit or implicit. Explicit methods use the $\vec{U}^n$ known solution and march in time using the corresponding residual $\vec{R}^n$ to obtain solution at $(t + \Delta t)$. On the other hand the implicit schemes use $R(\vec{U}^{n+1}) = \vec{R}^{n+1}$ to obtain the new solution and are favored because they allow larger time-steps. Since $\vec{R}^{n+1}$ is unknown, the following linear approximation is used:

$$\vec{R}^{n+1} \approx \vec{R}^n + \left(\frac{\partial\vec{R}}{\partial\vec{U}}\right)_n \cdot \Delta\vec{U}^n, \quad \Delta\vec{U}^n = \vec{U}^{n+1} - \vec{U}^n \tag{65}$$

In MaPFlow a finite difference scheme is used for the time derivative [21]:

$$\frac{1}{\Delta t}\left[\phi_{n+1}\left(D\vec{U}\right)^{n+1} + \phi_n\left(D\vec{U}\right)^{n} + \phi_{n-1}\left(D\vec{U}\right)^{n-1} + \phi_{n-2}\left(D\vec{U}\right)^{n-2} + \dots\right] = -R^{n+1} \tag{66}$$

Depending on the choice of ϕ_n the corresponding backwards difference formulae (BDF) of the temporal scheme is defined. $BDF2OPT$, refers to a class of optimized, second-order, backward difference methods with an error constant half as large as the conventional 2^{nd} order scheme [22].

Table 1: Backwards Difference Schemes

order	ϕ_{n+1}	ϕ_n	ϕ_{n-1}	ϕ_{n-1}
1^{st}	1	-1	0	0
2^{nd}	3/2	-2	1/2	0
3^{rd}	11/6	-3	3/2	$-1/3$
$BDF2OPT$	$3/2 - \phi_{n-2}$	$-2 + 3\phi_{n-2}$	$1/2 - 3\phi_{n-2}$	$-0.58/3$

2.3.1 Steady State Computations

Even when steady state simulations are considered a pseudo-unsteady technique is followed. For steady state simulations 1^{st} order scheme is chosen to march the solution in pseudo-time until convergence is reached. At 1^{st} order, after linearizing R^{n+1}, (66) becomes:

$$\frac{\left(D_I\Delta\vec{U}_I^n\right)}{\Delta t_I} = \vec{R}_I^n + \left(\frac{\partial\vec{R}}{\partial\vec{U}}\right)_I \Delta\vec{U}_I^n \tag{67}$$

By rearranging the terms the final system of discrete equations is obtained in which the system matrix defines the *implicit operator* of the scheme:

$$\underbrace{\left[\frac{(D)_I}{\Delta t_I} + \left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)_I\right]}_{\text{Implicit Operator}} \Delta \vec{U}_I^n = -\vec{R}_I^n \tag{68}$$

Local Time Stepping In order to facilitate convergence, the Local Time Step technique is used [23]. The time step for steady state calculation can be defined using the spectral radii of each cell. For every cell, a different time step is defined by:

$$\Delta t = CFL \frac{D_I}{\left(\hat{\Lambda}_c + C\hat{\Lambda}_v\right)_I} \tag{69}$$

where $\hat{\Lambda}_c$, $\hat{\Lambda}_v$ is the sum of convective and viscous eigenvalues over all cell faces. The convective spectral radii is defined by:

$$(\hat{\Lambda}_c)_I = \sum_{J=1}^{N_f} (|\vec{u}_{IJ} \cdot \vec{n}_{IJ}| + c_i j) \Delta S_{IJ} \tag{70}$$

and the viscous spectral radii by:

$$(\hat{\Lambda}_v)_I = \frac{1}{D_I} \sum_{J=1}^{N_f} [max(\frac{3}{3\rho_{IJ}}, \frac{\gamma_{IJ}}{\rho_{IJ}})(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T})_{IJ} (\Delta S_{IJ})^2] \tag{71}$$

2.3.2 Time True Computations

When making Time True computations, temporal discretization is crucial because any remaining numerical error will propagate in the flow as a disturbance. In order to minimize temporal errors higher order schemes should be used in conjunction with the Dual Time-Step technique [24].

Dual Time-Stepping The Dual Step approach adds an extra time-like derivative in the transport equation that refers to a different "time variable" τ, called "pseudo-time". The conservative variables in the pseudo-time problem are denoted by $U^\star$, because they don't satisfy the original unsteady problem until convergence is reached

Using this approach the unsteady problem is transformed into a steady one. In every true time-step the following problem is solved in the pseudo-time (τ) :

$$\frac{\partial (D^{n+1} \vec{U}^\star)}{\partial \tau} + \vec{R}^\star = 0 \tag{72}$$

Setting,

$$\vec{R}^\star = \frac{\partial (D\vec{U}^\star)}{\partial t} + R(\vec{U}^\star) \tag{73}$$

the following final form is obtained:

$$\frac{\partial(D^{n+1}\vec{U}^{\star})}{\partial\tau}+\frac{\partial(D\vec{U}^{\star})}{\partial t}=-R(\vec{U}^{\star}) \tag{74}$$

When (72) converges, $R^{\star}=0,\vec{U}^{\star}=\vec{U}$ which satisfies the original unsteady problem.

The discretized form of (74) can be written as:

$$\frac{D^{n+1}\Delta U^{\star k}}{\Delta\tau}+\frac{1}{\Delta t}\left[\phi_{n+1}\left(D\vec{U}^{\star}\right)^{n+1}+\phi_n\left(D\vec{U}\right)^{n}+\phi_{n-1}\left(D\vec{U}\right)^{n-1}\right]=-R^{k+1} \tag{75}$$

or

$$\frac{D^{n+1}\Delta U^{\star k}}{\Delta\tau}=-R^{\star k+1} \tag{76}$$

with

$$\Delta U^{\star k}=U^{\star k+1}-U^{\star k} \tag{77}$$

and

$$R^{\star k+1}=R^{k+1}-\frac{1}{\Delta t}\left[\phi_{n+1}\left(D\vec{U}^{\star}\right)^{n+1}+\phi_n\left(D\vec{U}\right)^{n}+\phi_{n-1}\left(D\vec{U}\right)^{n-1}\right] \tag{78}$$

where k denotes the steady state problem sub-iteration.

In order to apply an Implicit Scheme in the Dual Time-Step procedure we must linearize the unsteady residual $R^{\star k+1}$:

$$\vec{R}^{\star k+1}\approx\vec{R}^{\star k}+\left(\frac{\partial\vec{R}}{\partial\vec{U}^{\star}}\right)_k\cdot\Delta\vec{U}^{\star k} \tag{79}$$

or

$$\begin{aligned}\vec{R}^{\star k+1}\approx\vec{R}^{k}&-\frac{1}{\Delta t}\left[\phi_{n+1}\left(D\vec{U}^{\star}\right)^{n+1}+\phi_n\left(D\vec{U}\right)^{n}+\phi_{n-1}\left(D\vec{U}\right)^{n-1}\right]\\&+\frac{\partial\vec{R}}{\partial\vec{U}^{\star}}\Delta\vec{U}^{\star}-\phi_{n+1}\frac{D^{n+1}}{\Delta t}\Delta\vec{U}^{\star}\end{aligned} \tag{80}$$

The correction $\Delta U^{\star}$ refers to the steady problem defined in pseudo-time. Thus, when the steady problem converges, $\Delta U^{\star}=0$. However, this does not correspond to $U^{n+1}-U^{n}=0$ but to $U^{k+1,n+1}-U^{k,n+1}=0$. The ϕ coefficients change according to the desired time discretization scheme (see Table 1).

Substituting in (76) the final form is obtained:

$$\left[\frac{D^{n+1}}{\Delta\tau}+\phi_{n+1}\frac{D^{n+1}}{\Delta t}+\frac{\partial\vec{R}}{\partial\vec{U}^{\star}}\right]\Delta U^{\star k}=-R^{k}-Q^{k}_{dual} \tag{81}$$

where the dual step unsteady source-like terms Q^{k}_{dual} are given by:

$$Q^k_{dual} = \frac{1}{\Delta t}\left[\phi_{n+1}\left(D\vec{U}^{\star}\right)^{n+1} + \phi_n\left(D\vec{U}\right)^{n} + \phi_{n-1}\left(D\vec{U}\right)^{n-1}\right] \tag{82}$$

It is noted here that the pseudo time step $\Delta\tau$ is defined as in the Steady state computations using local timestepping (69).

2.4 Boundary conditions

In external aerodynamics the following boundary conditions are needed:

- Far-field Boundaries
- Solid wall Boundaries
- Symmetry Boundaries
- Periodic Boundaries

Before analyzing each one of the boundary condition types, it is important to discuss the concept of dummy cells. Dummy cells are additional virtual cells that extend the computational domain. Their purpose is to provide assistance in calculating the flow variables at the computational domain boundaries. Far-field and solid wall conditions are defined exactly at the boundary face while the other two are applied at the center of the dummy cell.

2.4.1 Far-field Boundaries

In the far-field, it is important to comply with the hyperbolic character of the problem as expressed when formulated in characteristic variables. The information provided is related to the sign of the eigenvalues of the flow state at the far-field boundary and the associated Riemann invariants. Both must be respected and therefore the far-field boundary conditions must be accordingly defined. The approach followed is based on the characteristics of the 1D Euler equations along the normal direction to the boundary [25], $u = \vec{V} \cdot \vec{n}$. The sign of each of the three eigenvalue u, $u + c$, $u - c$, defines the direction of propagation while along the corresponding characteristic the associated Riemann invariants R, $R+$, $R-$ (see 3).

$$\begin{aligned} R^{\pm} &= u \pm \frac{2c}{\gamma - 1} \\ R &= s \end{aligned} \tag{83}$$

remain constant.

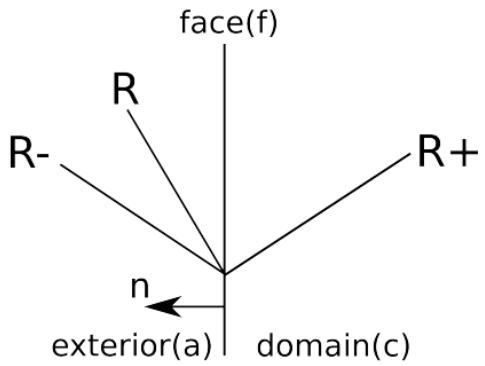


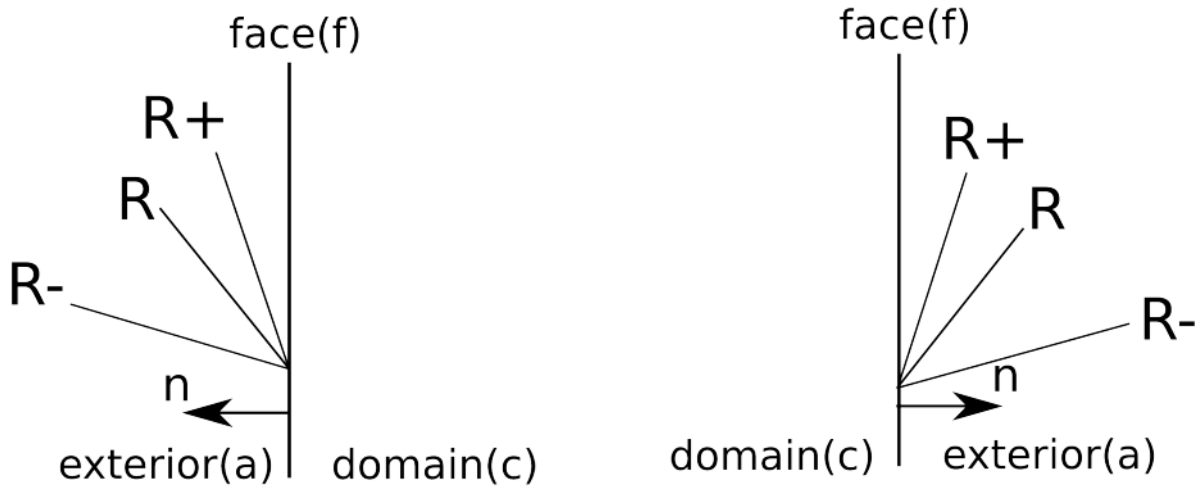
Figure 3: The case of a subsonic inlet face. Note that on an inlet face and the normal defined to point outwards, the normal to the boundary velocity component $u = \vec{V} \cdot \vec{n} < 0$. This means that in reality the flow information associated to $R, R-$ is provided by the state defined in (a).

Thus on an inflow face (and similarly for an outflow face), using the invariants:

$$\begin{array}{lll} \text{from } R^+ & u_f + \dfrac{2c_f}{\gamma-1} = u_c + \dfrac{2c_c}{\gamma-1} \\ \text{from } R^- & u_f - \dfrac{2c_f}{\gamma-1} = u_a - \dfrac{2c_a}{\gamma-1} \\ \text{isentropic assumption} & s_f \quad = s_a \end{array} \tag{84}$$

Although the flow variables at boundary faces can be obtained as linear combinations of these invariants, in the present formulation, the characteristic equations are used instead:

$$\begin{array}{llll} d\rho - \dfrac{1}{c^2}dp = 0 & \text{along} & \lambda_1 = u \\ du + \dfrac{1}{\rho c}dp = 0 & \text{along} & \lambda_2 = u + c \\ du - \dfrac{1}{\rho c}dp = 0 & \text{along} & \lambda_3 = u - c \end{array} \tag{85}$$



(a) Supersonic Inlet (b) Supersonic Outlet

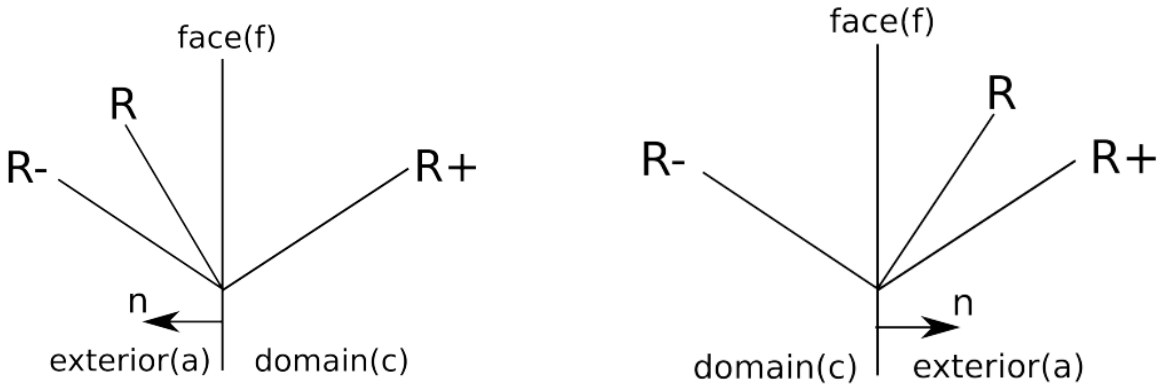
Figure 4: Riemann Invariants on a far-field supersonic boundary

Supersonic Inlet-Outlet In the supersonic case all eigenvalues are positive since $V > c$. This means that in the case of inflow, all flow information is propagating from outside into the domain (4) and therefore all flow variables must be given as input:

$$\vec{V}_{inlet} = \vec{V}_{\infty} \tag{86}$$

On the contrary in the case of outflow all flow information propagates from inside of the domain:

$$\vec{V}_{outlet} = \vec{V}_{domain} \tag{87}$$



(a) Subsonic Inlet (b) Subsonic Outlet

Figure 5: Riemann Invariants on a far-field subsonic boundary

Subsonic inlet-outlet For subsonic inflow and along the normal to the boundary, the two characteristics propagate information from outside of the domain while the third propagates flow information from inside of the domain (5). The situation reverses in case of subsonic outflow where two characteristics propagate information from inside the domain while the third propagates information from outside into the flow domain.

Note that the three Riemann invariants, associated with the eigenvalues, are defined with respect to the normal direction; hence for the subsonic inlet with the normal vector pointing outwards $u - c < 0$, $u < 0$, $u + c > 0$, while for the subsonic outlet $u - c < 0$, $u > 0$, $u + c > 0$.
Based on (85), with the normal pointing outwards of the inlet face:

$$\begin{aligned} \frac{p_f}{\rho_f^\gamma} &= \frac{p_a}{\rho_a^\gamma} \qquad \text{along} \qquad \lambda_1 = u, (R) \\ p_f - p_c + \frac{1}{\rho_a c_a}(u_f - u_c) &= 0 \qquad \text{along} \qquad \lambda_2 = u + c, (R+) \\ p_f - p_a - \frac{1}{\rho_a c_a}(u_f - u_a) &= 0 \qquad \text{along} \qquad \lambda_3 = u - c, (R-) \end{aligned} \tag{88}$$

By combining R^+ and R^- pressure and velocity at the boundary are determined. Density can be retrieved from the isentropic relation. As reference state at the inlet, that at the exterior of the domain is used. Similarly, under the assumption that the normal direction is pointing outside of the domain, at the outlet boundary,

$$\begin{aligned} \frac{p_f}{\rho_f^\gamma} &= \frac{p_c}{\rho_c^\gamma} \qquad \text{along} \qquad \lambda_1 = u, (R) \\ p_f - p_c + \frac{1}{\rho_c c_c}(u_f - u_c) &= 0 \qquad \text{along} \qquad \lambda_2 = u + c, (R+) \\ p_f - p_a - \frac{1}{\rho_c c_c}(u_f - u_a) &= 0 \qquad \text{along} \qquad \lambda_3 = u - c, (R-) \end{aligned} \tag{89}$$

in which the reference state is defined from inside of the computational domain.

Preconditioned characteristic equations As already discussed in Low Mach Preconditioning (Section 2.1.3) , the eigenvalues and eigenvectors of the system change (26). Hence, it is expected that the characteristic equations change too. When Eriksson's Preconditioning Matrix is used, like in the present case, the 1-D characteristic equations take the form [26]:

$$\begin{aligned} d\rho - \frac{1}{c^2} dp &= 0 \qquad \text{along} \qquad \lambda_1 = u \\ du + \frac{c' - \alpha_m u}{\rho \alpha c^2} dp &= 0 \qquad \text{along} \qquad \lambda_2 = u' + c' \\ du - \frac{c' + \alpha_m u}{\rho \alpha c^2} dp &= 0 \qquad \text{along} \qquad \lambda_3 = u' - c' \end{aligned} \tag{90}$$

with,

$$\begin{aligned} \alpha_m &= \frac{(1 - \alpha)}{2} \\ u' &= \frac{1}{2}(1 + \alpha) u \\ c' &= \frac{1}{2}\sqrt{[(1 - \alpha) u]^2 + 4\alpha c^2} \end{aligned} \tag{91}$$

As in the un-preconditioned case, the characteristic equations are combined to obtain $\rho, \vec{u}, p$ at the boundary face,

$$\begin{aligned}
\rho_f &= \rho_o \left(\frac{p_b}{p_o}\right)^{\gamma} \\
V_{nf} &= V_{nc} + \left(\frac{1}{\rho c'}\right)\left(\frac{p_c - p_a}{2}\right) - \left(1 - \frac{\alpha_m V_{no}}{c'}\right)\left(\frac{V_{nc} - V_{na}}{2}\right) \\
p_f &= p_a + \left(1 - \frac{\alpha_m V_{no}}{c'}\right)\left(\frac{p_c - p_a}{2}\right) + \left(\frac{\rho \alpha c^2}{c'}\right)\left(\frac{V_{nc} - V_{na}}{2}\right) \\
u_f &= u_o + (V_{nf} - V_{no}) \cdot n_x \\
v_f &= v_o + (V_{nf} - V_{no}) \cdot n_y \\
w_f &= w_o + (V_{nf} - V_{no}) \cdot n_z
\end{aligned} \tag{92}$$

where

$$V_{n(\cdot)} = u_{(\cdot)} \cdot n_x + v_{(\cdot)} \cdot n_y + w_{(\cdot)} \cdot n_z \tag{93}$$

The reference state, here denoted by subscript o, is decided by the isentropic wave, λ_1. At a subsonic inlet face it corresponds to the outer flow data whereas at a subsonic outlet face the values from inside of the computational domain are used.

2.4.2 Wall Boundary Conditions

Inviscid Wall When the fluid is assumed inviscid on solid boundaries,

$$(\vec{u} - \vec{u}_g) \cdot \vec{n} = 0 \tag{94}$$

where $\vec{u}_g$ denotes the mesh velocity. Density and pressure are set equal to their values at the cell center next to the wall,

$$p_w = p_I,\ \rho_w = \rho_I \tag{95}$$

Viscous Wall In the general case, the fluid is viscous and the no slip wall condition is applied,

$$\vec{u} = \vec{u}_g \tag{96}$$

Density and pressure are treated as in the inviscid case. Regardless of the assumptions made for the fluid, the convective fluxes take the form,

$$\vec{F}_{cwall} = \begin{pmatrix} 0 \\ n_x p_w \\ n_y p_w \\ n_z p_w \\ p_w V_g \end{pmatrix} \tag{97}$$

where

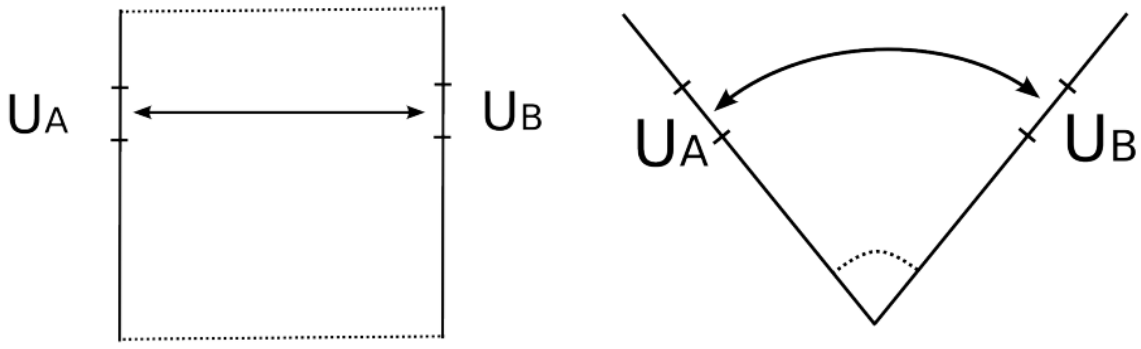
$$V_g = \vec{u}_g \cdot \vec{n} \tag{98}$$

2.4.3 Symmetry and Periodic Boundary Conditions

Symmetry When the flow is symmetric with respect to a plane, the simulation can only concern half of the flow domain. Over the symmetry plane, the condition imposed resembles to that applied to an Inviscid Wall. In both cases there is no flux across the boundary. However, in symmetry conditions gradients normal to the boundary must also vanish [20]. So,

$$\begin{aligned} \vec{u} \cdot \vec{n} &= 0, \qquad \text{zero flux} \\ \nabla \vec{U} \cdot \vec{n} &= 0, \qquad \text{zero normal gradients} \end{aligned} \tag{99}$$

Periodic Flows If the flow is periodic, the simulation is conducted on one single period. In such cases, parts of the outer boundary of the mesh will correspond to the periodic surfaces on which explicit boundary data are not available. Closure of the problem is done by the Periodic boundary conditions that basically transfer the flow variables from one periodic surface to the next one. Periodicity can be either translational (e.g. as in the case of a wing equipped with an array of vortex generators) or rotational (e.g. as in the case of multi-bladed rotors) depending on the motion required for collapsing a periodic surface to its associate. The implementation is based on ghost cells [20].



(a) Translational Periodicity

(b) Rotational Periodicity

Figure 6: Periodic Boundary Conditions

Let A, B denote the two associated periodic surfaces (6). For every cell in contact with A, it's corresponding ghost cell is associated to a cell in contact with B. Scalar quantities are transferred from B to the ghost cells of A as they are, while vector (and tensor) variables follow the motion required for collapsing A on B. The motion is defined as a transformation matrix R_A, and so,

$$\begin{aligned} U_A &= U_B \\ \vec{U}_A &= R_A \vec{U}_B \end{aligned} \tag{100}$$

In translational periodicity $R_A = I$ while in rotational periodicity R_A is the corresponding rotation matrix.

2.5 Solution of the System of Equations

The final form of the discrete equations corresponds to a linear system

$$AX = B \tag{101}$$

of large dimension. The above system can be either solved directly or iteratively. Direct solvers are accurate but have demanding memory requirements and are not easily parallelised. On the contrary, iterative solvers might need many iterations to converge but are suitable for parallel coding and have limited memory requirements. Therefore in the present work, an iterative solver was chosen. Following the work of [27], the Jacobi iterative solver was implemented.

Noting that all but one terms in the discrete form of the equations for cell I,

$$\left[\frac{(D)_I}{\Delta t_I} + \left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)_I\right] \Delta \vec{U}_I^n = -\vec{R}_I^n \tag{102}$$

refer to the cell in consideration, the following splitting

$$D_I \Delta \vec{U}_I^n + O_I \sum \Delta \vec{U}_J^n = -\vec{R}_I^n \tag{103}$$

is introduced. In (103), the first term is block diagonal and is linked to cell I, while the second contains the off diagonal contributions in (102) that are linked to $\left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)_I$. This term involves the cells that surround I.

Jacobi iterative solver A solution to (103) can be achieved using the Jacobi method:

$$D_I \Delta \vec{U}_I^{n,k+1} = -\vec{R}_I^n - O_I \sum \Delta \vec{U}_J^{n,k} \tag{104}$$

where k is the Jacobi iterations index.

Gauss-Seidel iterative solver As an alternative, the Gauss-Seidel method can be used. It is similar to Jacobi solver, except the fact that the off diagonal terms are calculated using the current update for $\vec{U}$,

$$D_I \Delta \vec{U}_I^{n,k+1} = -\vec{R}_I^n - O_I \sum \Delta \vec{U}_L^{n,k+1} - O_I \sum \Delta \vec{U}_J^{n,k} \tag{105}$$

where U_L concerns the cell values that have been updated in $k+1$ iteration.

The performance of the Gauss-Seidel method strongly depends on the type of the matrix A in (101). If A is banded, the matrix can be split in an Upper and Lower part and thus Gauss-Seidel becomes:

$$D_I \Delta \vec{U}_I^{n,k+1} = -\vec{R}_I^n - O_I \sum \Delta \vec{U}_L^{n,k+1} - O_I \sum \Delta \vec{U}_R^{n,k} \tag{106}$$

However if the sparsity of A is substantial, the Gauss-Seidel solver has the same convergence properties as the Jacobi method [28].

In case the mesh is structured, the matrix is indeed banded and the Gauss-Seidel solver will behave well. On the contrary, if an unstructured mesh is used, because the band width of the matrix depends on the cell

numbering, good performance is directly linked to proper renumbering. In this respect the Reverse Cuthill-Mckee (RCM) reordering scheme [28] substantially reduces the band-width and therefore the Gauss-Seidel methods outperforms the Jacobi solver.

It is important to note here that in a parallel environment even if Gauss-Seidel iterative solver is used the update ΔU_J^n must remain in the k iteration if U_J is a multi-block ghost cell. The reason behind this constraint is to ensure that the solution will be continuous across the blocks at all times.

3 Immersed Boundary Method

3.1 Introduction - Description of the IBM

The idea of the Immersed Boundary Method was introduced by Peskin [29] who used momentum forcing to study the blood flow in the heart valves. Another approach was introduced by Goldstein et al. [30]. Feedback forcing was employed in order to represent the solid body which resulted in spurious oscillations and restricted the computational time step. Mohd-Yusof [31] suggested another approach to evaluating the momentum forcing in a spectral method which does not require a smaller computational time step. This same method was applied by Fadlun et al. [32] to a finite-difference method on a staggered grid and showed that the discrete time forcing suggested by Mohd-Yusof [31] is more efficient than feedback forcing for three dimensional flow problems [31, 1]. The difference between the approaches of Mohd-Yusof [31] and Fadlun et al. [32] on where momentum forcing is applied [1]. In the study of Mohd-Yusof [31] momentum forcing is applied on the body surface or inside the body. On the other hand, in Fadlun et al. [32], the velocity at the first grid point external to body is obtained by linearly interpolating the velocity at the second grid point (calculated through the Navier-Stokes equations) and the velocity at the body surface, which corresponds to applying momentum forcing inside the flow field [1]. Finally, Ye et al. [33] proposed a different approach using a Cartesian grid and without using momentum forcing. The cells cut by the boundary are reshaped into body fitted trapezoidal shapes by discarding the solid part of the cell and adding the neighboring cells. In order to accurately discretize the governing equations at the cell containing the immersed boundary, they presented a new interpolation procedure that preserves the second-order spatial accuracy [33, 1].
The Immersed Boundary Methods can be categorized into three branches:

- Cut-cell methods
- Penalization methods
- Momentum forcing methods

Cut-cell methods focus on the discretization of the equations in the cells cut by the immersed boundary (see [33] for instance). In order to represent the immersed boundary, an algebraic distance to the solid boundary is used. The staggered arrangement of the unknows for both the velocity field and the pressure is adapted to the geometry of the cut cells. There is no diffusion of the fluid-solid interface as the boundary conditions are directly imposed. These methods are highly efficient from a computational point of view as they are based on the MAC solver on cartesian grids which has been extensively and succesfully used in numerical simulations of turbulent flows. The only drawback of these methods is the local construction of the discreet operator that depends on the position of the immersed boundary in the cut-cells. On the other hand, in a penalization method the presence of a solid obstacle in the computational domain is modeled by adding a penalty term, depending on a penalization parameter, in the Navier-Stokes equations. The penalization method does not depend on the choice of the discretization schemes used to approximate the

equations and is very easy to implement. In order to bound the difference between the penalization solution and the solution of the Navier-Stokes equations Sobolev norms are used in terms of this parameter. In this approach the immersed interface is not directly represented in the flow simulation and depends on the mesh properties. Finally these methods use both Eulerian and Lagrangian variables which are related via interaction equations. Discrete versions of Dirac functions are used to describe the fluid structure interaction forcing and to link the Eulerian and Lagrangian variables. The main idea is to compute the force applied by the obstacle on the fluid so that the velocity field satisfies the boundary conditions on the immersed boundary (see [34]). These methods are efficient and easy to implement, see [32]. The main and only drawback of these methods is that their coupling with projection schemes introduces difficulties in imposing at the same time level the continuity equation and the boundary conditions on the immersed interface and can alter the incompressibility of the flow in the vicinity of the immersed boundary [35].

3.2 Implementation of the Immersed Boundary Method

The main objective is to simulate the effects of a viscous laminar flow around a 2D cylinder and an airfoil using only Immersed Boundary grids with a cut-cell approach. Towards this goal a cylindrical grid was generated without excluding the points of the computational domain which are situated inside the solid boundary. Additionally an O-type grid with a Cartesian refined region was created with different discretizations inside the cartesian domain. In both cases the results obtained with non-body fitted grids were compared to those obtained with the use of a body fitted grid for two bodies, an airfoil and a circular cylinder. In all cases low Reynolds numbers were used. Interpolation schemes were selected to simulate the effect of the solid boundary as well as a simple differentiation of the cell centers of the computational domain in order to make a clear distinction between the points which are entirely inside the flow, those who are cut by the boundary and those who are entirely (all nodes and cell centers) inside the solid body.

1. The body is defined as a set of points with a given orientation (clockwise or anti-clockwise).
2. Mesh cells are defined as internal,flow or immersed cells.
3. An interpolation scheme is selected (M_4 or the Inverse Distance Weighting scheme) and applied to the immersed cells. In both cases, along with the interpolation scheme, an appropriate kernel is selected for each immersed cell or its mirror point as will be presented in the following pages.
4. Depending on the cell type (internal,flow,immersed) the cells are treated differently with flow cells being solved using MaPFlow (2.5), the immersed cells are solved through the use of the selected interpolation scheme and internal cells take no part in the solution.

The whole process starts with the identification of the grid points. Grid points refer to cell centers and not to nodes or face centers. The body geometry is represented by a set of x, y coordinates and has to be sorted so that the numbering of the body nodes follows a clockwise or counter-clockwise orientation. It is

important to make a distinction between the upper and the lower part of a body. Given the maximum and the minimum of the x coordinates of the body nodes the chord line is calculated. The position of each body node compared to the chord line is obtained and a node is treated differently depending on whether it is on the upper or the lower side of the body.
Given $(x_{max}^{body}, y_{max}^{body})$ and $(x_{min}^{body}, y_{min}^{body})$:

$$\alpha = \frac{y_{max}^{body} - y_{min}^{body}}{x_{max}^{body} - x_{min}^{body}} \tag{107}$$

$$\beta = y_{max}^{body} - \alpha \cdot x_{max}^{body} \tag{108}$$

After this is calculated, for each body node a slope and an intercept is calculated for each line that connects every $i, i+1$ body node. For every grid point, situated sufficiently close to the solid body an algorithm checks between which body nodes this point is and whether it is closer to the upper or to the lower side of the body. When this is established and the $i, i+1$ nodes neighboring the grid point are known then the midpoint (x_m, y_m) of this interval is calculated.

$$\vec{V_1} = \begin{pmatrix} x_i^{body} - x_m \\ y_i^{body} - y_m \end{pmatrix} \tag{109}$$

$$\vec{V_2} = \begin{pmatrix} x_{grid} - x_m \\ y_{grid} - y_m \end{pmatrix} \tag{110}$$

Finally, a grid point is considered as inside the solid boundary if two conditions are met. The first condition being that it is lower (higher) than the line connecting the $i, i+1$ nodes if it is on the upper (lower) side of the body. The second conditions is that the angle θ between $\vec{V_1}$ and $\vec{V_2}$ is positive.

$$\theta = \arccos \frac{\vec{V_1} \cdot \vec{V_2}}{|\vec{V_1}||\vec{V_2}|} \tag{111}$$

In the same way, the aforementioned distinction (see Fig. 7) between the grid points is made. In that case, the above algorithm is not implemented on the cell centers only but also on the cell nodes. There are three possibilities in total:

- **Wall cell** : A cell that has all its nodes and its center *inside* the body.
- **Immersed cell** : The center of the cell is *inside* the body and at least one node *outside* of it.
- **Flow cell** : A cell that has its center *outside* the body. In that case the position of the cell center is of little importance.

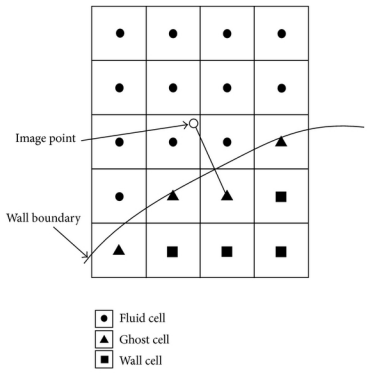


Figure 7: Discretization of points depending on their position with regards to the boundary.

This way the computational grid is divided into three categories and the MaPFlow (Chapter 2) solver treats the cells differently.

The values at the cell centers were obtained depending on their location. The flow characteristics in the flow cells were calculated with the method presented in Chapter 2. The cells inside the solid body were given zero values as they take no part in the calculation whatsoever. The immersed cells, those who are cut by the boundary were given values depending on their neighbors. Neighboring cells are defined as two cells that share the same face. Given the geometry of the computational grid, a cell can have, at most, four neighbors. After the neighbors are defined, the value at the immersed cell is calculated using an interpolation scheme. Two schemes were chosen and tested. The first one is a simple Inverse Distance Weighting scheme and the second one is the M_4 interpolation scheme.

In this simple IDW weighting function, see [36], x denotes the interpolated point, x_i denotes the interpolating point with known values, d is the distance between the known point x_i and the unknown point x, N is the total number of known points used in the interpolation and p is a positive real number, called the power parameter, which in our case is equal to 1.

$$u(x) = \begin{cases} \dfrac{\sum_{i=1}^{N} w_i(x) u_i}{\sum_{i=1}^{N} w_i(x)} & \text{if } d(x, x_i) \neq 0 \text{ for all } i, \quad (112a) \\ u_i & \text{if } d(x, x_i) = 0 \text{ for some } i. \quad (112b) \end{cases}$$

where $w_i(x)$ is either a simple Inverse Distance Weighting function given as,

$$w_i(x) = \frac{1}{d(x,x_i)^p}$$

As in Palha et al. [37] the M_4 scheme is, as follows,

$$M_4(\rho) = \begin{cases} 1 - \dfrac{5}{2}\rho^2 + \dfrac{3}{2}\rho^3 & \text{if } |\rho| < 1, \quad (113a) \\ \dfrac{1}{2}(2 - |\rho|)^2(1 - |\rho|) & \text{if } 1 \leq |\rho| < 2, \quad (113b) \\ 0 & \text{if } |\rho| \geq 2 \quad (113c) \end{cases}$$

where ρ is defined as:

$$\rho = \begin{cases} \frac{x-x_i}{dx} \\ \frac{y-y_i}{dy}, \end{cases} \tag{114}$$

where dx, dy refer to the cell,and thus,

$$M_4(\rho) = M_4^x(\rho) \cdot M_4^y(\rho) \tag{115}$$

The implementation of the M_4 interpolation scheme requires the selection of an appropriate mirror point with respect to the solid boundary. More specifically an immersed cell is projected into the flow, interpolation cells depending on the distance are selected for the mirror, its values are calculated through the use of the M_4 scheme and are then transferred appropriately to the immersed cell. The transfer of these values must be done in such a way as to respect the wall boundary conditions required to simulate the solid boundary. For a mirror cell m, given its neighbours $j = 1, 2, 3, ..., N$, the values at its center will be:

$$\vec{V}_m(i) = \begin{cases} \frac{\sum_{j=1}^{N} M_4(\rho_{m,j})\vec{V}_j(i)}{\sum_{j=1}^{N} M_4(\rho_{m,j})} & \text{if } 0 \leq |\rho| \leq 2, \quad (116a) \\ 0 & \text{if } |\rho| \geq 2, \quad (116b) \end{cases}$$

where $\vec{V}$ is given in 8 and i cycles through all primitive variables. Whether primitive or conservative variables are used it makes no difference in the final result. In order to simulate the effect of the solid boundary, in the immersed cell the $\vec{V}_m$ values are transferred as:

$$\begin{aligned} \rho_{imm} &= \rho_m \\ u_{imm} &= -u_m \\ v_{imm} &= -v_m \\ p_{imm} &= p_m \end{aligned} \tag{117}$$

Accordingly when the IDW interpolation scheme is used:

$$\vec{V}_m(i) = \begin{cases} \frac{\sum_{j=1}^{N} w(x_m, x_j)\vec{V}_j(i)}{\sum_{j=1}^{N} w(x_m, x_j)} & \text{if } d(x_m, x_j) \neq 0 \text{ and } d(x_m, x_j) \leq 2 \cdot dx, \quad (118a) \\ 0 & \text{if } d(x_m, x_j) > 2 \cdot dx, \quad (118b) \end{cases}$$

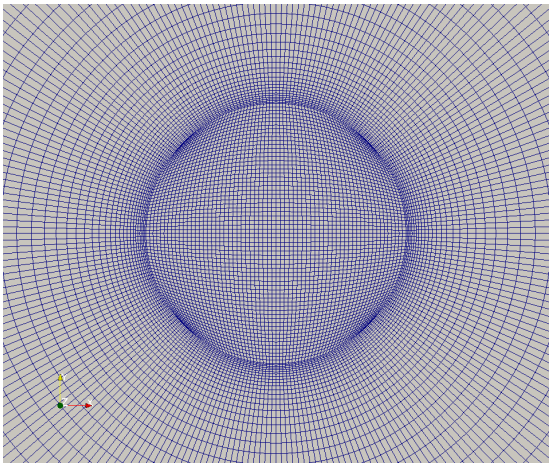
In this case, no mirror point is needed and the interpolation takes place immediately in the immersed cell.

4 Body-fitted grid without a physical solid boundary

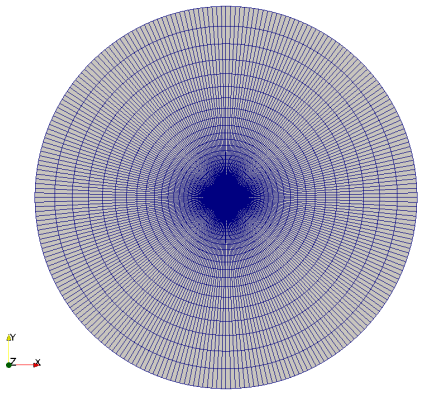
4.1 Introduction

In this chapter the flow around a circular cylinder will be examined with the use of a body-fitted grid which does not, however, have a predefined physical boundary. The results obtained with the use of the Immersed Boundary Method will be compared to those obtained with a typical body-fitted grid with the Gauss-Seidel method as well as those obtained from the study of Kim et al. ([1]).

A cylindrical geometry was studied. The solid body centered at $O(0, 0)$ with radius $r = 0.5$. For the cylinder two different methods were used for the implementation of the immersed boundary. The first method was based on the M_4 interpolation scheme as presented by [37] and the second method was based on the IDW method.



(a)



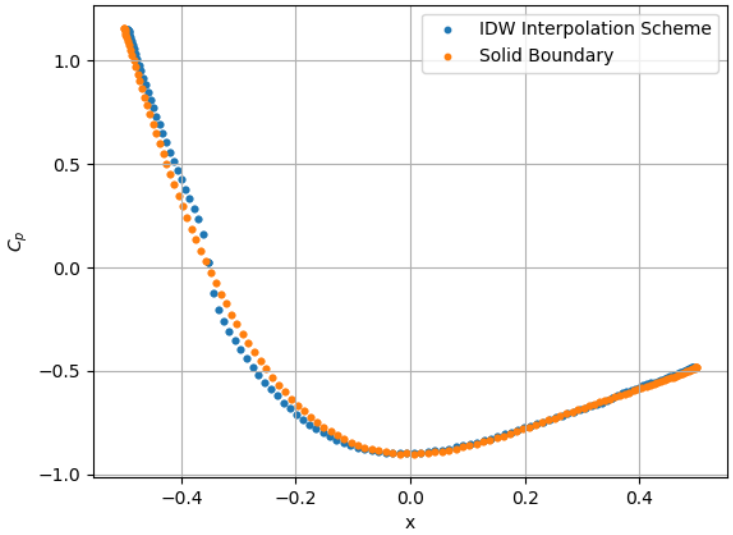
(b)

Figure 8: Computational domain for the cylinder problem. In 8a a closeup of the mesh where the cylindrical geometry is situated.

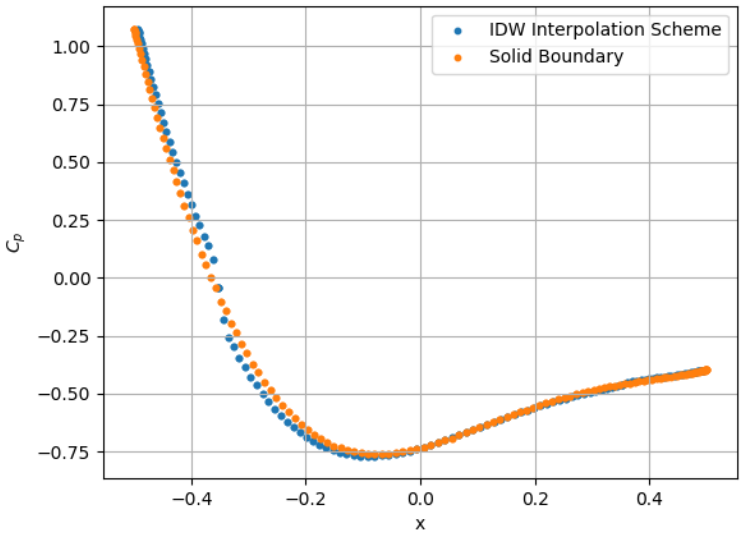
4.2 Inverse Distance Weighting Scheme

In all cases the flow was solved for low Reynolds numbers ($Re = 40, Re = 100$) and results comparable to those from the MaPFlow solver were obtained.

First the C_p diagrams are presented in Figures 9a,9b. For each case the flow has also been solved with MaPFlow and the C_p obtained through both calculations is presented.

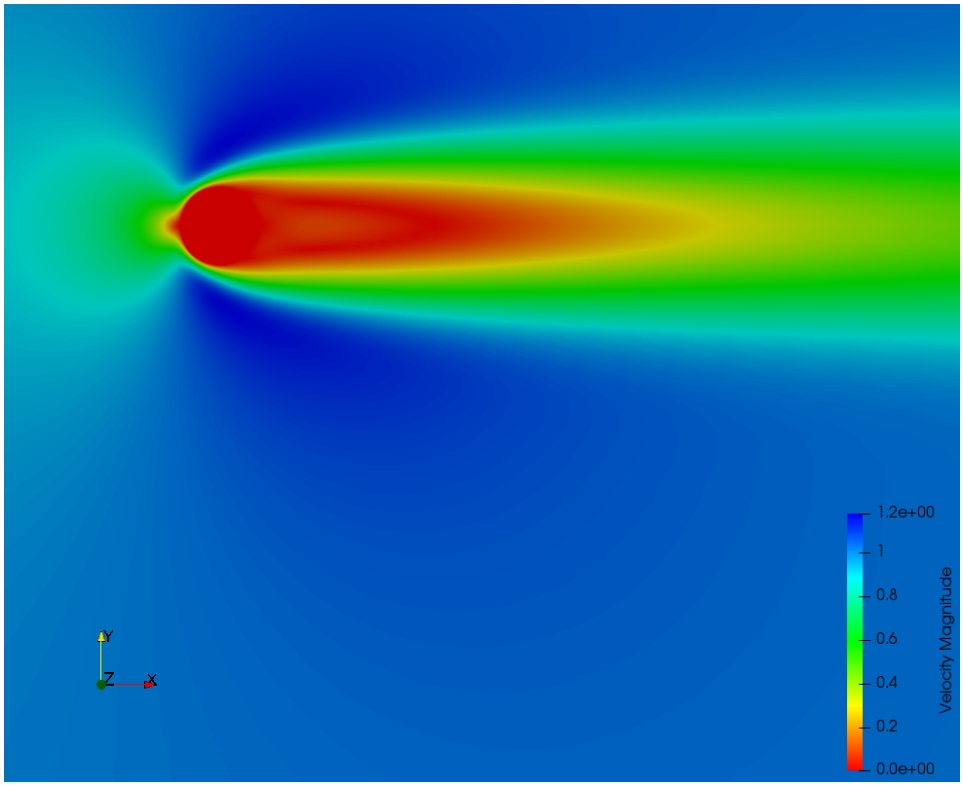


(a) $Re = 40$

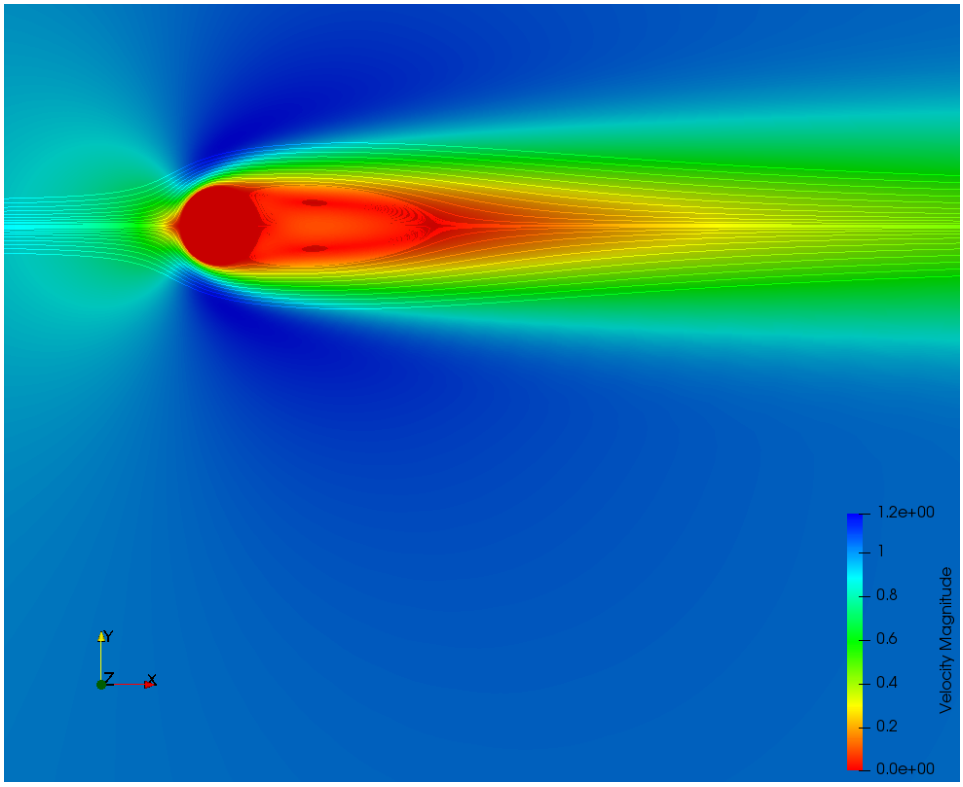


(b) $Re = 100$

Figure 9: C_p obtained with the use of the IBM with the IDW interpolation scheme for $Re = 40, 100$ compared to the results when a physical solid boundary exists.



(a)



(b)

Figure 10: Velocity distribution and flow lines with clear trailing edge symmetrical vortices for $Re = 40$ with the use of the IBM and the IDW interpolation scheme.

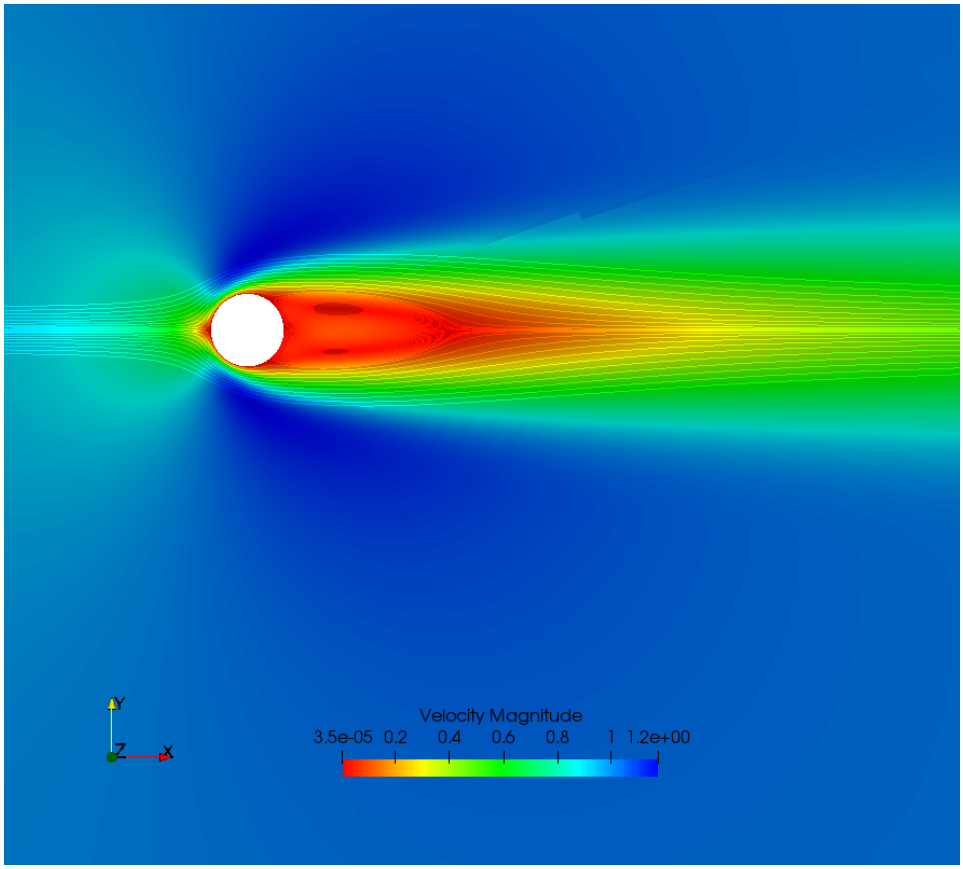
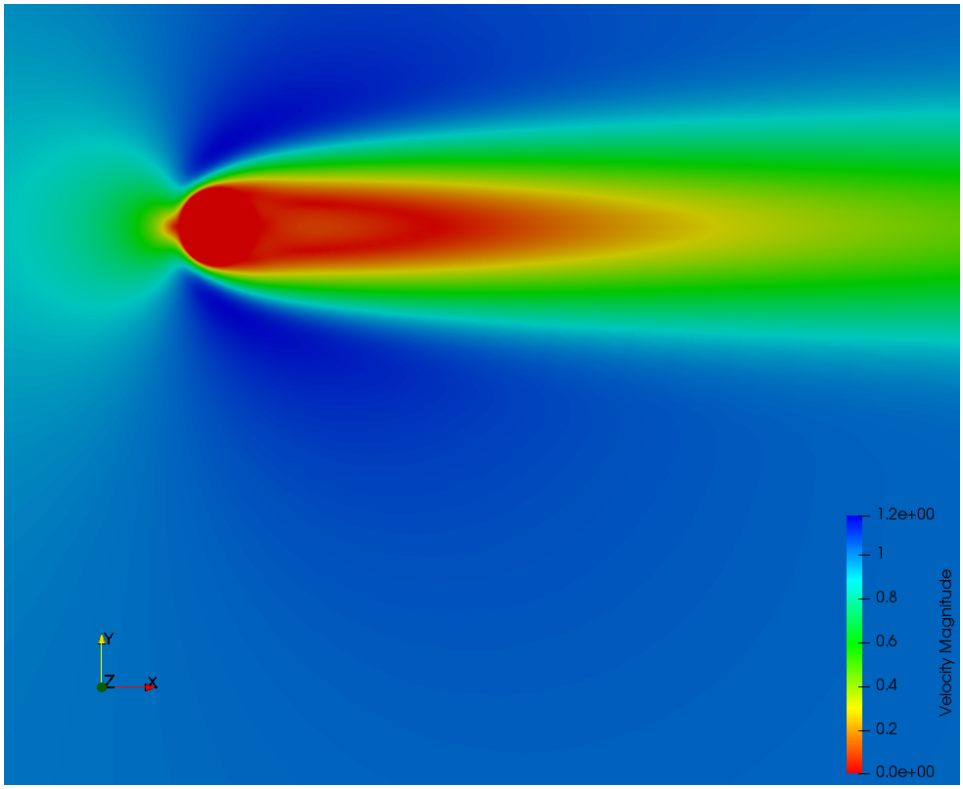
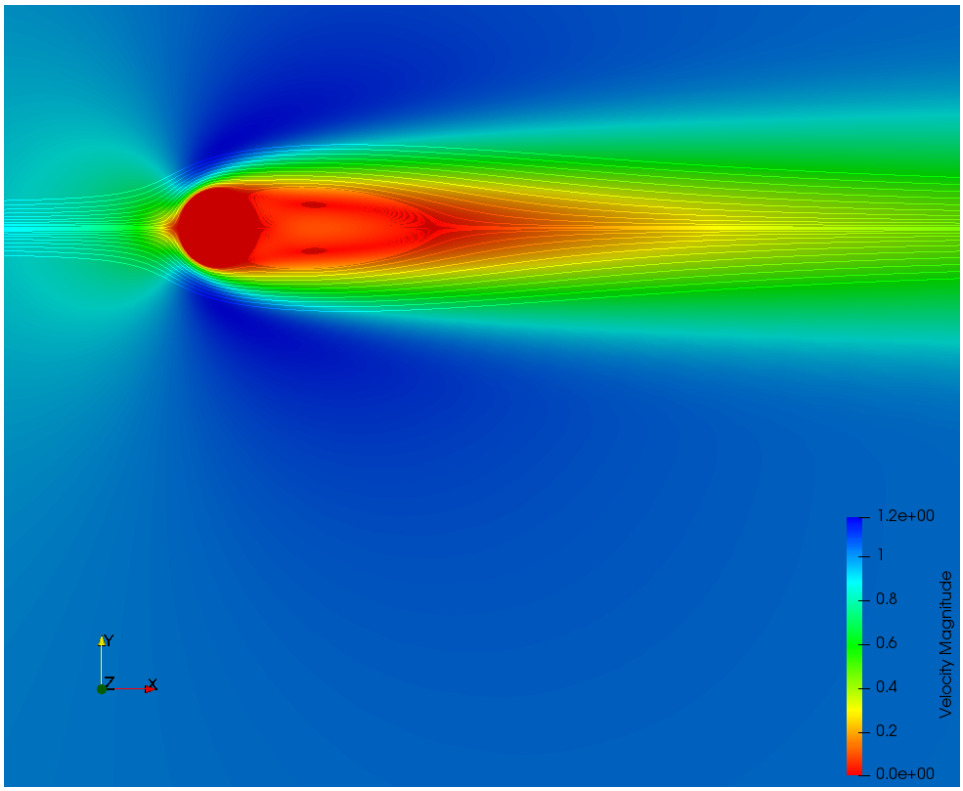


Figure 11: $Re = 40$. Velocity distribution and flow lines without the use of the Immersed Boundary Method.



(a)



(b)

Figure 12: $Re = 100$. Velocity distribution around the cylinder with flow-lines. The symmetrical trailing vortices can be seen as expected.

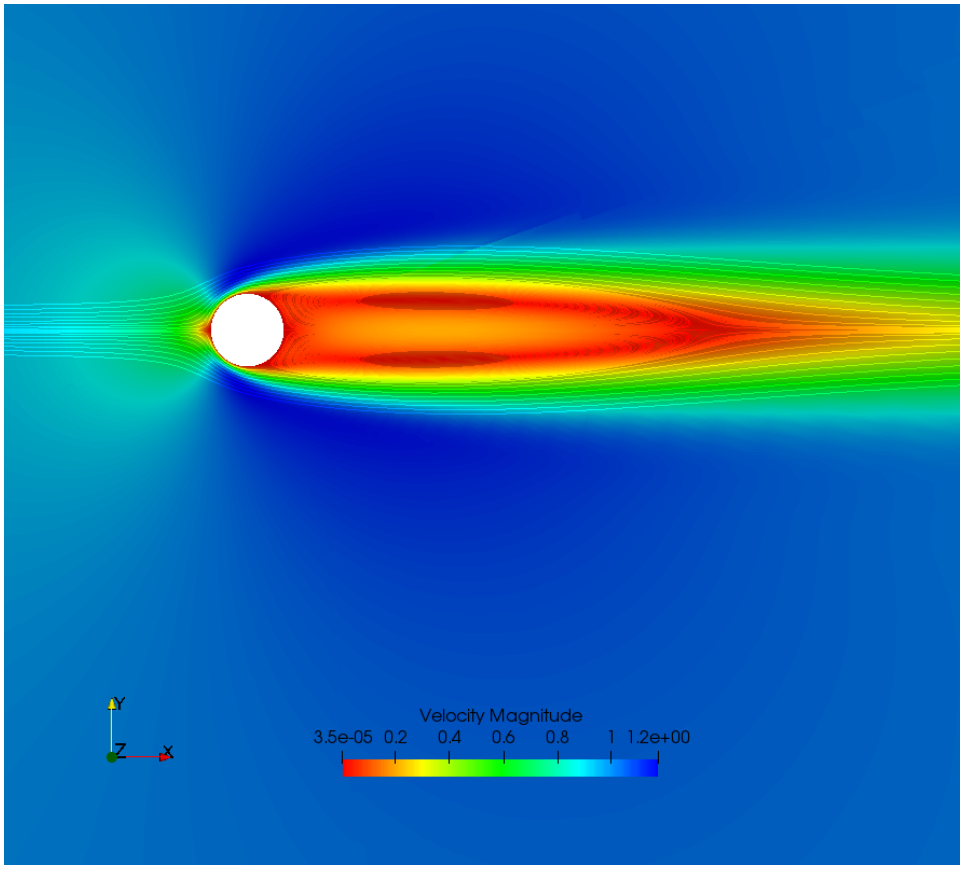
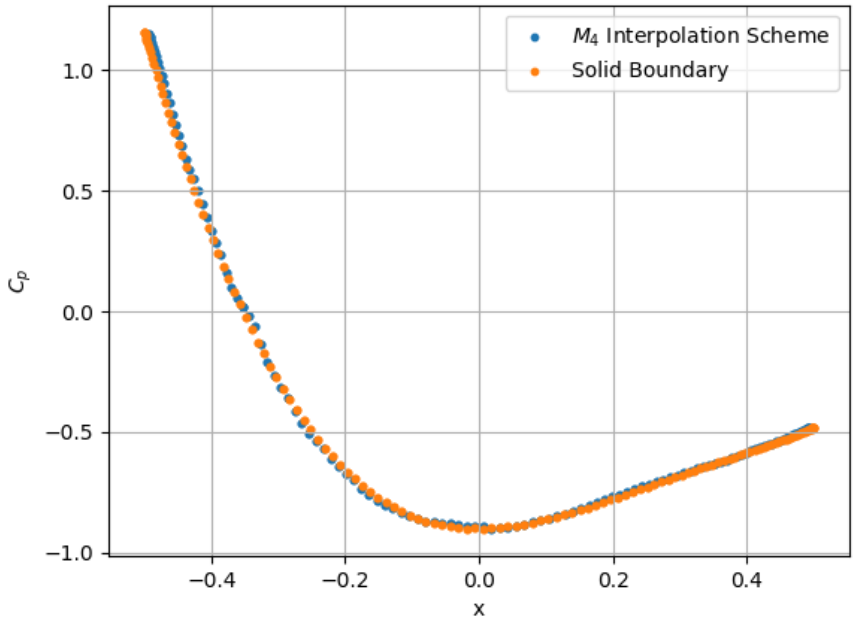
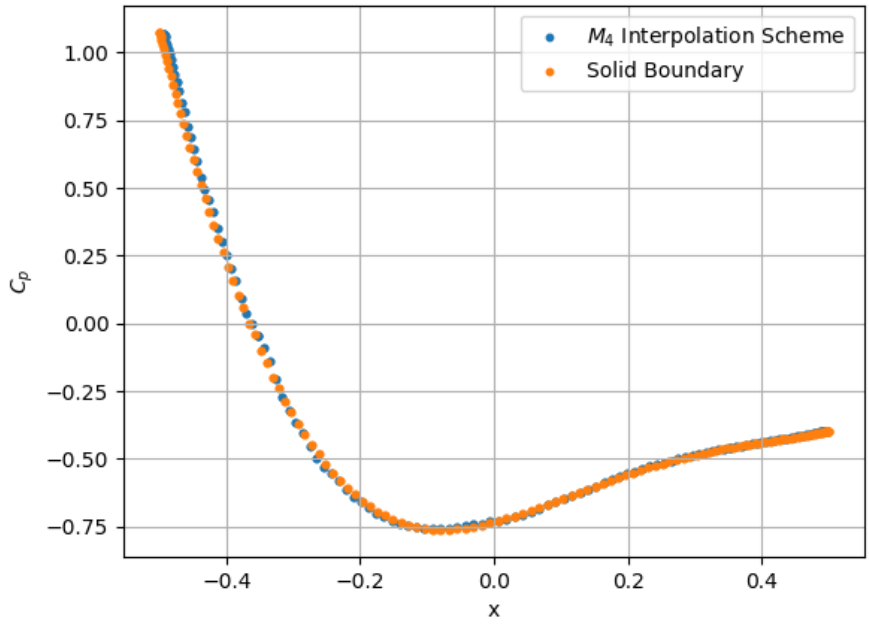


Figure 13: $Re = 100$. Velocity distribution without the use of the Immersed Boundary Method.

4.3 M_4 Interpolation Scheme

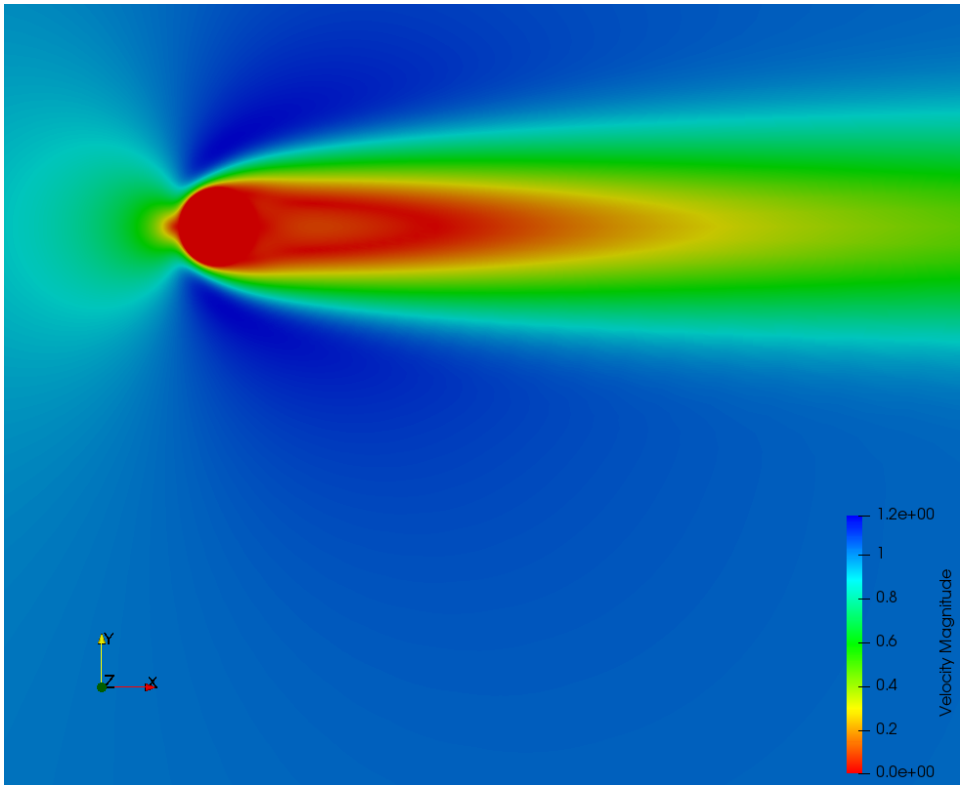


(a) $Re = 40$

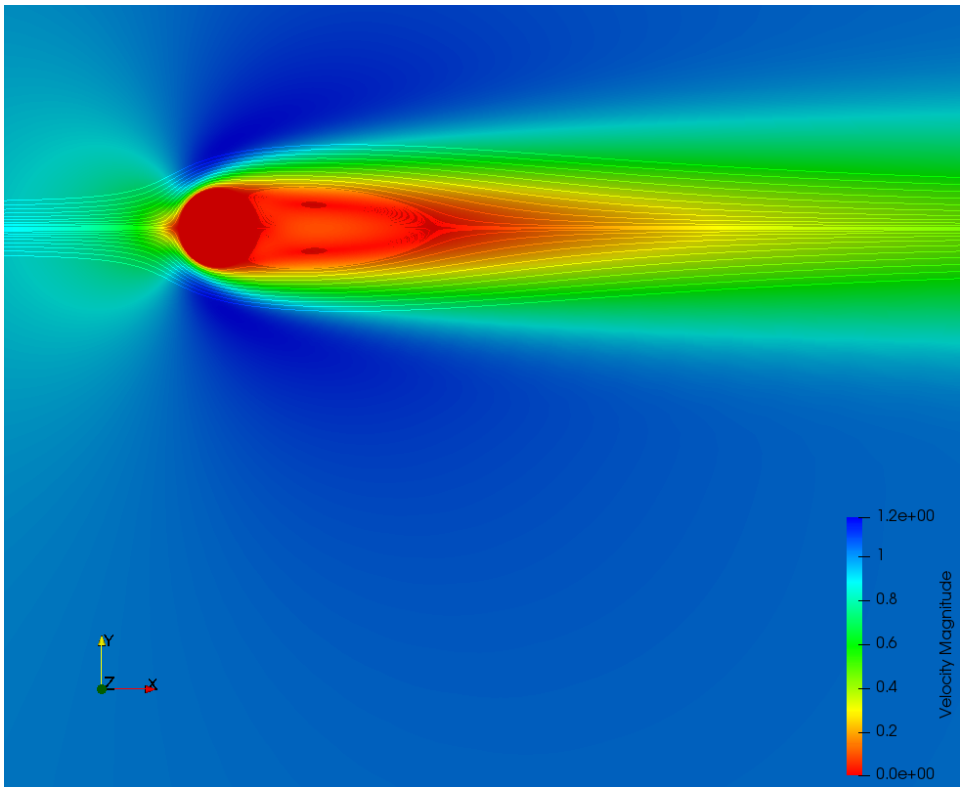


(b) $Re = 100$

Figure 14: C_p obtained with the use of the IBM with the M_4 interpolation scheme for $Re = 40, 100$ compared to the results when a physical solid boundary exists.

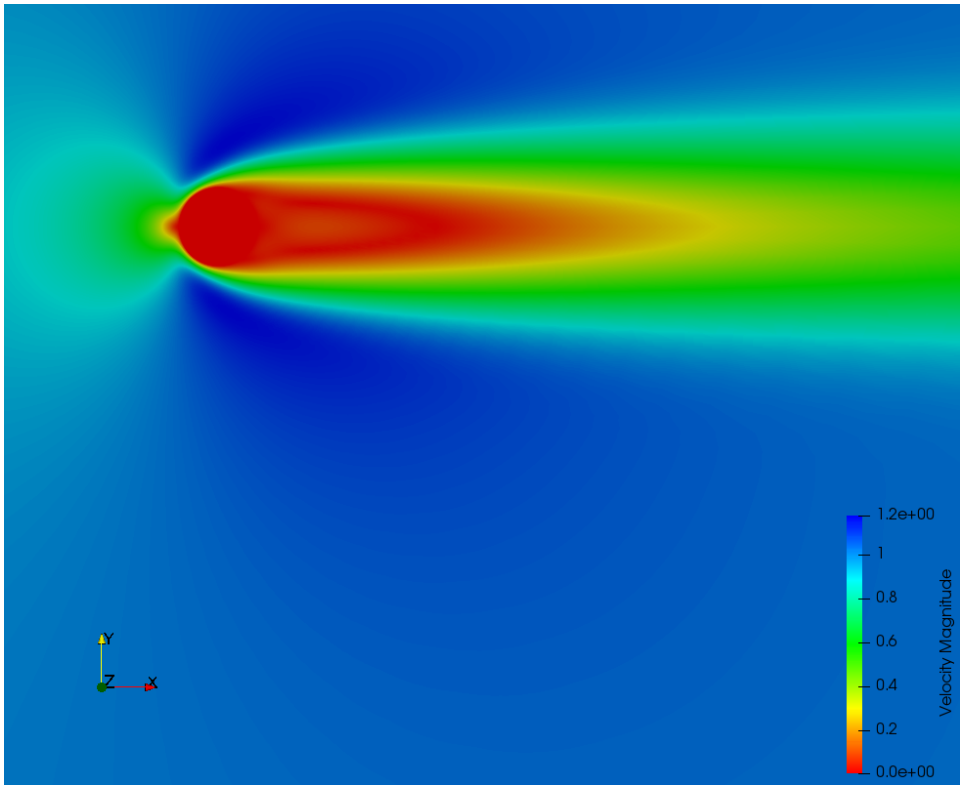


(a)

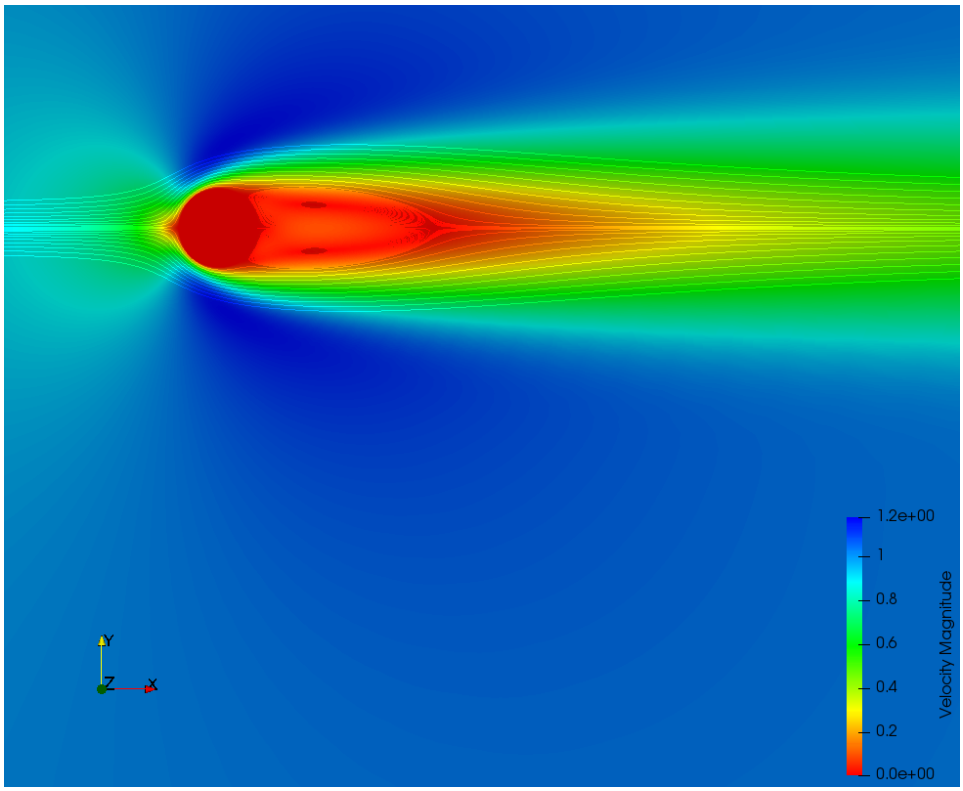


(b)

Figure 15: Velocity distribution and flow lines with clear trailing edge symmetrical vortices for $Re = 40$ with the use of the IBM and the M_4 interpolation scheme.



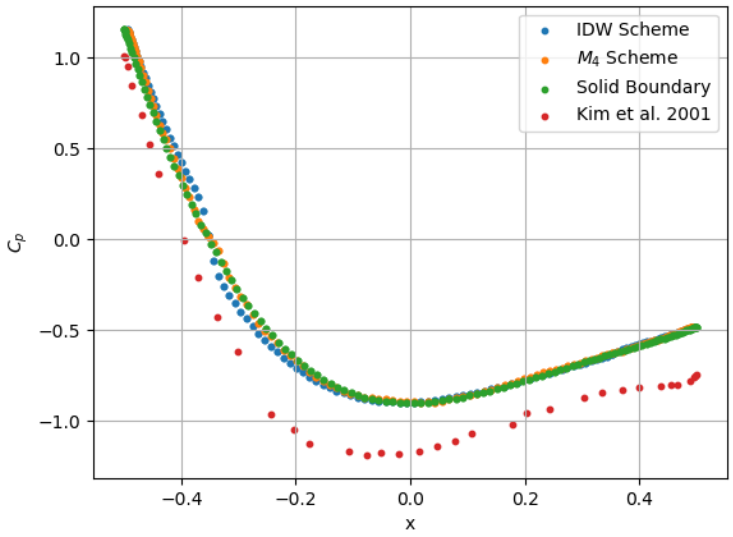
(a)



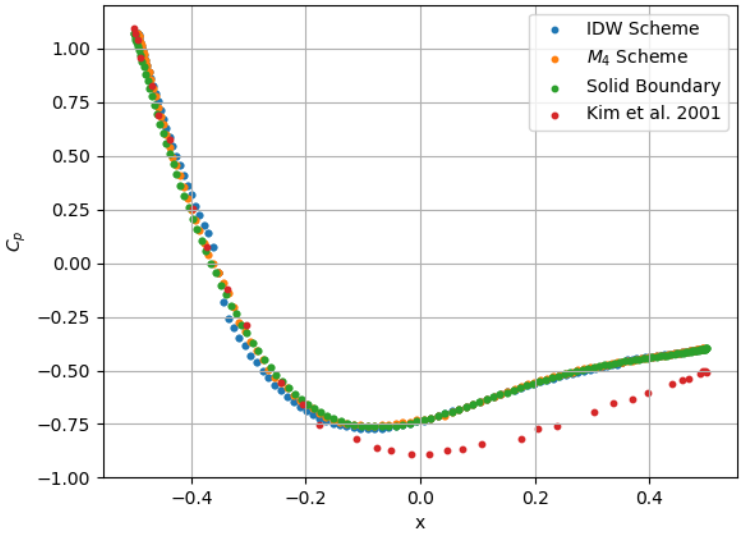
(b)

Figure 16: Velocity distribution and flow lines with clear trailing edge symmetrical vortices for $Re = 100$ with the use of the IBM and the M_4 interpolation scheme.

4.4 Comparative results



(a) $Re = 40$



(b) $Re = 100$

Figure 17: C_p for both Reynolds numbers with all interpolation schemes compared with a normal computation without the use of the Immersed Boundary Method. Results of the present thesis can be compared with those of Kim et al. [1].

The results obtained with MaPFlow ([2]) using the Immersed Boundary Method and the typical Gauss-Seidel solution method are in very good accordance with each other. This is normal due to the fact that the grid coincides with the boundary very precisely, there is no resolution issue near the boundary and the grid nodes inside the solid body are simply not taken into account. The solid boundary is not obtained through the implementation of the solid body boundary conditions as the faces of the cells already coincide with it.

5 Cartesian domains

The body is placed inside an O-type domain with a refined region of rectangular shape. The refinement is completely independent of the geometric properties of a body. For this reason, the flow was solved using the same grid both for the airfoil as well as for the cylinder. The effects of the Inverse Distance Weighting interpolation scheme were clearly worse than those of the M_4 scheme and therefore in the case of the Cartesian grids the M_4 interpolation scheme was used. Additionally, the effect of the discretization of the domain is examined. To this end, three grids were used with different $\delta x, \delta y$. Additionally, the calculations were performed for two Reynolds numbers, namely $Re = 40, Re = 100$ for the steady problem.

- $\delta x = \delta y = 0.02$,
- $\delta x = \delta y = 0.01$,
- $\delta x = \delta y = 0.005$

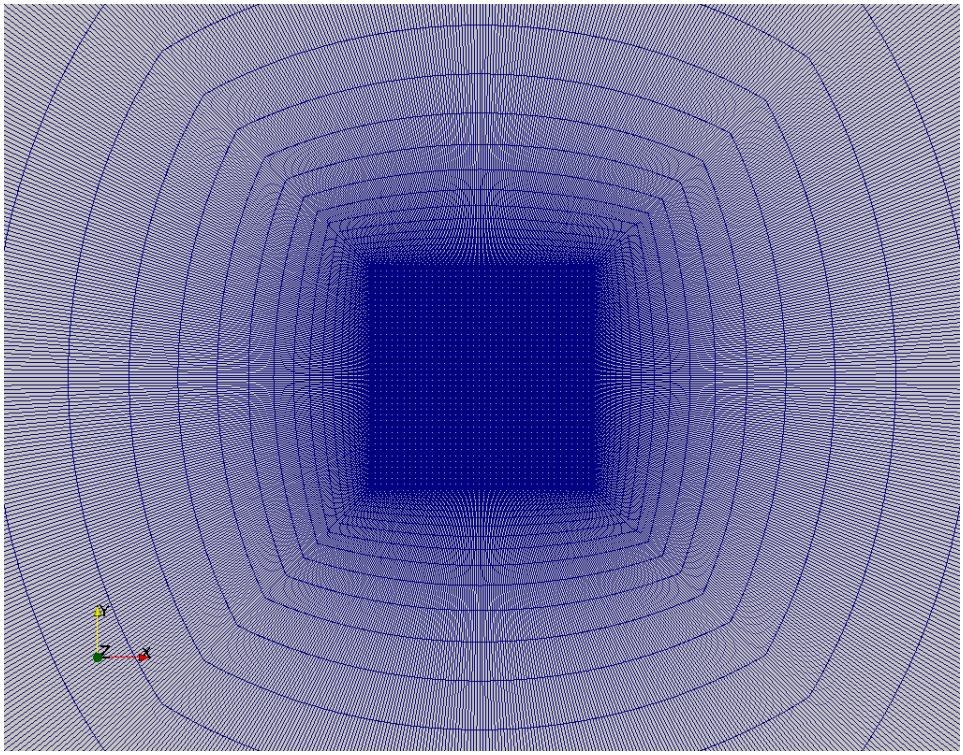
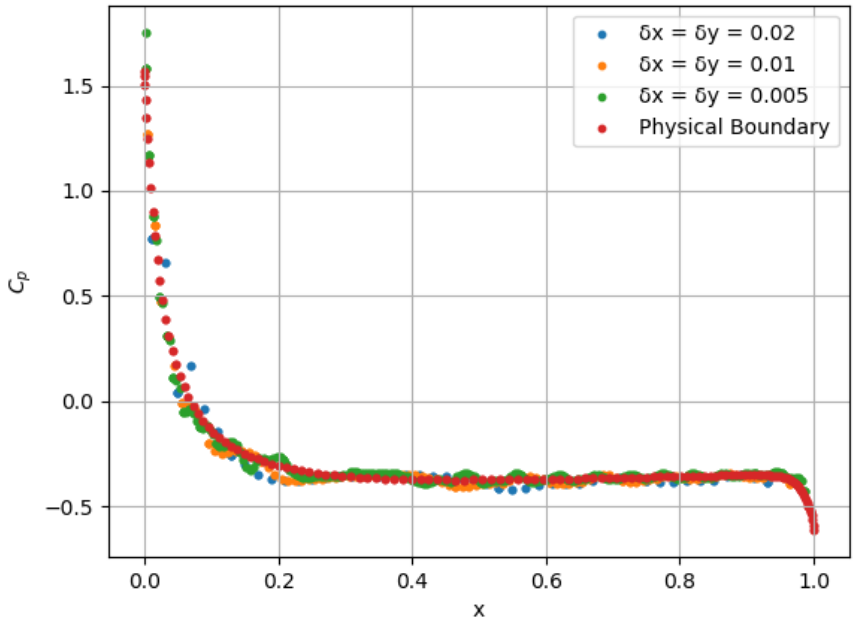
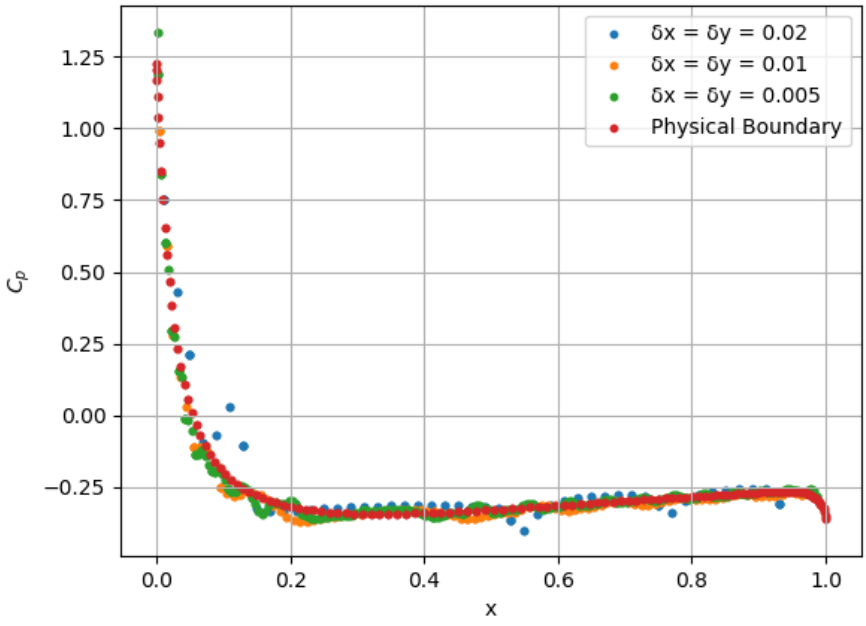


Figure 18: Rectangular, refined region of an O-type domain with $\delta x = \delta y = 0.01$.

5.1 Case 1: NACA 0012 Airfoil

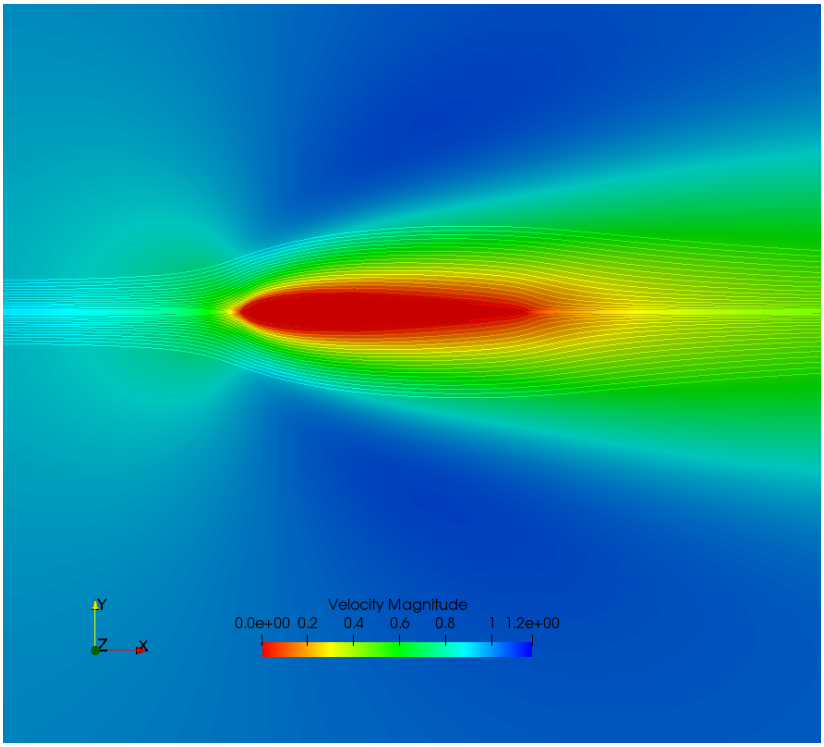


(a) $Re = 40$

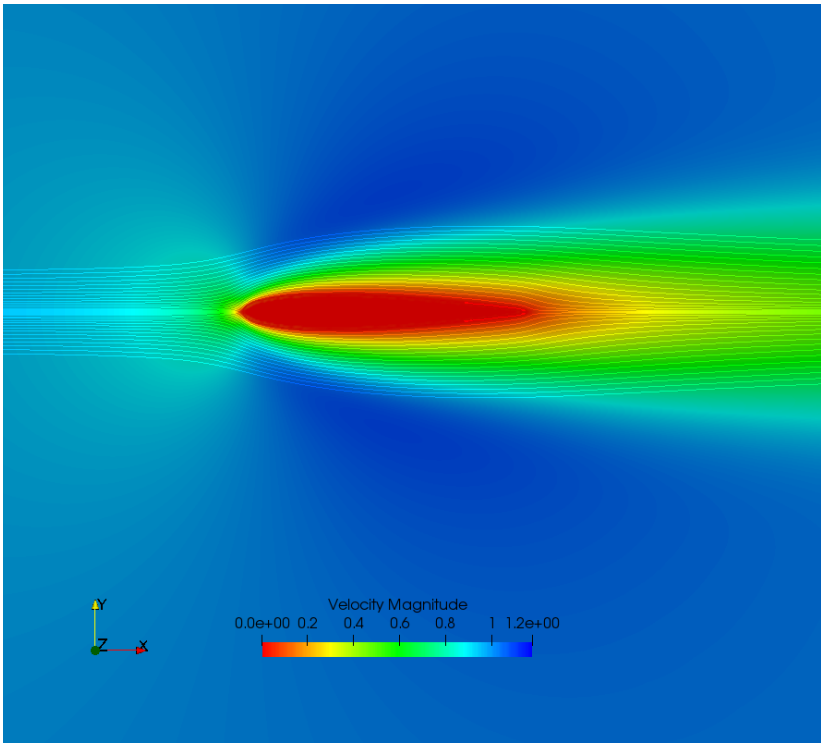


(b) $Re = 100$

Figure 19: C_p comparison for both Reynolds numbers, using the M_4 interpolation scheme with a Cartesian grid.



(a) $Re = 40$



(b) $Re = 100$

Figure 20: Flowfield for both Reynolds numbers when a grid with $\delta x = \delta y = 0.005$ is used.

From the C_p diagrams (figure 19 an oscilating behavior can be observed for $Re = 100$ near the maximum thickness of the airfoil for the mesh with the larger discretization ($\delta x = \delta y = 0.02$). The resolution of the mesh, in this case, is insufficient to describe faithfully the flow characteristics near the location of the

maximum thickness which is situated at 12% of the chord length which is equal to 1. The higher Reynolds number (compared to $Re = 40$ where no such problems occur) causes this effect to occur because the viscous effects are not as strong and the flow is more easily separated from the solid body.

5.1.1 Angle of attack 1^o

The flow around the NACA-0012 airfoil was simulated when the body is placed with an angle $\theta = 1^o$ and $Re = 100$. Results compared with normal MaPFlow run with a body-fitted grid using the Gauss-Seidel solution method.

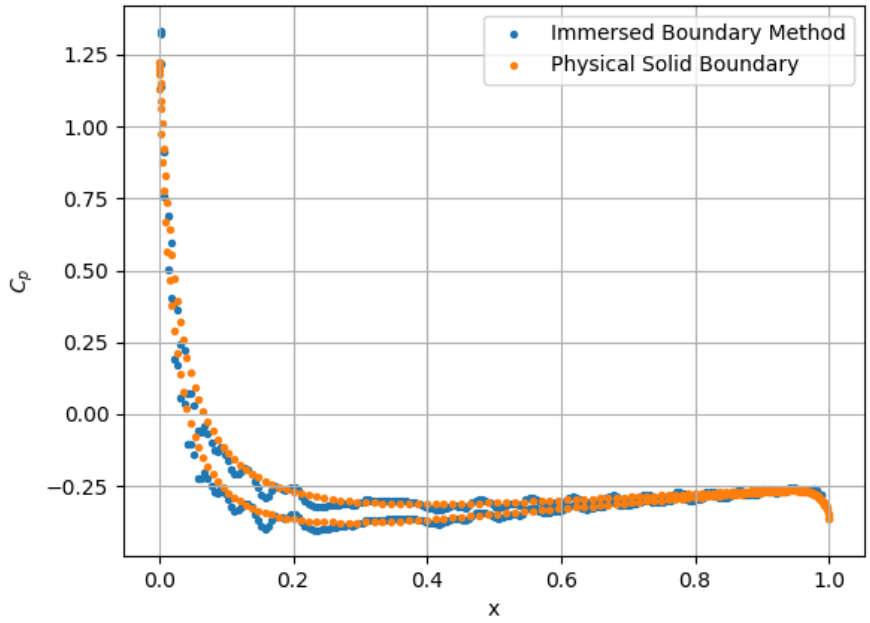
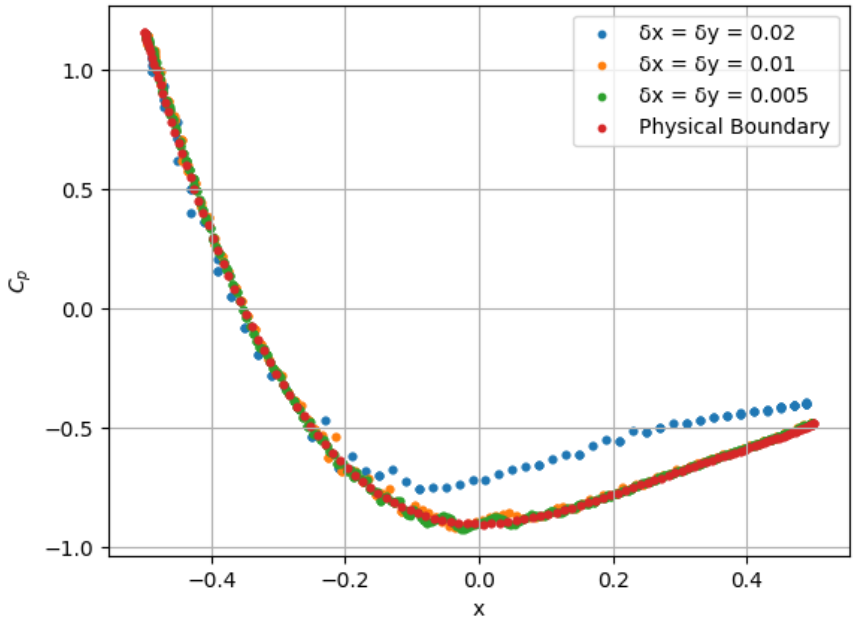


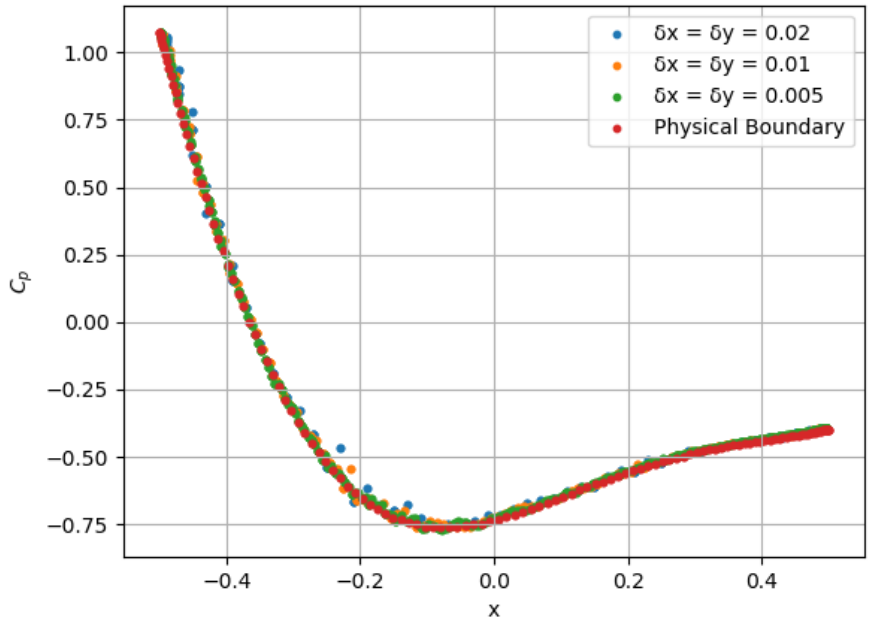
Figure 21: C_p comparison for $\theta = 1^o$ and $Re = 100$.

Oscillations are observed throughout the length of the airfoil. A possible reason for that is the stair-like layout of the mesh near the solid boundary as well as the absence of filtering from the results. Nonetheless there is a remarkable matching with the reference results which can be made only better if the Immersed Boundary Method results were to be filtered.

5.2 Case 2: Circular cylinder

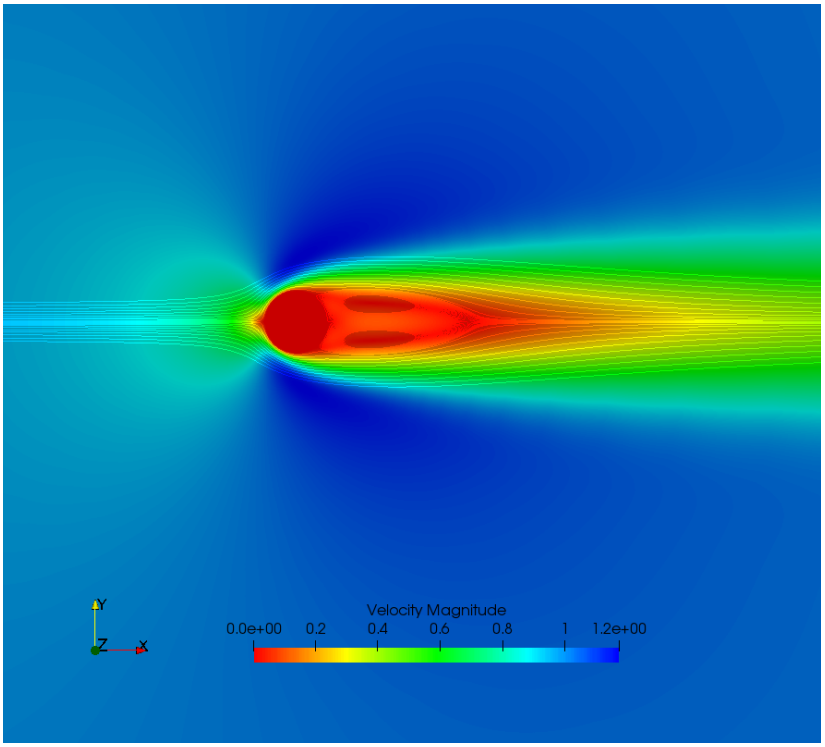


(a) $Re = 40$

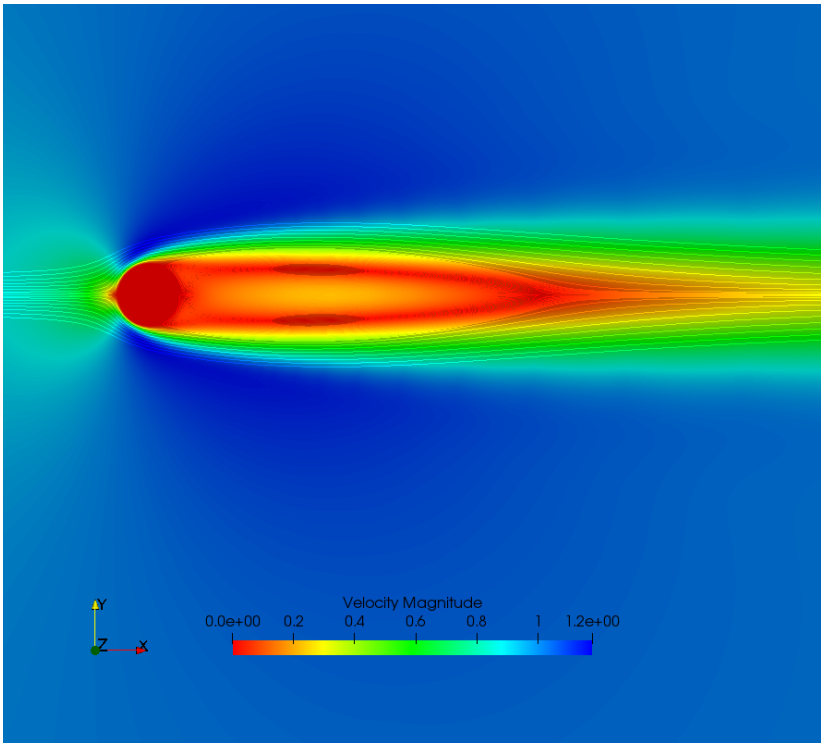


(b) $Re = 100$

Figure 22: C_p comparison for both Reynolds numbers, using the M_4 interpolation scheme with a Cartesian grid.



(a) $Re = 40$



(b) $Re = 100$

Figure 23: Flowfield for both Reynolds numbers when a grid with $\delta x = \delta y = 0.005$ is used.

Regarding the C_p diagrams (figure 22) a divergent behavior is observed in the case of the results for $Re = 40$ using the grid with $\delta x = \delta y = 0.02$. This is due to an overestimation of the viscous effects. The acceleration of the flow is underestimated due to the discretization of the grid and the maximum thickness of the solid body. A larger discretization means that the boundary layer is overestimated and the flow is not accelerated as it should. This leads to a smaller acceleration of the flow and, thus, higher C_p values. This effect is reinforced by the low Reynolds number and the prevalence of the viscous effects. For that reason, it is not observed when the Reynolds number is higher ($Re = 100$). Additionally, the discretization of the mesh has an effect on the thickness of the boundary layer.

5.2.1 Unsteady flow

At $Re = 200$ an unsteady flow was simulated around the cylinder with a time-step of 0.01 (total number of timesteps 20000) for two meshes with $\delta x = \delta y = 0.02$ and $\delta x = \delta y = 0.01$. The results of the Immersed Boundary Method show no significant difference with the typical solution method given there is no turbulence modelling.

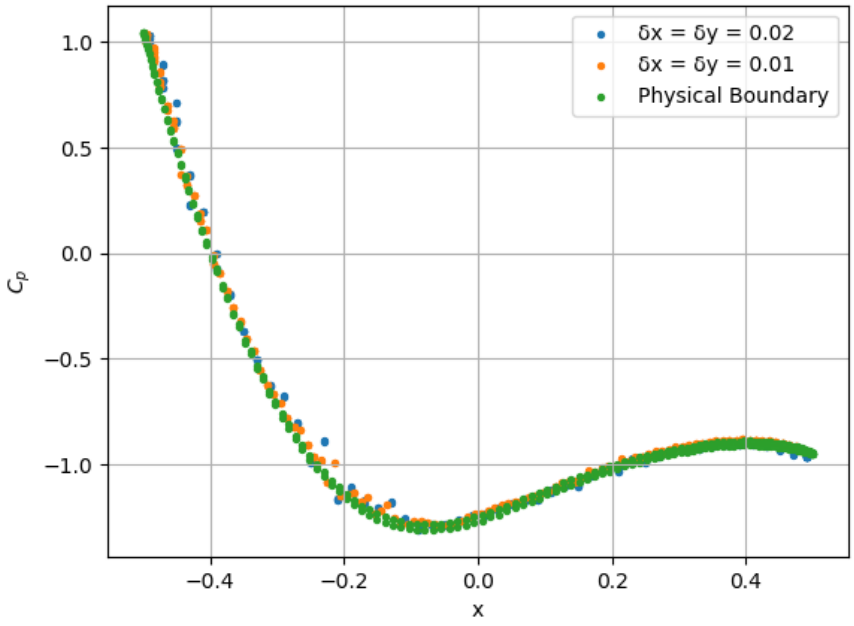
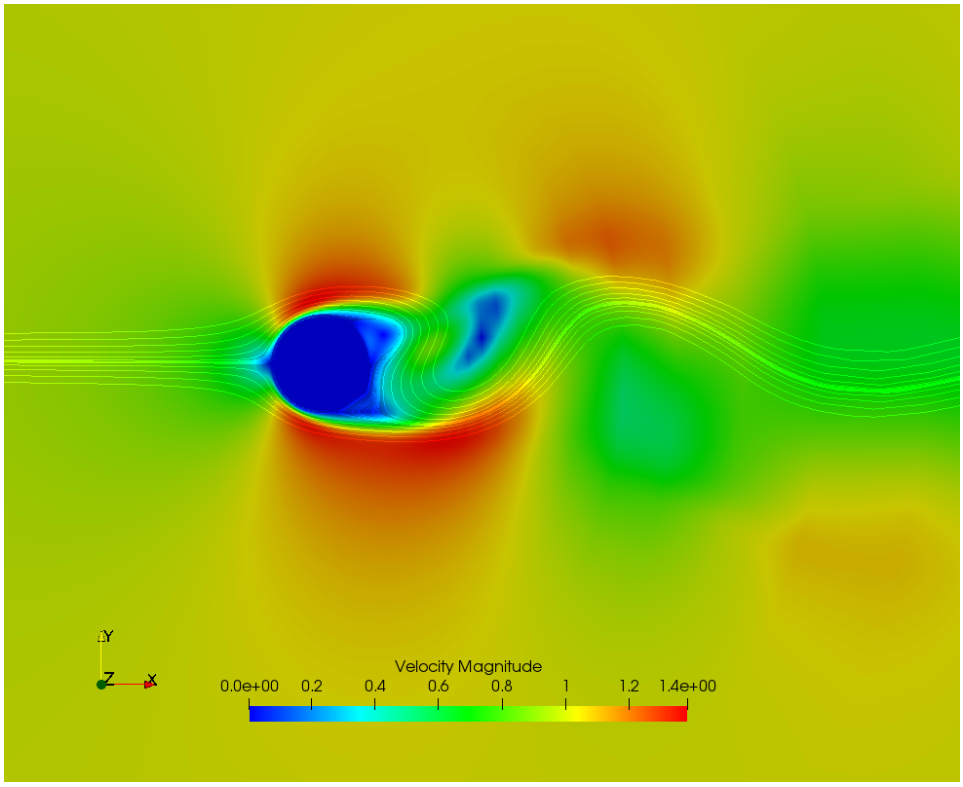
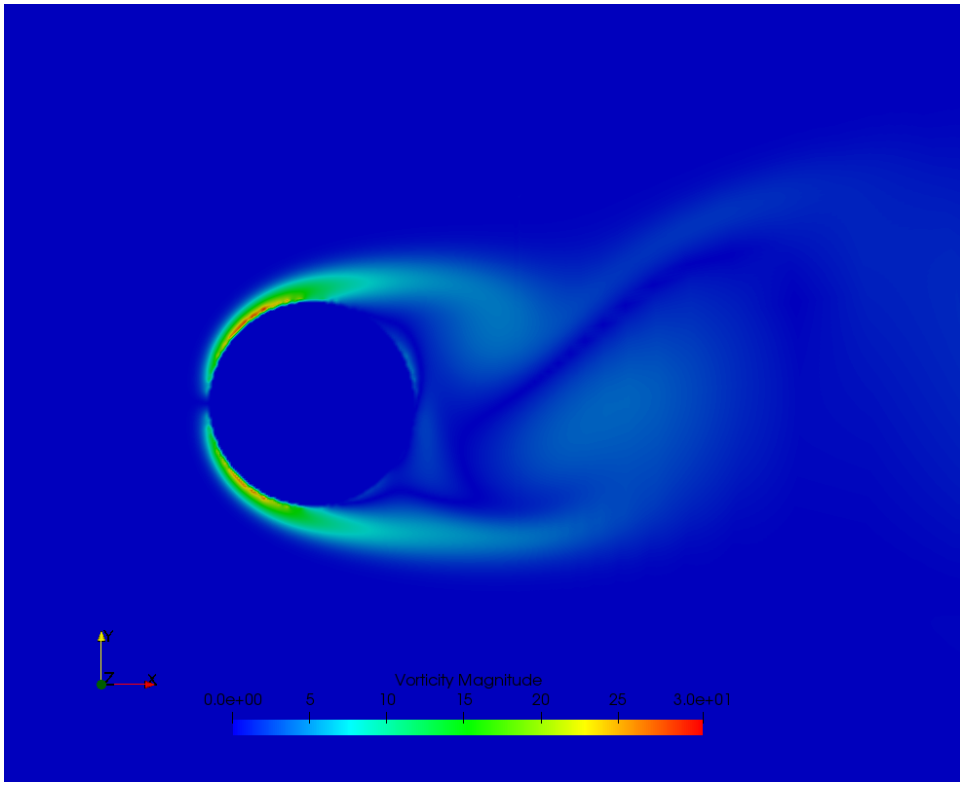


Figure 24: C_p comparison for the Immersed Boundary Method and the typical Gauss-Seidel method with a body-fitted grid for the unsteady case. $Re = 200$.



(a) Velocity field



(b) Vorticity field

Figure 25: Velocity and vorticity fields resulting from the Immersed Boundary Method for $\delta x = \delta y = 0.01$. $Re = 200$.

6 Concluding remarks

In the present thesis, the solid boundary was not respected neither from the mesh geometry nor from the cell-faces. The main goal was to satisfy the boundary conditions were the solid boundary was located. Additionally the boundary conditions are implemented asynchronously. In each time step, the values inside the flow cells are computed and only then the immersed cells are given a value.

The Immersed Boundary Method gave acceptable results for two-dimensional, external, laminar, low Reynolds number flows. For both steady and unsteady cases the results were very close to those obtained with a body-fitted grid and the Gauss-Seidel solution method. For the unsteady problem the results were, of course, averaged after a sufficient number of time steps (15000). Turbulence methods should also be implemented alongisde the I.B.M, thus enabling it to solve more complicated problems. A possible step towards this direction would be a uniform distribution of the distances between immersed cells and the solid boundary. Oscillations observed in the results are partly due to the fact that each immersed point is randomly distanced from the solid boundary. In all cases, however, finer grids gave results in better accordance with the reference values. This led to an increase in computational time as the MaPFlow solver, when using the Immersed Boundary Method, could not be used as a parallel solver. However, the main advantage of the I.B.M is its ability to save substancial pre-processing time and make the computation overall less time expensive. Serial computing is therefore not suitable for this method as it makes computations more costly. Finally, no moving-bodies problems were studied which is an area where the Immersed Boundary Method can prove to be exceptionally useful as it will remedy the need for a reconstruction of the mesh in each time step.

References

[1] J. Kim, D. Kim, and H. Choi, "An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries," *Journal of Computational Physics*, 2001.

[2] G. Papadakis and S. G. Voutsinas, "In view of accelerating CFD simulations through coupling with vortex particle approximations," *The Science of Making Torque from Wind Journal of Physics: Conference Series*, vol. 524, no. 1, 2014.

[3] C. Hirsch, *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*. 2007.

[4] Y. H. H. Choi and C. L. Merkle, "The Application of Preconditioning in Viscous Flows," *Journal of Computational Physics*, vol. 105, pp. 207–223, apr 1993.

[5] L.-E. Eriksson, "A Preconditioned Navier-Stokes Solver for Low Mach Number Flows," *Computers and Fluids*, pp. 199–205, 1996.

[6] V. G. Asouti, D. I. Papadimitriou, D. G. Koubogiannis, and K. C. Giannakoglou, "Low Mach Number Preconditioning for 2D and 3D Upwind Flow Solution Schemes on Unstructured Grids," *5th GRACM International Congress on Computational Mechanics*, 2005.

[7] Y. Colin, H. Deniau, and J. F. Boussuge, "A robust low speed preconditioning formulation for viscous flow computations," *Computers and Fluids*, vol. 47, pp. 1–15, aug 2011.

[8] V. Venkatakrishnan, "On the Accuracy of Limiters and Convergence to Steady State Solutions," *AIAA Paper 93-0880*, 1993.

[9] B. van Leer, "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method," *Journal of Computational Physics*, vol. 32, no. 1, pp. 101–136, 1979.

[10] S. K. Godunov, "A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics," *Math-Net.Ru (translated)*, vol. 47, no. 3, pp. 271–306, 1959.

[11] P. L. Roe, "Characteristic-Based Schemes for the Euler Equations," *Annual Review of Fluid Mechanics*, vol. 18, no. 1, pp. 337–365, 1986.

[12] G. D. van Albada, B. van Leer, and W. W. Roberts, "A Comparative Study of Computational Methods in Cosmic Gas Dynamics," *Astronomy and Astrophysics*, vol. 108, 1982.

[13] B. P. Leonard, "A stable and accurate convective modelling procedure based on quadratic upstream interpolation," *Computer Methods in Applied Mechanics and Engineering*, vol. 19, no. 1, pp. 59–98, 1979.

[14] B. P. Leonard, "Simple high-accuracy resolution program for convective modelling of discontinuities," *International Journal for Numerical Methods in Fluids*, vol. 8, no. 10, pp. 1291–1318, 1988.

[15] A. Jameson, W. Schmidt, and E. Turkel, "AIAA 1981 – 1259 Numerical Solution of the Euler Equations by Finite Volume Methods Schemes AIAA 14th Fluid and Plasma Dynamic Conference Palo Alto , California," *Convergence*, vol. M, no. AIAA 81-1259, pp. 1–19, 1981.

[16] B. van Leer, "Flux-vector splitting for the Euler equations," in *Computers & Fluids*, vol. 9, pp. 507–512, dec 1982.

[17] J. L. Steger and R. F. Warming, "Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods," *Journal of Computational Physics*, vol. 40, pp. 263–293, apr 1981.

[18] P. L. Roe, "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, vol. 43, pp. 357–372, 1981.

[19] V. G. Asouti, *Aerodynamic Analysis and Design Methods at High and Low Speed Flows on Multiprocessor Platforms*. PhD thesis, National Technical University of Athens, 2009.

[20] J. Blazek, *Computational fluid dynamics:Principles of Grid Generation*. 2001.

[21] R. Biedron, V. Vatsa, and H. Atkins, "Simulation of Unsteady Flows Using an Unstructured Navier-Stokes Solver on Moving and Stationary Grids," in *23rd AIAA Applied Aerodynamics Conference*, no. June, (Reston, Virigina), American Institute of Aeronautics and Astronautics, jun 2005.

[22] V. Vatsa, M. Carpenter, and D. Lockard, "Re-evaluation of an Optimized Second Order Backward Difference (BDF2OPT) Scheme for Unsteady Flow Applications," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, (Reston, Virigina), pp. 1–15, American Institute of Aeronautics and Astronautics, jan 2010.

[23] D. J. Mavriplis and V. Venkatakrishnan, "Agglomeration Multigrid for Viscous Turbulent Flows," *ICASE report*, vol. 24, no. 5, pp. 553–570, 1994.

[24] D. J. Mavriplis and A. Jameson, "Multigrid solution of the Navier-Stokes equations on triangular meshes," *AIAA Journal*, vol. 28, pp. 1415–1425, aug 1990.

[25] C. Hirsch, "Numerical Computation of Internal and External Flows," 1990.

[26] K. Hejranfar and R. Kamali-Moghadam, "Preconditioned characteristic boundary conditions for solution of the preconditioned Euler equations at low Mach number flows," *Journal of Computational Physics*, vol. 231, no. 12, pp. 4384–4402, 2012.

[27] D. G. Koubogiannis, L. C. Poussoulidis, D. V. Rovas, and K. C. Giannakoglou, "Solution of flow problems using unstructured grids on distributed memory platforms," *Computer Methods in Applied Mechanics and Engineering*, vol. 160, pp. 89–100, jul 1998.

[28] Y. Saad, "Iterative Methods for Sparse Linear Systems Second Edition Yousef Saad," 2003.

[29] C. S. Peskin, "The fluid dynamics of computational methods," *Annual Review*, no. 1969, pp. 235–259, 1982.

[30] D. Goldstein, R. Handler, and L. Sirovich, "Modeling a No-Slip Flow Boundary with an External Force Field," 1993.

[31] J. Mohd-Yusof, "Combined immersed boundary/B-spline methods for simulation of flow in complex geometries," *Center for Turbulence Research Annual Research Briefs1*, pp. 317–328, 1997.

[32] E. A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof, "Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations," *Journal of Computational Physics*, vol. 161, no. 1, pp. 35–60, 2000.

[33] T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy, "A Cartesian grid method for viscous incompressible flows with complex immersed boundaries," *14th Computational Fluid Dynamics Conference*, vol. 240, pp. 547–557, 1999.

[34] N. Zhang and Z. C. Zheng, "An improved direct-forcing immersed-boundary method for finite difference applications," *Journal of Computational Physics*, vol. 221, no. 1, pp. 250–268, 2007.

[35] Y. Bazilevs, M. Hsu, J. Kiendl, R. Wüchner, and K. Bletzinger, "3D Simulation of Wind Turbine Rotors at Full Scale. Part II: Fluid – Structure Interaction Modeling with Composite Blades," *International Journal for Numerical Methods in Fluids*, vol. 65, no. October 2010, pp. 236–253, 2011.

[36] SHEPARD D, "Two- dimensional interpolation function for irregularly- spaced data," *Proc 23rd Nat Conf*, pp. 517–524, 1968.

[37] A. Palha, L. Manickathan, C. S. Ferreira, and G. van Bussel, "A hybrid Eulerian-Lagrangian flow solver," 2015.