TMR4900 - Master Thesis

Meshfree Least Square-based Finite Difference method in CFD applications

by

Pål Grøthe Sandnes

Trondheim, June, 2011

Supervisor: Bjørnar Pettersen
Co-supervisor: Håvard Holm

NTNU
Faculty of Engineering Science
and Technology
Department of Marine Technology

Development of a Navier-Stokes solver

Master thesis 2011
for
Stud.techn. Pål Grøthe Sandnes

The candidate shall develop a Navier-Stokes solver based on the meshfree least square-based finite difference method, which shall be well documented. All assumption taken with respect to the numerical methods– and modeling shall be reasoned. The solver shall be limited to two dimensional, incompressible and laminar flow regimes, and tested against well known solutions for:

- Couette flow
- Poiseuille flow
- Lid-driven cavity flow
- Viscous flow past a circular cylinder

The thesis should be organized in a rational manner to give a clear exposition of results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Telegraphic language should be avoided.

The thesis shall contain the following elements: A text defining the scope, preface, list of contents, summary, main body of thesis, conclusions with recommendations for further work, list of symbols and acronyms, references and (optional) appendices. All figures, tables and equations shall be numerated.

The supervisors may require that the candidate, in an early stage of the work, presents a written plan for the completion of the work. The plan should include a budget for the use of computer and laboratory resources which will be charged to the department. Overruns shall be reported to the supervisors.

The original contribution of the candidate and material taken from other sources shall be clearly defined. Work from other sources shall be properly referenced using an acknowledged referencing system.

The report shall be submitted in two copies:

- Signed by the candidate
- The text defining the scope included
- In bound volume(s)

Abstract

Most commercial computational fluid dynamics (CFD) packages available today are based on the finite volume– or finite element method. Both of these methods have been proven robust, efficient and appropriate for complex geometries. However, due to their crucial dependence on a well constructed grid, extensive preliminary work have to be invested in order to obtain satisfying results. During the last decades, several so-called meshfree methods have been proposed with the intension of entirely eliminating the grid dependence. Instead of a grid, meshfree methods use the nodal coordinates directly in order to calculate the spatial derivatives.

In this master thesis, the meshfree least square-based finite difference (LSFD) method has been considered. The method has initially been thoroughly derived and tested for a simple Poisson equation. With its promising numerical performance, it has further been applied to the full Navier-Stokes equations, describing fluid motions in a continuum media. Several numerical methods used to solve the incompressible Navier-Stokes equations have been proposed, and some of them have also been presented in this thesis. However, the temporal discretization has finally been done using a 1st order semi-implicit projection method, for which the primitive variables (velocity and pressure) are solved directly. In order to verify the developed meshfree LSFD code, in total four flow problems have been considered. All of these cases are well known due to their benchmarking relevance, and LSFD performs well compared to both earlier observations and theory.

Even though the developed program in this thesis only supports two dimensional, incompressible and laminar flow regimes, the idea of meshfree LSFD is quite general and may very well be applied to more complex flows, including turbulence.

Acknowledgments

This master thesis has been carried out at the Department of Marine Technology at the Norwegian University of Science and Technology (NTNU) in Trondheim, and marks the end of the five years M.Sc. program.

I would first like to thank my supervisors Professor Bjørnar Pettersen and co-superviser Håvard Holm for their both technical and motivational guidance and support during this spring. With their help it has been possible to develop the meshfree Navier-Stokes solver presented in this thesis.

The motivation for studying the meshfree LSFD method originates from my exchange year at National University of Singapore (NUS) in 2009/2010, where the subject was treated in a course given by Professor Shu Chang. Among others at the Department of Mechanical Engineering at NUS, Professor Chang has been contributing severely in the research on meshfree methods. I would like to thank Professor Chang for the answers he has provided me on the LSFD method and its implementation.

Finally, thanks to Pål Christian Hagen at Ceetron for his technical support on GLview.

Pål Grøthe Sandnes

Trondheim, June 2011

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

BC	Boundary Condition
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
GFD	General Finite Difference
IC	Initial Condition
LSFD	Least Square-based Finite Difference
NS	Navier-Stokes
OOA	Order Of Accuracy
PDE	Partial Differential Equation
SOR	Successive Over-Relaxation
VIV	Vortex Induced Vibrations

Greek Symbols

Δ	Difference operator
ϵ_{tol}	Convergence tolerance
η	Normal vector
μ	Dynamic viscosity
∇	Divergence operator
∇^2	Laplace operator

ν	Kinematic viscosity
Ω	Angular velocity, chapter 5
Ω	Physical domain
ω	Relaxation factor
ψ	Stream function
ρ	Density
τ	Tangential vector
ζ	Vorticity
U_η	Normal velocity
U_τ	Tangential velocity

Roman Letters

c	Courant number
f_v	Vortex shedding frequency
h	Node spacing
p	Pressure
r	Radius
Re	Reynolds number
St	Strouhal number
Ta	Taylor number
u	Velocity x-component
U_∞	Incident uniform velocity
u_θ	Angular velocity component
v	Velocity y-component
w	Velocity z-component

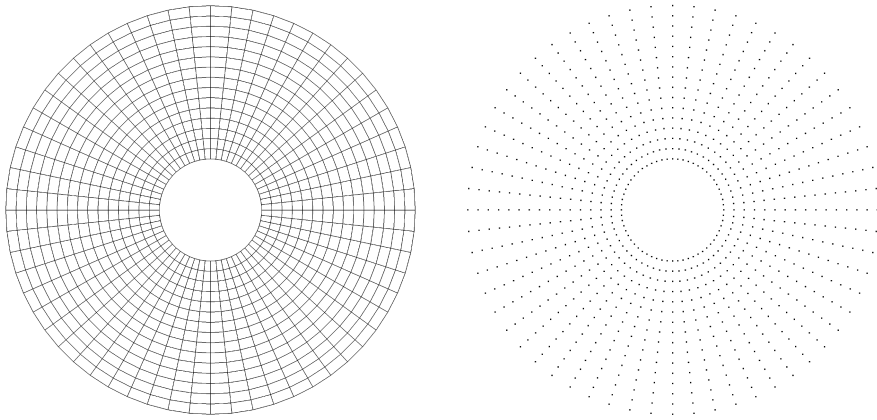
Chapter 1

Introduction

The ability to mathematically describe physical phenomenons in nature is essential in most fields of science and engineering. The models describing complexities in nature are called partial differential equations (PDE's) and are in general difficult if not even impossible to solve analytically. In order to solve problems of high complexity, the reliance on numerical methods has therefore been crucial.

Traditionally, the favored methods applied to problems in fluid dynamics are the finite difference method (FDM), finite element method (FEM) and finite volume method (FVM). While FEM and FVM are robust and well suited for complex geometries, they are both more computational expensive than the FDM. FDM on the other hand, represents difficulties when applied to complex geometries due to its reliance on coordinate transformation, which in fact is impossible for fairly complex geometries. However, FDM performs efficiently on regular domains.

The flexibility in geometrical complexity has made FVM and FEM the preferred methods in commercial packages. Despite their strengths they both have an important common shortcoming. They need a mesh with certain qualities fulfilled in order to approximate an accurate solution. Not only does the meshing procedure often turn out to be the most time consuming part of the analysis, but also mesh adaptivity represents a tremendous challenge. In the last decade extensive research has therefore been done in the field of so-called meshfree methods. In a meshfree method, the main idea is basically to eliminate the entire meshing procedure and instead use nodal coordinate information in order to obtain an approximation to the PDE. Instead of mesh generation, the meshfree methods rely on node generation which is considered an easier and faster procedure in terms of computational effort.



(a) Mesh representation (b) Node representation

Figure 1.1: Mesh– and node-based geometry representation.

Adaptivity is also much easier in meshfree procedures because nodes may simply be added or removed from the domain, without having to re-mesh and verify the mesh quality for every adaption step.

Several meshfree methods have been proposed to date. In this thesis however, only the least square-based finite difference (LSFD) method has been considered. The LSFD procedure can be viewed as a further development from an earlier meshfree method called general finite-difference (GFD). The idea of LSFD was proposed by Ding et al. [6], and has been treated in chapter 2. LSFD is a method used to approximate the spatial derivatives and may in general be applied to any desired PDE. By combining LSFD with a temporal scheme for solving the Navier-Stokes equations, its potential can be fully utilized.

Earlier, computational fluid dynamics (CFD) had no serious commercial interest simply because the computers were too weak at that moment. With increasing availability in computational power, the situation today has turned entirely. While experimental– and empirical methods earlier were the main sources for engineering analysis, most of todays effort is done in CFD and later confirmed with experiments. The complexity of problems such as CFD today can handle is increasing, and the accuracy continuously gets better. The implementation of the fluid dynamics equations has been discussed in chapter 3, and the LSFD method has further ben programmed throughly, and executed for a set of problem cases. The programming issues have been considered in chapter 4, while the analysis have been performed and discussed in chapter 5.

Chapter 2

Development of the LSFD scheme

One of the shortcomings in conventional finite difference method is that the derivatives are approximated along one dimensional paths in a multidimensional domain. For very simple geometries this is satisfactory, but as more complex geometries are considered, difficulties may quickly arise. This is a great motivation for developing a more general discretization method for multi dimensional geometries. The LSFD method presented in this thesis, was originally developed by Ding et al. [6].

2.1 Taylor series expansion

In conventional finite difference method, the functional value at every node in the domain is approximated by one dimensional Taylor series expansion on its adjacent neighbors.

$$f\left(x+\Delta x\right)=\sum_{n=0}^{N}\left(\frac{\Delta x^{n}}{n!}\frac{\partial^{n}f}{\partial x^{n}}\right)+\underbrace{O\left(\Delta x^{N+1}\right)}_{\text{Order of accuracy}} \tag{2.1}$$

In equation (2.1), N is the desired number of truncation terms. To illustrate how the derivatives can be approximated, three points with individual distance Δx are considered in figure 2.1.

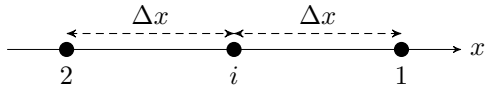


Figure 2.1: One dimensional node distribution

Taylor expansion up to 2nd order is introduced on node i in order to approximate the functional values on node 1 and 2.

$$f_1 = f_i + \Delta x \frac{\partial f_i}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 f_i}{\partial x^2} + O\left(\Delta x^3\right) \tag{2.2}$$

$$f_2 = f_i - \Delta x \frac{\partial f_i}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 f_i}{\partial x^2} + O\left(\Delta x^3\right) \tag{2.3}$$

By combining equation (2.2) and (2.3), the derivatives of f_i can be approximated as

$$\frac{\partial f_i}{\partial x} = \frac{f_1 - f_2}{2\Delta x} + O\left(\Delta x^2\right) \tag{2.4}$$

$$\frac{\partial^2 f_i}{\partial x^2} = \frac{f_2 - 2f_i + f_1}{\Delta x^2} + O\left(\Delta x\right) \tag{2.5}$$

Equation (2.4) and (2.5) are *central difference* approximations to the 1st– and 2nd order derivatives of f_i because they are found by evaluating the functional values in both spatial directions. If the Taylor expansion in equation (2.2) and (2.3) were only to be considered up to 1st order, f_1 and f_2 would be expressed as

$$f_1 = f_i + \Delta x \frac{\partial f_i}{\partial x} + O\left(\Delta x^2\right) \tag{2.6}$$

$$f_2 = f_i - \Delta x \frac{\partial f_i}{\partial x} + O\left(\Delta x^2\right) \tag{2.7}$$

giving two different approximations to the 1st order derivative of f_i.

$$\frac{\partial f_i}{\partial x} = \frac{f_1 - f_i}{\Delta x} + O\left(\Delta x\right) \tag{2.8}$$

$$\frac{\partial f_i}{\partial x} = \frac{f_i - f_2}{\Delta x} + O\left(\Delta x\right) \tag{2.9}$$

Equation (2.8) and (2.9) are the *forward difference*– and *backward difference* approximations. Obviously expression (2.4), (2.8) and (2.9) approximate the same

derivative. However, the central difference expression in (2.4) provides one order of accuracy higher. In general, if an n-order derivative with p-order of accuracy is required, Taylor expansion have to be applied to $(n + p - 1) = j$ neighbor nodes. Exceptions to this rule do however exist. If for instance Taylor expansion on node i in figure 2.1 is truncated up to 3rd order, the system of equations would initially be expected to be underdetermined due to more unknown derivative terms than neighbor nodes.

$$f_1 = f_i + \Delta x \frac{\partial f_i}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 f_i}{\partial x^2} + \frac{\Delta x^3}{6} \frac{\partial^3 f_i}{\partial x^3} + O\left(\Delta x^4\right) \tag{2.10}$$

$$f_2 = f_i - \Delta x \frac{\partial f_i}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 f_i}{\partial x^2} - \frac{\Delta x^3}{6} \frac{\partial^3 f_i}{\partial x^3} + O\left(\Delta x^4\right) \tag{2.11}$$

Combining equation (2.10) and (2.11) by elimination of the 3rd order derivative gives the following expression.

$$\frac{\partial^2 f_i}{\partial x^2} = \frac{f_2 - 2f_i + f_1}{\Delta x^2} + O\left(\Delta x^2\right) \tag{2.12}$$

The 1st order derivative term vanish automatically, giving one order of accuracy higher than expected. Equation (2.5) will therefore show 2nd order of accuracy, not 1st.

The concept of Taylor series approximations of derivatives can easily be extended to higher dimensions. Two dimensional Taylor series expansion is generally written as

$$f\left(x + \Delta x, y + \Delta y\right) = \sum_{n=0}^{N} \left(\sum_{k=0}^{n} \left(\frac{\Delta x^{n-k} \Delta y^k}{(n-k)!k!} \frac{\partial^n f}{\partial x^{n-k} \partial y^k} \right) \right) + \underbrace{O\left(\Delta x^{N+1}, \Delta y^{N+1}\right)}_{\text{Order of accuracy}} \tag{2.13}$$

In two dimensional Taylor series expansion, there are no path-alignment restrictions on the nodes. This provides much higher flexibility, though more neighbor nodes are needed in order to evaluate the same derivatives as earlier, because the cross derivatives will have to be encountered for as well.

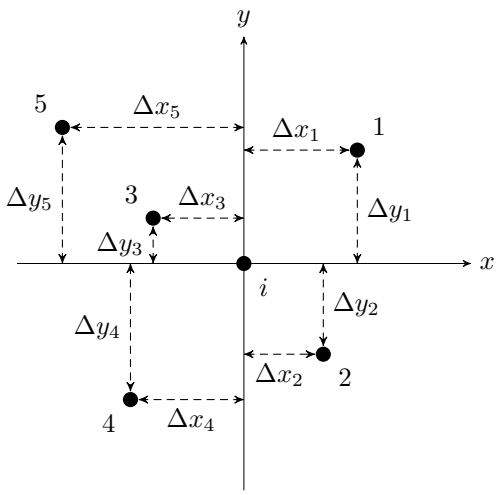


Figure 2.2: Two dimensional node distribution

In order to illustrate how two dimensional Taylor series expansion can be performed, the node distribution in figure 2.2 will be considered during the following. To approximate the derivatives of f_i up to 2nd order, $D = 5$ number of derivative terms have to be considered. Therefore $N = 5$ number of neighbor nodes are needed in order to determine all unknowns. As earlier, the functional values at each neighbor node can be expressed as

$$f_j = f_i + \Delta x_j \frac{\partial f_i}{\partial x} + \Delta y_j \frac{\partial f_i}{\partial y} + \frac{\Delta x_j^2}{2} \frac{\partial^2 f_i}{\partial x^2} + \Delta x_j \Delta y_j \frac{\partial^2 f_i}{\partial x \partial y} + \frac{\Delta y_j^2}{2} \frac{\partial^2 f_i}{\partial y^2} + O\left(\Delta x_j^3, \Delta y_j^3\right) \tag{2.14}$$

where $j \in \{1, 2, ..., N-1, N\}$, and $(\Delta x_j, \Delta y_j)$ denotes the distance from the reference node i to the neighbor node j. Expression (2.14) can be reformulated into a matrix system of the form

$$\boldsymbol{\Delta f} = \boldsymbol{S df} \tag{2.15}$$

where $\boldsymbol{\Delta f}$ is the $N \times 1$ array

$$\boldsymbol{\Delta f} = \left[f_j - f_i \right] \tag{2.16}$$

$\boldsymbol{S}$ is the $N \times D$ matrix

$$\boldsymbol{S} = \begin{bmatrix} \Delta x_j & \Delta y_j & \frac{1}{2}\Delta x_j^2 & \Delta x_j \Delta y_j & \frac{1}{2}\Delta y_j^2 \end{bmatrix} \tag{2.17}$$

and $\boldsymbol{df}$ is the $D \times 1$ array

$$\boldsymbol{df} = \begin{bmatrix} \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial y} & \frac{\partial^2 f_i}{\partial x^2} & \frac{\partial^2 f_i}{\partial x \partial y} & \frac{\partial^2 f_i}{\partial y^2} \end{bmatrix}^T \tag{2.18}$$

$\boldsymbol{S}$ is a square matrix if $N = D$ and invertible given non-singular matrix condition. The derivatives of f_i can therefore be obtained as

$$\boldsymbol{df} = \boldsymbol{S}^{-1} \boldsymbol{\Delta f} \tag{2.19}$$

The numerical condition of $\boldsymbol{S}$ is critically dependent on the node distribution. Even though the matrix is non-singular it tends to become ill-conditioned if the neighbor nodes are located too near the reference, or too near the other neighbors. To overcome these issues, the LSFD method uses local scaling and weighted least squares technique.

2.2 Local scaling

The supporting node located furthest from its reference i in figure 2.3, will define the size d_i of the local domain Ω_i comprising all the neighbor nodes.

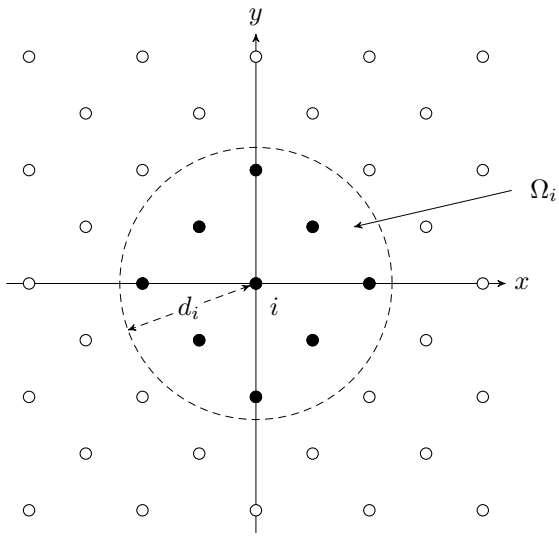


Figure 2.3: Local subdomain

All the spatial differences inside the local domain is then scaled to the domain size $d_i = 1.2 \cdot \max\left(\sqrt{\Delta x_{ij}^2 + \Delta y_{ij}^2}\right)$ [26], improving the numerical condition of the coefficient matrix $\boldsymbol{S}$.

$$\Delta \overline{x}_j = \frac{\Delta x_j}{d_i} \tag{2.20}$$

$$\Delta \overline{y}_j = \frac{\Delta y_j}{d_i} \tag{2.21}$$

By defining $\overline{\boldsymbol{S}} = \boldsymbol{S}\boldsymbol{D}$, where $\boldsymbol{D}$ is the diagonal matrix

$$\boldsymbol{D} = \text{diag}\begin{bmatrix} d_i^{-1} & d_i^{-1} & d_i^{-2} & d_i^{-2} & d_i^{-2} \end{bmatrix} \tag{2.22}$$

Equation (2.19) can be written as

$$\boldsymbol{df} = \boldsymbol{D}\overline{\boldsymbol{S}}^{-1}\boldsymbol{\Delta f} \tag{2.23}$$

2.3 Weighted least-squares approximation

The implementation of local scaling deals mainly with the relation between the reference node and its neighbors. The distribution of nodes inside the local domain may also cause an ill-conditioned coefficient matrix if neighbor nodes are located too close to each other. One way to deal with this issue could be to eliminate the nodes causing the problem. However, this approach may severely increase the computational time [6]. An alternative way is to apply the least squares technique. The least squares technique is a procedure used to approximate the solution of an overdetermined set of equations. An approximation to $\boldsymbol{df}$ in equation (2.15) may therefore be achieved by introducing the definition [21].

$$\boldsymbol{S}^T \boldsymbol{\Delta f} = \boldsymbol{S}^T \boldsymbol{S} \boldsymbol{df} \tag{2.24}$$

The right hand side-term $\boldsymbol{S}^T \boldsymbol{S}$ in equation (2.24) is a definite square matrix and can hence be inverted in order to obtain the expression

$$\boldsymbol{df} = \left(\boldsymbol{S}^T \boldsymbol{S} \right)^{-1} \boldsymbol{S}^T \boldsymbol{\Delta f} \tag{2.25}$$

By following this procedure, $\boldsymbol{df}$ can be approximated by including more supporting nodes than derivative terms, giving the coefficient matrix better numerical condition. The more supporting nodes included, the better the condition of the matrix will become. However, the drawback is that local errors will increase simultaneously due to the enlarged local domain size d_i. Therefore, the number of supporting nodes should be kept large enough to ensure good numerical conditions for the coefficient matrix, but small enough to minimize its errors.

When the least-squares approach shown above is applied, every node contributes equally to the approximation of $\boldsymbol{df}$. In the case of derivative approximation, this is not satisfactory because nodes located near the reference should have a higher impact on the solution. By introducing a weighting function, this can easily be adjusted for. The weighted least-squares approximation is defined as

$$\boldsymbol{S}^T \boldsymbol{W} \boldsymbol{\Delta f} = \boldsymbol{S}^T \boldsymbol{W} \boldsymbol{S} \boldsymbol{df} \tag{2.26}$$

Where the weighting $\boldsymbol{W}$ is a diagonal matrix

$$\boldsymbol{W} = \text{diag} \begin{bmatrix} w_1 & w_2 & ... & w_{N-1} & w_N \end{bmatrix} \tag{2.27}$$

Each component along the diagonal in expression (2.27) represents the weighting of the respective supporting node $j \in \{1, 2, ..., N-1, N\}$. The weighting is simply a function of Euclidean distance to the reference node. Several different weighting functions have been suggested, but according to [6] the following function should be applied. A visualization is given in figure 2.4.

$$w_j = \sqrt{\frac{4}{\pi}} \left(1 - \overline{r}_j^2\right)^4 \tag{2.28}$$

$\overline{r}_j$ is the scaled distance to the supporting node

$$\overline{r}_j = \frac{\sqrt{\Delta x_j^2 + \Delta y_j^2}}{d_i} \tag{2.29}$$

where d_i is the local scaling factor presented in chapter 2.2.

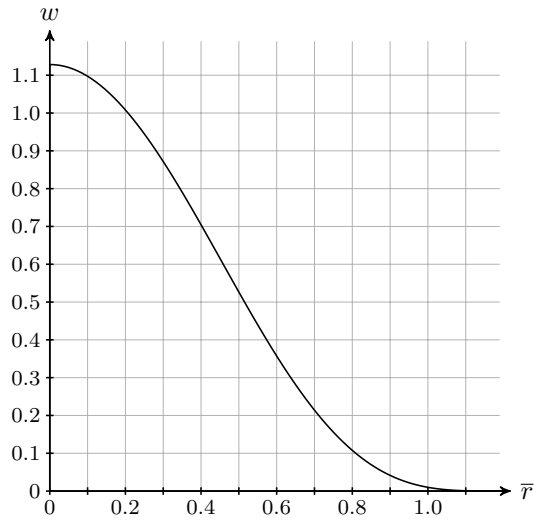


Figure 2.4: Weighting function

Now that the details of least-square-based finite difference method are derived, the final algebraic expression will be given.

$$\boldsymbol{df} = \boldsymbol{C}\boldsymbol{\Delta f} \tag{2.30}$$

$$\boldsymbol{C} = \boldsymbol{D}\left(\overline{\boldsymbol{S}}^T \boldsymbol{W} \overline{\boldsymbol{S}}\right)^{-1} \overline{\boldsymbol{S}}^T \boldsymbol{W} \tag{2.31}$$

The coefficient matrix $\boldsymbol{C}$ needs to be calculated only once, as long as the nodal positions are fixed.

2.4 Spatial discretized equations

As shown during the previous sections, all derivative terms, including the cross derivatives, are made available up to the truncated term through the LSFD coefficients. Therefore, a spatial discretization of any PDE can easily be constructed. Recall the derivative matrix form in equation (2.30). Each of the derivatives are a summation of the functional differences $(f_j - f_i)$ and their coefficients $c_{i,kj}$.

$$df_{i,k} = \sum_{j=1}^{N} c_{i,kj} \left(f_j - f_i\right) = \sum_{j=1}^{N} c_{i,kj} f_j - f_i \sum_{j=1}^{N} c_{i,kj} \tag{2.32}$$

i and j represent the reference– and supporting node, while k represents the considered derivative term. N is the total number of supporting nodes included. If the Taylor series expansion is truncated up to 2nd order, each derivative term will therefore be approximated respectively as

$$\begin{aligned}
\frac{\partial f_i}{\partial x} &= \sum_{j=1}^{N} c_{i,1j} f_j - f_i \sum_{j=1}^{N} c_{i,1j} \\
\frac{\partial f_i}{\partial y} &= \sum_{j=1}^{N} c_{i,2j} f_j - f_i \sum_{j=1}^{N} c_{i,2j} \\
\frac{\partial^2 f_i}{\partial x^2} &= \sum_{j=1}^{N} c_{i,3j} f_j - f_i \sum_{j=1}^{N} c_{i,3j} \\
\frac{\partial^2 f_i}{\partial x \partial y} &= \sum_{j=1}^{N} c_{i,4j} f_j - f_i \sum_{j=1}^{N} c_{i,4j} \\
\frac{\partial^2 f_i}{\partial y^2} &= \sum_{j=1}^{N} c_{i,5j} f_j - f_i \sum_{j=1}^{N} c_{i,5j}
\end{aligned} \tag{2.33}$$

For the Poisson equation $\nabla^2 f = g\left(x, y\right)$, the discretized LSFD scheme would give the following algebraic equation system, ready to be solved.

$$f_i = \frac{\sum_{j=1}^{N} \left(c_{i,3j} + c_{i,5j}\right) f_j - g\left(x_i, y_i\right)}{\sum_{j=1}^{N} \left(c_{i,3j} + c_{i,5j}\right)} \tag{2.34}$$

2.5 Numerical performance

First of all, the spatial order of accuracy in the LSFD method is equal to the ordinary finite difference method, and will not be degraded due to increasing number of supporting nodes [6]. However, as mention in chapter 2.3, introducing more supporting nodes will expand the local domain size and hence increase the scaled spatial differences. In other words, the truncation errors will become larger. The accuracy of the LSFD method is in fact proportional to d_i^p [25], where d_i is the local scaling factor, and p is the order of accuracy (OOA).

In order to study the numerical performance of LSFD method, the following 2D Poisson equation is solved on a square domain $(x, y) \in [(0, 0), (1, 1)]$.

$$\nabla^2 f = -2\pi^2 \sin\left(\pi x\right) \sin\left(\pi y\right) \tag{2.35}$$

The boundary conditions are $f = (1 + x)$ on all edges $\partial\Omega$. The analytical solution to the problem is

$$f = 1 + x + \sin\left(\pi x\right) \sin\left(\pi y\right) \tag{2.36}$$

Numerical performance is investigated by solving the problem using four different node distributions, all of them uniform (h=constant).

Increasing number of supporting nodes is expected to decrease the accuracy of the method by expanding the local domain size. The rate of convergence (order of accuracy) should however not be degraded. In order to investigate this, the LSFD coefficients are obtained by expanding the Taylor series to 3rd order. In theory, the 1st order derivatives should now be approximated with 3rd OOA, 2nd order derivatives with 2nd OOA, and 3rd order derivatives with 1st OOA [6]. The highest derivative terms occurring in the PDE will therefore decide the overall spatial OOA of the discretized equation system. The PDE (2.35) contains two 2nd order derivatives, hence the solution is expected to be of 2nd OOA.

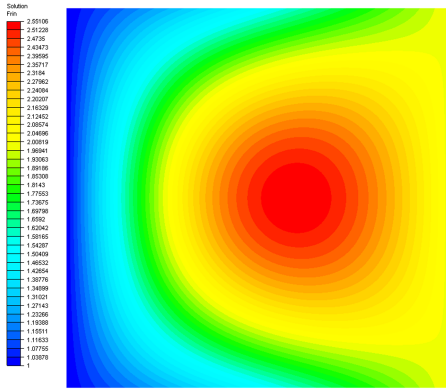


Figure 2.5: Contour plot of Poisson equation (2.35)

The errors between numerical– and exact solution are measured in the relative L_2 error norm defined as

$$L_2 = \frac{\sqrt{\sum_{i=1}^{N} \left(f_{i,num} - f_{i,exact}\right)^2}}{\sqrt{\sum_{i=1}^{N} f_{i,exact}}} \tag{2.37}$$

Results in table 2.1 are plotted in figure 2.6, from which it can be clearly seen that the rate of convergence is the same no matter the number of supporting nodes. As expected, the error increases with the number of supporting nodes.

Table 2.1: Logarithmic L_2 error norm with varying node spacing h and supporting nodes.

Node spacing h		0.1	0.05	0.025	0.0125	**OOA**
Supporting nodes	**11**	-2.237029	-2.830069	-3.426431	-4.025362	2
	16	-2.140827	-2.732095	-3.329643	-3.928001	2
	21	-1.921275	-2.514601	-3.111741	-3.711016	2
	26	-1.849553	-2.443697	-3.040754	-3.639906	2

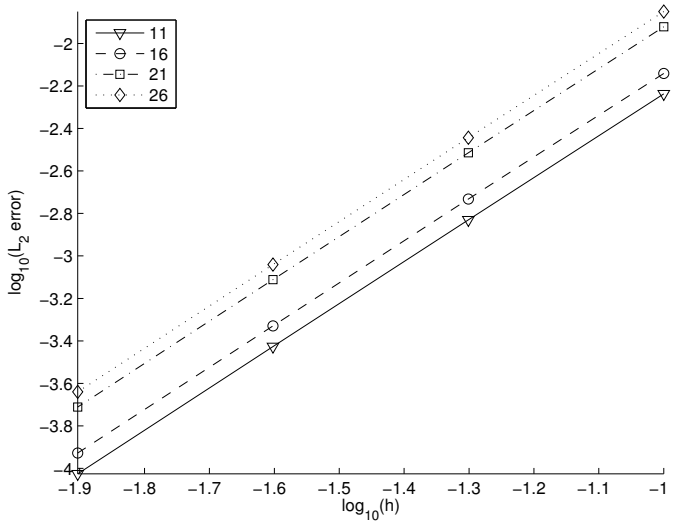


Figure 2.6: Convergence plot for varying number of supporting nodes

The rate of convergence can be calculated using a best fit of the curves in figure 2.6, or simply by a linear approximation.

$$\log_{10}(err) = a \log_{10} h + b \tag{2.38}$$

err is the L_2 error norm and a is the rate of convergence. Using the results from table 2.1, a convergence rate of ~ 2 is hence obtained, supporting the theoretical OOA.

In theory, if the Taylor series expansion is truncated up to 2nd order, expected OOA is 1. Likewise, if truncated up to 4th– and 5th order, OOA is respectively 3– and 4. To investigate this, a convergence study is performed using Taylor series expansions up to the respective orders.

Table 2.2: Logarithmic L_2 error norm with varying node spacing h and truncated terms.

Node spacing h		0.1	0.05	0.025	0.0125	**OOA**
Truncated terms	**2**	-2.699964	-3.290763	-3.886573	-4.485465	2
	3	-2.237029	-2.830069	-3.426431	-4.025362	2
	4	-4.059690	-5.021991	-6.185197	-7.232897	4
	5	-3.351632	-4.569932	-5.770477	-6.986294	4

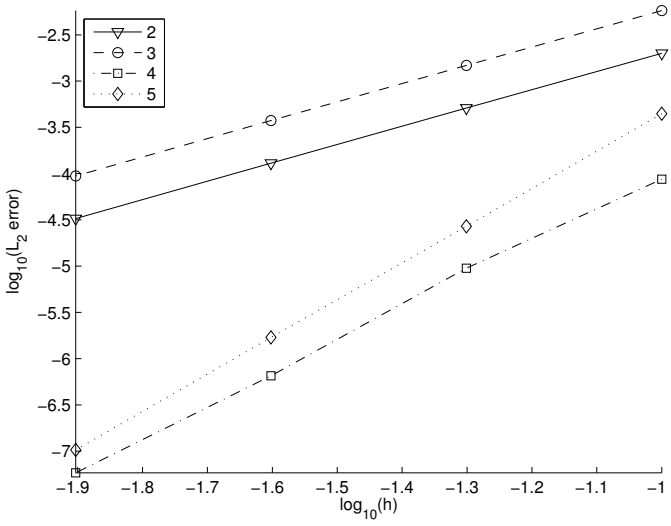


Figure 2.7: Convergence plot for varying number of truncated terms

5th order truncation yields the theoretical accuracy of 4th order. When truncated to the 2nd– and 4th order, the rate of convergence proves to be respectively ~ 2– and ~ 3.6, providing one OOA higher than expected. This phenomena has also been pointed out in [10], and can be explained by the fact that the applied node spacing h is uniform. Recall the 2nd order central difference equation (2.12), where 2nd OOA was achieved due to the uniform spacing.

For demonstration purpose, one dimensional Taylor series expansion is applied for conventional FDM.

$$\boldsymbol{\Delta f} - \boldsymbol{\epsilon} = \boldsymbol{A df} \tag{2.39}$$

$\boldsymbol{A}$ is a square matrix containing the Taylor coefficients and $\boldsymbol{\epsilon}$ is the truncation error. Inverting $\boldsymbol{A}$ reveals the vector of discretized derivatives.

$$\boldsymbol{df} = \boldsymbol{A}^{-1} \left(\boldsymbol{\Delta f} - \boldsymbol{\epsilon} \right) \tag{2.40}$$

$\boldsymbol{df}$ truncated to 2nd order, assuming constant Δx, corresponds to the following coefficient matrix.

$$\boldsymbol{A} = \begin{bmatrix} \Delta x & \frac{1}{2}\Delta x^2 \\ -\Delta x & \frac{1}{2}\Delta x^2 \end{bmatrix} \tag{2.41}$$

Inverting $\boldsymbol{A}$ and substituting into equation (2.40) yields

$$\begin{bmatrix} f_x \\ f_{xx} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\Delta x^{-1} & -\frac{1}{2}\Delta x^{-1} \\ \Delta x^{-2} & \Delta x^{-2} \end{bmatrix} \left(\begin{bmatrix} f_1 - f_i \\ f_2 - f_i \end{bmatrix} - \begin{bmatrix} O\left(\Delta x^3\right) \\ O\left(\Delta x^3\right) \end{bmatrix} \right) \tag{2.42}$$

Following the same procedure, though truncating equation (2.40) to 3rd order, gives

$$\begin{bmatrix} f_x \\ f_{xx} \\ f_{xxx} \end{bmatrix} = \begin{bmatrix} \Delta x^{-1} & -\frac{1}{3}\Delta x^{-1} & \frac{1}{6}\Delta x^{-1} \\ \Delta x^{-2} & \Delta x^{-2} & 0 \\ -3\Delta x^{-3} & -\Delta x^{-3} & \Delta x^{-3} \end{bmatrix} \left(\begin{bmatrix} f_1 - f_i \\ f_2 - f_i \\ f_3 - f_i \end{bmatrix} - \begin{bmatrix} O\left(\Delta x^4\right) \\ O\left(\Delta x^4\right) \\ O\left(\Delta x^4\right) \end{bmatrix} \right) \tag{2.43}$$

Comparing equation (2.42) and (2.43) reveals identical approximation of the 2nd order derivative term, except that higher order of accuracy is achieved for (2.43). If 2nd order approximation of the 2nd order derivative term is required, Taylor series truncated to 2nd order is hence sufficient for the uniform node spacing. The reason why 3rd order Taylor series expansion in figure 2.7 indicates larger error is because more supporting nodes are used in order to prevent an ill-conditioned coefficient matrix.

The same can be observed when equation (2.40) is truncated to 4th– and 5th order. Actually, all even derivative terms will achieve one OOA higher than expected when equation (2.40) is truncated up to an even term[1]. The LSFD scheme is basically of central difference form since the supporting nodes are chosen entirely based on their Euclidean distance to the reference node. Therefore, this behavior also applies to LSFD method.

[1] Valid only for central difference approximations with uniform node spacing

Chapter 3

Computational fluid dynamics

The PDEs governing fluid flow were first derived by Claude-Louis Navier and Sir George Gabriel Stokes for more than a century ago. The Navier-Stokes equations are mathematically very complex, and in most cases impossible to solve analytically. However, by discretizing the equations in space and time, they may be solved numerically.

3.1 Governing equations

The unknown variables in a general three dimensional fluid dynamics problem are mainly the velocity components u,v and w, the pressure p and the temperature T. In order to solve for these unknown variables, five equations are hence needed. Through the laws of conservation, these equations can be obtained. Note that the expressions in this chapter are given in Einstein summation notation, where index i and j takes the values 1,2 and 3 (representing the dimensions of Euclidean space). If either i or j appear more than once in a single term, as below, a summation is present.

$$\frac{\partial u_i}{\partial x_i} = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} \tag{3.1}$$

where

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \text{ and } \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.2}$$

3.1.1 Conservation of mass

Conservation of mass is a fundamental requirement which leads to the continuity equation. Total mass $m = \rho V$ have to be maintained at all time.

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0 \tag{3.3}$$

Equation (3.3) is the general continuity equation, where the density ρ is a function of pressure and temperature. By requiring zero velocity divergence

$$\frac{\partial u_j}{\partial x_j} = 0 \tag{3.4}$$

the flow is considered incompressible. This condition is fulfilled when the density is considered constant, which is the case for true incompressible fluids– or compressible fluids at Mach number< 0.3.

The entire derivation of continuity equation (3.3) can be found in [31].

3.1.2 Conservation of momentum

Conservation of momentum, based on Newton's 2nd law, is widely known as the Navier-Stokes equations which preserves the total momentum in a system at all time.

$$\frac{\partial \rho u_i}{\partial t} + u_j \frac{\partial \rho u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \sigma'_{ij}}{\partial x_j} + \rho f_i \tag{3.5}$$

σ'_{ij} is the viscous stress term

$$\sigma'_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \frac{\partial u_j}{\partial x_j} \right) \tag{3.6}$$

μ is the dynamic viscosity, which for Newtonian fluids is a function of temperature and pressure. f_i denotes a source term, often the gravitational acceleration. δ_{ij} is the Kronecker delta, defined as

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{3.7}$$

When incompressible flow and constant viscosity are considered, the NS-equations will reduce to

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j^2} + f_i \tag{3.8}$$

where ν is the kinematic viscosity defined as

$$\nu = \frac{\mu}{\rho} \tag{3.9}$$

The entire derivation of equation (3.5) can be found in [31].

3.1.3 Conservation of energy

Conservation of energy is based on the first law of thermodynamics and will maintain the energy balance in a system at all time.

$$\frac{\partial \rho e}{\partial t} + u_j \frac{\partial \rho e}{\partial x_j} = \frac{\partial}{\partial x_j}\left(k \frac{\partial T}{\partial x_j}\right) + \sigma'_{ij}\frac{\partial u_i}{\partial x_j} \tag{3.10}$$

e is the internal energy, and k is the thermal conductivity, which is a function of pressure and temperature. In the case of incompressible flow and constant thermal conductivity, equation (3.10) will reduce to

$$\frac{\partial T}{\partial t} + u_j \frac{\partial T}{\partial x_j} = \frac{k}{\rho c_p}\frac{\partial^2 T}{\partial x_j^2} \tag{3.11}$$

where c_p is the specific heat. An important feature with incompressible assumption is that the coupling between the conservation of energy and conservation of mass– and momentum disappears.

The complete derivation of (3.10) can be found in [31].

3.1.4 Two dimensional incompressible isothermal flow

By reducing the problem by one dimension, one of the velocity components will vanish with its respective momentum equation completely. Incompressible flow is assumed, uncoupling the temperature from velocity and pressure, and temperature is considered constant, eliminating the energy equation. By neglecting gravity, or other source terms for that sake, the complete set of equations treated throughout this thesis are hence obtained.

$$\text{x-momentum:} \qquad \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{3.12}$$

$$\text{y-momentum:} \qquad \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{3.13}$$

$$\text{continuity:} \qquad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3.14}$$

Equation (3.12) and (3.13) contains a convective– and diffusive term, which both are of great interest. The non-linear convective term represents an acceleration of velocity as a function of spatial position. In case of viscous dominating Stokes flows, $Re \ll 1$, the convective term will become small compared to the viscous stress, and may be neglected.

$$\frac{\partial u_i}{\partial t} + \underbrace{u_j\frac{\partial u_i}{\partial x_j}}_{\text{convective}} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \underbrace{\nu\frac{\partial^2 u_i}{\partial x_j^2}}_{\text{diffusive}} \tag{3.15}$$

In steady-state problems, the time dependent term vanish. However, when analyzing a steady-state problem numerically, the term cannot usually be removed. Time marching will have to be performed until the solution does no longer change with time.

3.2 Solution methods

In chapter 2, the spatial discretization of the LSFD method has been explained in detail. The temporal (time) discretization shall be treated in this section. As with space derivatives, time derivatives may also be approximated with Taylor series expansion. There are generally two main categories for which time discretization can be divided into, explicit– and implicit. When the time derivatives are treated explicitly, functional values at $(k+1)\,\Delta t$ are found using the solution at the previous time level $k\Delta t$. Implicit treatment of the time derivatives means that the solution are found by iteration at time level $(k+1)\,\Delta t$.

Equation (3.12), (3.13) and (3.14) all contain the velocity components. The pressure term however, is only appearing in the momentum equations. The lack of an independent pressure equation invokes difficulties when solving the governing equations numerically. This issue is treated differently in the following methods.

3.2.1 Direct methods

As the name implies, direct methods solve the governing equations directly. However, an individual pressure equation is also introduced by taking the divergence of the momentum equations (3.12) and (3.13).

$$\frac{\partial}{\partial x_i}\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu\frac{\partial^2 u_i}{\partial x_j^2}\right) \tag{3.16}$$

By imposing the continuity restriction on equation (3.16) and noting that

$$\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + 2\frac{\partial u}{\partial x}\frac{\partial v}{\partial y} = 0 \tag{3.17}$$

The complete expression for the pressure is thus obtained.

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = 2\rho\left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial y} - \frac{\partial u}{\partial y}\frac{\partial v}{\partial x}\right) \tag{3.18}$$

The continuity requirement is included in equation (3.18), and instead of solving the individual continuity equation (3.14), the new pressure-Poisson equation (3.18) will be solved in its place. Even though the solution procedure is pretty straight forward, direct methods are usually computationally expensive, especially for three dimensional problems [23].

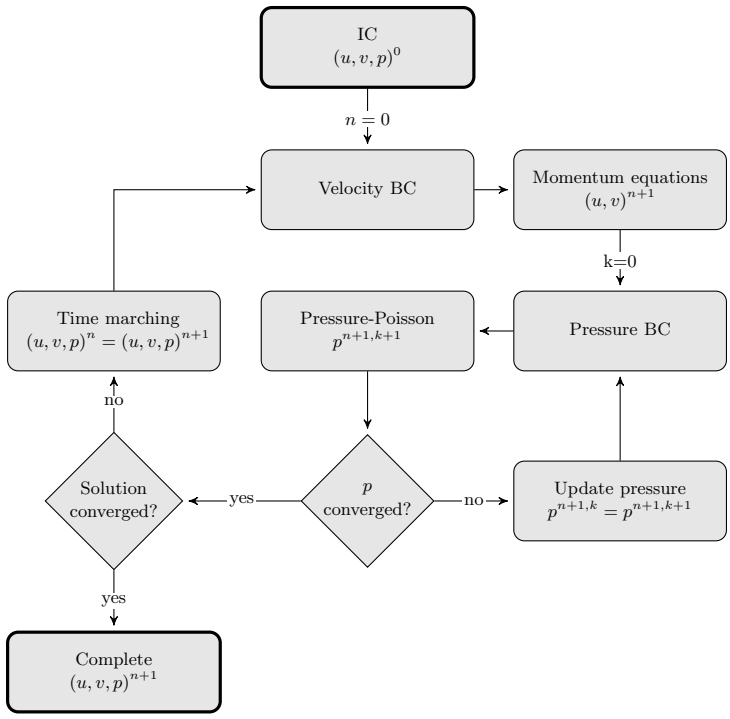


Figure 3.1: Explicit direct method algorithm

3.2.2 Vorticity-stream function method

The vorticity-stream function approach are one of the earliest methods used to solve two dimensional fluid problems numerically. This procedure was actually invented before the computer and solved entirely by hand [31]. The method does not aim to solve the original NS-equation, but instead the vorticity transport equation, which can be derived directly from NS.

Two dimensional vorticity ζ is defined as the velocity curl in equation (3.19), and can be interpreted as the rotational ability in a flow.

$$\zeta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{3.19}$$

The stream function ψ is defined in a way such that the velocity components are partial derivatives of the stream function itself.

$$\begin{aligned} u &= \frac{\partial \psi}{\partial y} \\ -v &= \frac{\partial \psi}{\partial x} \end{aligned} \tag{3.20}$$

By taking the curl of the momentum equations (3.12) and (3.13)

$$\frac{\partial}{\partial x}\,(\text{y-momentum}) - \frac{\partial}{\partial y}\,(\text{x-momentum}) \tag{3.21}$$

and using the definition of vorticity ζ and stream function ψ, the vorticity transport equation is thus obtained.

$$\frac{\partial \zeta}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial \zeta}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \zeta}{\partial y} = \nu \left(\frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2} \right) \tag{3.22}$$

Substituting the definition of stream function in equation (3.20) into equation (3.19), the ψ-Poisson equation, linking vorticity and stream function is hence achieved.

$$-\zeta = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} \tag{3.23}$$

Equation (3.22) and (3.23) together make up the governing equations in the vorticity-stream function approach. During derivation of the vorticity transport equation (3.22), pressure was completely eliminated, reducing the dependent variables to vorticity and stream function only. The velocities can be obtained by solving equation (3.20) whenever needed. Since the problem is reduced to two equation instead of the original three, the procedure is generally very computational efficient.

If the pressure solution is required as well, it can be obtained by using a similar approach as for the direct methods.

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = 2\rho \left(\frac{\partial^2 \psi}{\partial x^2}\frac{\partial^2 \psi}{\partial y^2} - \left(\frac{\partial^2 \psi}{\partial x \partial y} \right)^2 \right) \tag{3.24}$$

If both pressure and velocity components are required within each time step, two individual Poisson equations then have to be solved, and the vorticity-stream function approach does no longer have any obvious advantages to the other solution

methods. Another counterargument against the use of vorticity-stream function approach is the difficulties regarding boundary conditions. Besides, since the stream function only exists in two dimensional flows, the method is hence only valid for two dimensions.

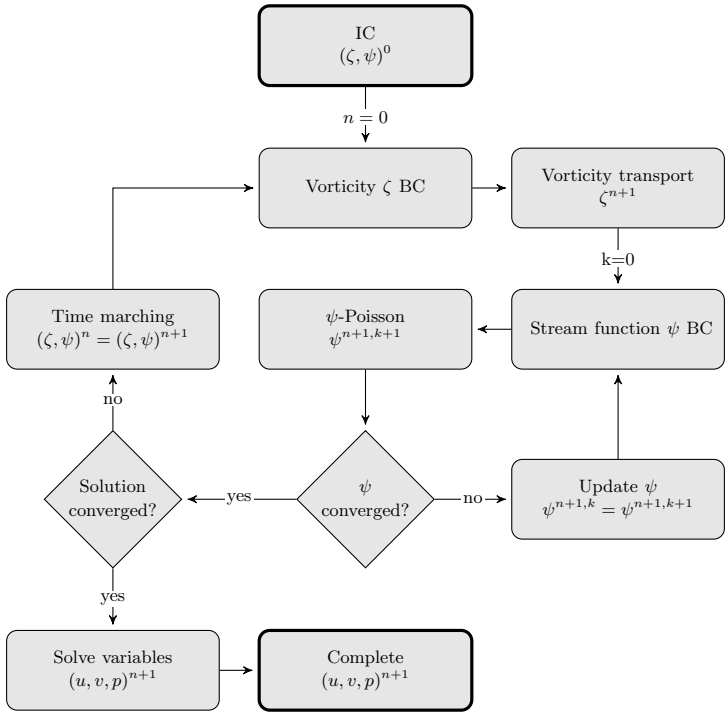


Figure 3.2: Explicit vorticity-stream function algorithm

3.2.3 Projection methods

The original projection method, initially developed by Chorin [4], is based on the Helmholtz-Hodge decomposition theorem [5]. This theorem claims that any vector field in a bounded domain with a smooth boundary may be decomposed into the sum of a divergence free– and an irrotational part. Projection methods are used in order to solve incompressible fluid flow problems in an efficient manner.

First, the intermediate velocities $u_i^{\overline{n+1}}$ are calculated using equation (3.12) and (3.13), though without the pressure terms. The viscous terms should be treated implicitly in order to avoid instabilities due to viscous effects.

$$\frac{\overline{u_i^{n+1}} - u_i^n}{\Delta t} = -u_j^n \frac{\partial u_i^n}{\partial x_j} + \nu \frac{\partial^2 \overline{u_i^{n+1}}}{\partial x_j^2} \tag{3.25}$$

The intermediate velocity fields does not generally satisfy the continuity restriction. In the next step (the projection), the objective is to force continuity onto the next velocity field u_i^{n+1}.

$$\frac{u_i^{n+1} - \overline{u_i^{n+1}}}{\Delta t} = -\frac{1}{\rho} \frac{\partial p^{n+1}}{\partial x_i} \tag{3.26}$$

Taking the divergence of equation (3.26), and requiring continuity for the next velocity time step, $\nabla \boldsymbol{u}^{n+1} = 0$, the pressure-Poisson equation is obtained.

$$\frac{\partial^2 p^{n+1}}{\partial x_j^2} = \frac{\rho}{\Delta t} \frac{\partial \overline{u_i^{n+1}}}{\partial x_i} \tag{3.27}$$

The boundary conditions for p^{n+1} can be deduced directly from equation (3.26). Boundary conditions for the intermediate– and next velocity step can be assumed equal. As the impermeable wall requires zero normal velocity at the wall, the pressure gradient normal to the wall will have to be zero as well.

$$\frac{\partial p^{n+1}}{\partial \eta} = 0 \tag{3.28}$$

The pressure solution p^{n+1} is found by iteration, and equation (3.26) can once again be applied to finally achieve the velocity components at time step $n + 1$.

$$u_i^{n+1} = \overline{u_i^{n+1}} - \frac{\Delta t}{\rho} \frac{\partial p^{n+1}}{\partial x_i} \tag{3.29}$$

The projection methods have gained huge attention due to their computational efficiency, especially for large scale problems [18]. They are valid also for three dimensional problems, and boundary condition can easily be determined[1].

[1] Valid only for 1st order methods. The boundary conditions for 2nd order methods are more complicated.

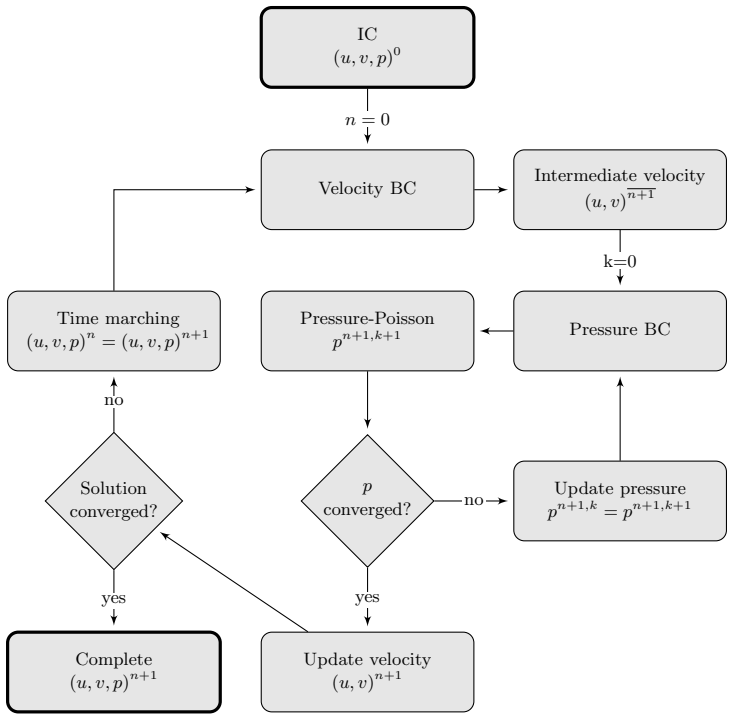


Figure 3.3: Semi-implicit projection method routine

3.3 Initial– and boundary conditions

In order to solve any PDE, boundary conditions are required on all edges (2D)– or surfaces (3D) that defines the domain boundaries $\partial\Omega$. In addition, if the PDE is time dependent, which is the case for Navier-Stokes equations, also an initial solution at time $t = 0$ will have to be defined. It is however quite common to set this initial condition to zero for all variables and perform time marching in order to achieve the final solution. Boundary conditions have to be treated carefully since they will affect the solution in the entire domain. The boundary conditions may be of Dirichlet– Neumann– or Periodic type.

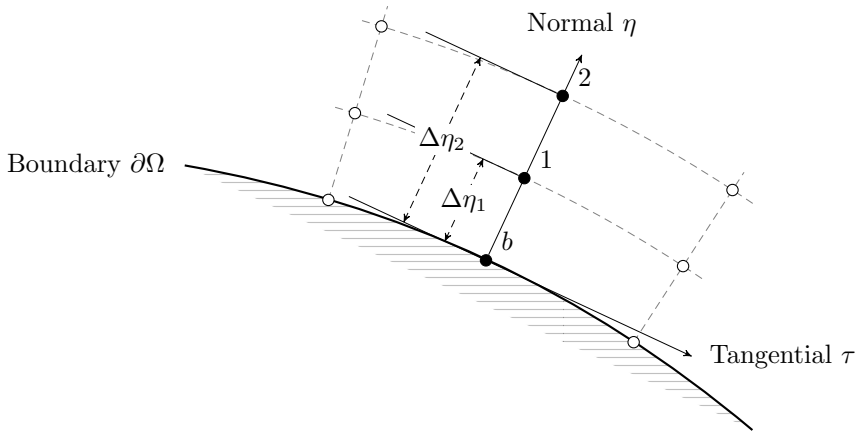


Figure 3.4: Boundary illustration

The *Dirichlet* boundary condition is the easiest condition to implement. For this condition type, the functional value f_b at the boundary in figure 3.4 is known, making it possible to determine f_1, f_2 etc.

The *Neumann* boundary condition is a derivative condition, for which the functional value f_b is not known, but its derivative term (usually the normal derivative) is. It is therefore possible to construct an algebraic expression for f_b, with respect to the functional values at its neighbor nodes. If for example the normal derivative at node b is known and $\Delta\eta_1 = \frac{1}{2}\Delta\eta_2 = \Delta\eta$, functional values at node 1 and 2 may be used to express f_b as following.

$$f_1 = f_b + \Delta\eta \frac{\partial f_b}{\partial \eta} + \frac{\Delta\eta^2}{2} \frac{\partial^2 f_b}{\partial \eta^2} + O\left(\Delta\eta^3\right) \tag{3.30}$$

$$f_2 = f_b + 2\Delta\eta \frac{\partial f_b}{\partial \eta} + 2\Delta\eta^2 \frac{\partial^2 f_b}{\partial \eta^2} + O\left(\Delta\eta^3\right) \tag{3.31}$$

By substituting equation (3.30) into (3.31) the final expression for f_b is obtained.

$$f_b = \frac{4f_1 - f_2}{3} - \frac{2\Delta\eta}{3} \frac{\partial f_b}{\partial \eta} + O\left(\Delta\eta^3\right) \tag{3.32}$$

Expression (3.32) can be used explicitly at the boundary nodes, while the internal nodes are treated as usual.

The *Periodic* boundary condition uses the solution at one boundary as the condition for another. This condition type will however not be discussed any further, nor used in this thesis.

If Δx and Δy are known, then $\Delta\tau$ and $\Delta\eta$ can be expressed in terms of the following relation [27].

$$\begin{bmatrix}\Delta\tau\\ \Delta\eta\end{bmatrix} = \begin{bmatrix}\boldsymbol{\tau}\\ \boldsymbol{\eta}\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} \tag{3.33}$$

$\boldsymbol{\eta}$ are the normal vectors $\begin{bmatrix}\eta_1 & \eta_2\end{bmatrix}$, while $\boldsymbol{\tau}$ are the tangential vectors $\begin{bmatrix}\tau_1 & \tau_2\end{bmatrix}$. The normal– and tangential vectors are related through a third vector $\boldsymbol{k}$, defined to be orthogonal to– and pointing into the xy- plane.

$$\boldsymbol{\eta} = \boldsymbol{\tau} \times \boldsymbol{k} \tag{3.34}$$

Solving expression (3.34) with $\boldsymbol{k} = i(0) + j(0) + k(-1)$ gives the direct relation between τ and η.

$$\begin{bmatrix}\eta_1\\ \eta_2\end{bmatrix} = \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} \tag{3.35}$$

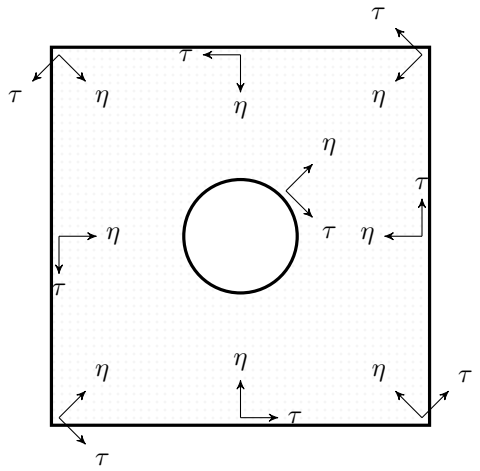


Figure 3.5: Boundary coordinate system

The exact same procedure can be applied in order to transform velocity components between the Cartesian– and local orthogonal coordinate system.

$$\begin{bmatrix}U_\tau\\ U_\eta\end{bmatrix} = \begin{bmatrix}\boldsymbol{\tau}\\ \boldsymbol{\eta}\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix} \tag{3.36}$$

3.3.1 No-slip wall

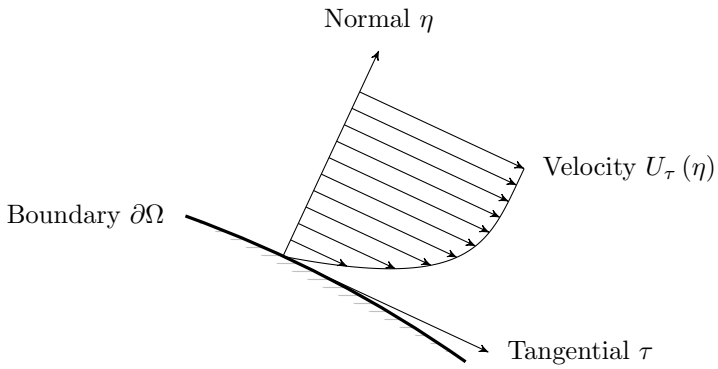


Figure 3.6: No-slip boundary condition

The wall is a boundary, for which no mass flow are allowed to cross. In other words, the velocity normal to the wall U_η, have to be zero relative to the normal velocity of the wall itself. This condition is universal, and is valid for both potential– and viscous flow theory. In viscous flows however, another restriction does also apply. The no-slip wall condition require the tangential fluid velocity U_τ to be zero relative to the tangential velocity of the wall itself. The no-slip boundary condition hence implies two individual Dirichlet conditions for the velocity.

$$\begin{bmatrix} U_\tau \\ U_\eta \end{bmatrix} = \begin{bmatrix} U_{\tau,wall} \\ U_{\eta,wall} \end{bmatrix} \tag{3.37}$$

From chapter 3.2.3, the wall boundary condition for the pressure have been given by the Neumann condition

$$\frac{\partial p}{\partial \eta} = 0 \tag{3.38}$$

3.3.2 Free-slip wall

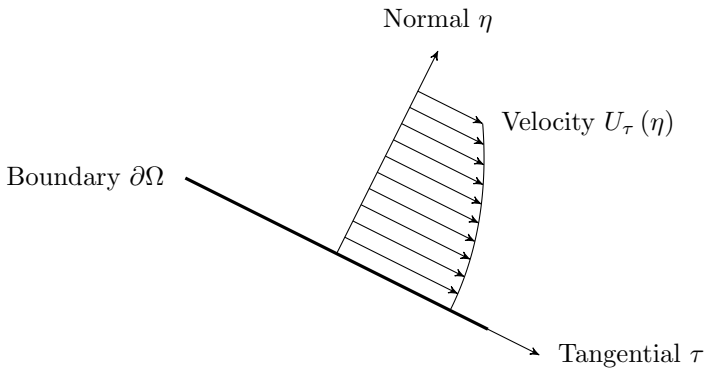


Figure 3.7: Free-slip boundary condition

Another important wall condition, the free-slip wall, is used to model both symmetry and free stream flow. In this boundary condition, the no-penetrating condition $U_\eta = 0$ applies together with a zero normal velocity gradient at the free-slip wall.

$$\begin{bmatrix} \dfrac{\partial U_\tau}{\partial \eta} \\ U_\eta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.39}$$

The pressure condition is, like for the no-slip boundary

$$\frac{\partial p}{\partial \eta} = 0 \tag{3.40}$$

3.3.3 Inlet

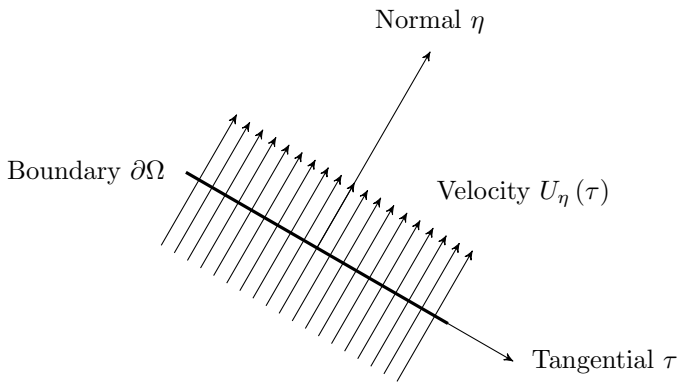


Figure 3.8: Inlet velocity boundary condition

At inlets, the velocity distribution is often considered as known. In these cases, the pressure at the boundary can be approximated as

$$\frac{\partial p}{\partial \eta} = 0 \tag{3.41}$$

By fixing the inlet velocity, it is guaranteed that no back flow will occur at the boundary. This type of boundary condition is hence restrictive, though stable.

3.3.4 Open boundary

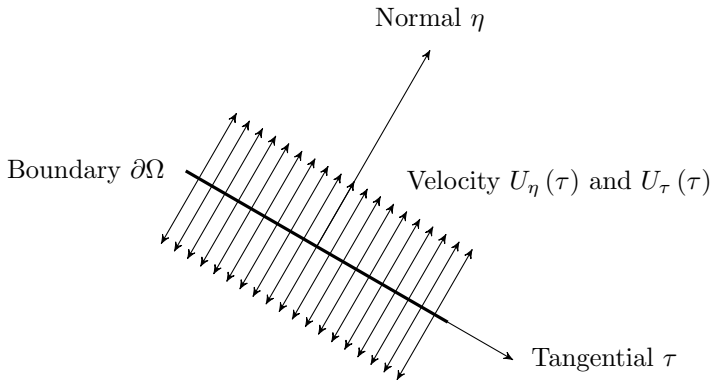


Figure 3.9: Open boundary condition

The open boundary condition is mainly used at outlet boundaries, where the velocity distribution is seldom known, but can also be applied to inlets. By fixing the pressure, velocity is allowed to develop, making this boundary condition less restrictive. There are two ways of defining the velocity condition, either by fixing the tangential velocity U_τ, or by fixing its normal gradient.

$$\begin{bmatrix} U_\tau \\ \dfrac{\partial U_\eta}{\partial \eta} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} \dfrac{\partial U_\tau}{\partial \eta} \\ \dfrac{\partial U_\eta}{\partial \eta} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.42}$$

The first option in equation (3.42) is more restrictive than the other since the tangential velocity is denied to develop. Making the boundary conditions restrictive may increase errors if it is located too near flow complexities. On the other hand, less restrictive conditions may introduce instabilities, which could lead to divergence.

3.4 Stability and convergence

In order to achieve the correct solution to a problem, the method have to be consistent and stable. Consistency means that the approximated solution will tend to the exact one as the node spacing h tends to zero. Stability is a subject only for temporal (time dependent) schemes which are said to be stable if its errors does not grow with time. In the discretized NS equations there are basically two terms, the convective– and the diffusive term, for which stability have to be ensured. The stability restrictions follow the applied scheme, not the PDE. In order for a discretization scheme to be useful it has to be at least conditionally stable, though unconditional stability is usually preferred. Unconditional stability comes with implicit schemes where the solution at time level $n+1$ is approximated by iteration on time level $n+1$. The conditionally stable schemes are explicit which means that the solution at time level $n+1$ is obtained directly from the solution at time level n. Even though the implicit schemes are unconditionally stable, they will usually be more computational expensive than the explicit schemes. The convective stability restriction in explicit schemes is called the CFL condition, and requires the *Courant number* to be less than 1. The Courant number is defined as

$$c = \frac{u\Delta t}{\Delta h} \tag{3.43}$$

where u is the velocity, Δt is the time step, and Δh is the spatial difference. Since the velocity is a function of the actual problem and Δh is given by the

node distribution, the only adjustable parameter is the time step. If the temporal discretization is implicit, no such restriction applies.

The term convergence is used in two different senses. The Lax equivalence theorem claims that if a method is both consistent and stable, it will converge to the exact solution as Δh and Δt tends to zero. The rate of convergence is therefore a measure of how fast a method converges to the exact solution. The spatial rate of convergence for the LSFD method have already been studied in chapter 2.5.

Iterative convergence is the other sense for which the term is used. Iterative methods are used in order to solve equation systems with a repetitive procedure until some given convergence criteria is satisfied. At the end of each iteration step, the residuals are compared to the convergence tolerance ϵ_{tol}.

$$\max\left(f_{ij}^{n+1} - f_{ij}^{n}\right) \leq \epsilon_{tol} \tag{3.44}$$

Equation (3.44) is used to determine wether the iteration procedure should continue or terminate. Iterative methods are therefore able to approximate the solution to a chosen tolerance.

Chapter 4

Program

This thesis has been based on the development of a meshfree method for solving two dimensional incompressible fluid flows. Most of the effort have therefore naturally been related directly to the programming which have been done in the C language. The generated *.vtf result file can be post-processed directly in Ceetron GLview. In this chapter, the developed code will be presented in detail. The flow chart can be found in appendix A, and the source code is included electronically.

4.1 Program input

In order to solve any fluid flow problem the physical geometry will have to be defined together with the boundary conditions, fluid properties and numerical solver properties. The geometry is constructed using the 2D meshing software MEGA (developed by Håvard Holm), and can be imported directly for analysis. The *.vtf geometry file exported from MEGA contains information on node coordinates, elements, and boundary conditions. Numerical– and fluid properties are specified in an individual input text file.

Table 4.1: Properties included in the input text file.

Fluid properties	**Numerical properties**	**Boundary conditions**
Density	Number of auxiliary nodes	No-slip
Dynamic viscosity	Number of truncated terms	Free-slip
	Residual error	Velocity
	Relaxation factor	Pressure
	Pressure steps	
	Timestep	
	Time duration	
	Courant number	
	Output interval	

4.2 Boundary conditions

When the geometry is constructed in MEGA, each boundary edge will be given an identification number. Edges that later will share common boundary conditions may be identified with equal numbers. This is however optional. In the input text file, boundary conditions will have to be assigned to each identification number for the problem to be fully defined. Boundary conditions are chosen based on the four different options listed in table 4.1.

No-slip: This boundary condition will model a solid wall with tangential velocity. "`noslip 0`" denotes a stationary wall, while "`noslip 0.5`" denotes a wall with velocity 0.5 $\left[\mathrm{m} \cdot \mathrm{s}^{-1}\right]$ in the tangential direction defined in figure 3.5.

Free-slip: The free-slip boundary condition can be used to model a symmetry– or free stream edge. The condition is applied with the command "`freeslip`"

Velocity: The velocity boundary condition is used to model an inlet with uniform normal velocity distribution. This condition does not allow back flow and is therefore more robust than the pressure boundary condition. "`velocity 0.5`" denotes a uniform velocity 0.5 $\left[\mathrm{m} \cdot \mathrm{s}^{-1}\right]$ in the normal direction defined in figure 3.5.

Pressure: The open boundary pressure condition can be used to model both inlets and outlets, but is less robust for inlets than the velocity condition above. The pressure condition is therefore mainly applied to outlet boundaries where the velocity distribution is unknown. "`pressure 0.5`" denotes a gauge pressure of 0.5 [Pa].

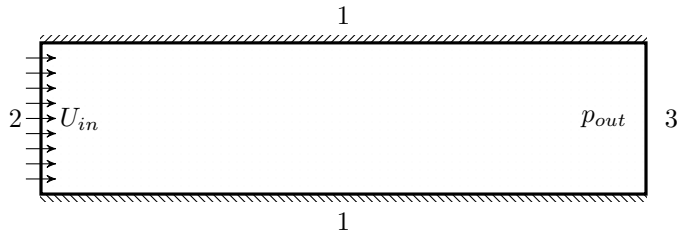


Figure 4.1: Boundary identities for a rectangular channel flow

In order to model a channel flow with inlet velocity $U_{in} = 1\,[\mathrm{m}\cdot\mathrm{s}^{-1}]$ and outlet gauge pressure $p_{out} = 0\,[\mathrm{Pa}]$ using the geometry in figure 4.1, the boundary identification numbers will have to be linked to their respective boundary conditions as following.

```
1  noslip    0
2  velocity  1
3  pressure  0
```

Since the edges are connected at the ends, corner nodes will have two possibly different identification numbers. A singularity will arise if two conflicting boundary conditions apply to the same corner node. One way to deal with this is to make some boundary conditions systematically superior to the others by ranking. When a node with two identities is then considered, the program will choose the identity referring to the highest ranked boundary condition of the two.

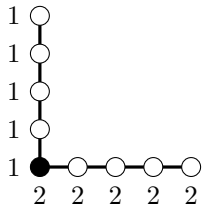


Figure 4.2: Boundary identities at corner node

The corner is a geometrical discontinuity where the normal vector does not formally exist. To deal with this issue, a normal vector is defined as a unit vector passing through the corner node and the geometrical center of a circle, tangential to the coinciding boundary edges as seen in figure 4.3.

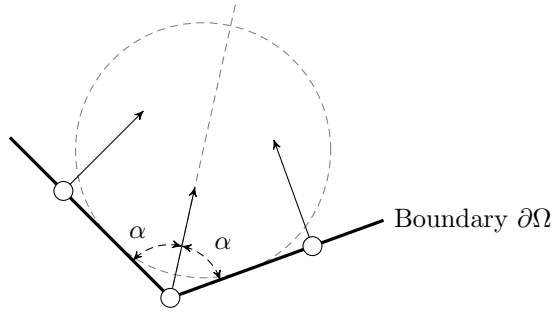


Figure 4.3: Normal direction at a discontinuous corner

The boundary conditions will become more or less unphysical at the corner nodes, and should hence be ranked in a way that minimize their unphysical effects. As an example, the inlet velocity in figure 4.1 should not be defined at the corners because the definition of the normal vector at discontinuities (figure 4.3) will induce a vertical velocity component. Instead, the no-slip condition should be applied and therefore also ranked higher than velocity. The ranking system used for the code developed in this thesis is showed in table 4.2.

Table 4.2: Boundary condition ranking

Boundary condition	**Rank**
No-slip $u_\tau = 0 \left[\mathrm{m \cdot s^{-1}}\right]$	5
No-slip $u_\tau \neq 0 \left[\mathrm{m \cdot s^{-1}}\right]$	4
Pressure	3
Free-slip	2
Velocity	1

4.3 Neighbor node identification

As explained in chapter 2, every node in the domain comprises its own subdomain with N number of neighbor/auxiliary/supporting nodes. In this program, the neighbor nodes need to be settled only once since the node coordinates are fixed throughout the calculation. The objective is therefore, for every single node in the domain, to find its adjacent nodes in terms of Euclidean distance. The easiest way to do this would be to simply calculate the distance to every node in the domain, and sort their distances in order to decide which of them are located closest to the reference. Even though the algorithm is easy to follow, it is extremely computational expensive as the number of calculations needed are proportional to the total number of nodes squared. This method would quickly end up as the

bottleneck, and therefore a better procedure for deciding the neighbors will have to be used.

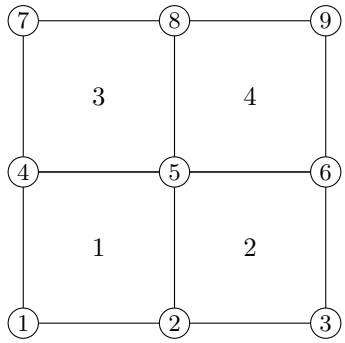


Figure 4.4: Node connection

Even though the meshfree method does not in general need a mesh of connected nodes, MEGA will provide one. This information can be used to identify the neighbor nodes. Figure 4.4 shows an element mesh defined by a set of nodes. The nodes assigned to each element are tabulated below.

Table 4.3: Nodes per element

Element	**Node**			
	1	**2**	**3**	**4**
1	1	2	5	4
2	2	3	6	5
3	4	5	8	7
4	5	6	9	8

From the information in table 4.3, a new table is constructed, stating the elements connected to each node.

Table 4.4: Elements per node

Node	**Element**			
	1	**2**	**3**	**4**
1	1	-	-	-
2	2	1	-	-
3	2	-	-	-
4	1	3	-	-
5	1	2	4	3
6	4	2	-	-
7	3	-	-	-
8	3	4	-	-
9	4	-	-	-

The information in table 4.3 and 4.4 can be utilized in order to decide the neighbor nodes for each reference node using the following algorithm.

```
for reference=1 → total number of nodes do
   status(all nodes) ← "false"
   status(node(reference)) ← "true"
   lower ← 1
   upper ← 1
   counter ← 1
   neighbor(counter) ← node(reference)
   while counter < required number of neighbor nodes do
      for i=lower → upper do
         temporary node ←neighbor(i)
         for j=1 → number of elements connected to temporary node do
            temporary element ← element(j) of temporary node
            for k=1 → number of nodes connected to temporary element do
               if status(node(k) of temporary element) = "false"  then
                  status(node(k) of temporary element) ← "true"
                  counter ← counter+1
                  neighbor(counter) ← node(k) of temporary element
               end if
            end for
         end for
      end for
      lower ← upper
      upper ← counter
   end while
   calculate distance between node(reference) and the neighbor nodes.
   sort the neighbor nodes with respect to difference and save.
end for
```

By following this procedure, the neighbor nodes can rapidly be identified. In fact, the number of calculations needed is proportional to the total number of nodes only. The neighbor nodes are decided this way for both internal– and boundary

nodes.

4.4 LSFD coefficients

Now that the neighbor nodes have been decided, the LSFD coefficients derived in chapter 2 can be solved for every node in the domain. The internal coefficients are solved with respect to the Cartesian coordinate system for a given number of truncated terms and neighbor nodes. The boundary coefficients are found using the same procedure as for the internal nodes, but in terms of a local orthogonal coordinate system where the unit normal vector points into the fluid domain, and the unit vector normal to the xy-plane points in negative z-direction as shown in figure 3.5. The boundary coefficients are truncated to 1st order, using 3 neighbor nodes. It has been found that this configuration provided better stability, and only 1st order derivatives needs to be evaluated at the boundaries anyway. It has also been suggested [24] that lower order truncations at the border in many cases actually give better accuracy than higher orders.

---

for reference=1 $\rightarrow$ total number of nodes **do**
  Calculate the local scaling (chapter 2.2).
  Calculate the Taylor series expansion (chapter 2.1).
  Calculate the weighting functions (chapter 2.3).
  Calculate the complete set of coefficients (chapter 2.3).
end for

4.5 Solver

The solver is based on the projection method explained in chapter (3.2.3), where the algebraic equations are solved using iterative successive over-relaxation (SOR) [32]. When solving Navier-Stokes equations, the advective terms can be written in either conservative– or non-conservative form. Recall the equations (3.12) and (3.13). The non-linear advection term can be expressed as

$$u_j \frac{\partial u_i}{\partial x_j} = \frac{\partial u_i u_j}{\partial x_j} - u_i \frac{\partial u_j}{\partial x_j} \tag{4.1}$$

The second term on the right hand side of equation (4.1) will vanish due to the continuity equation (3.14). Hence equation (4.1) becomes

$$\underbrace{u_j \frac{\partial u_i}{\partial x_j}}_{\text{Non-conservative}} = \underbrace{\frac{\partial u_i u_j}{\partial x_j}}_{\text{Conservative}} \tag{4.2}$$

Even though the non-conservative– and conservative form of the Navier-Stokes equations are equal on paper, they do behave numerically different. In fact, in the code written for this thesis the conservative form does not convergence to the correct solution, but the non-conservative form does and is hence adapted. The advective– and diffusive term behave differently, and should be dealt with individually for the temporal differencing. The diffusive term is made unconditionally stable by using an implicit Euler scheme. The advective terms are discretized with a semi-implicit Euler scheme, giving a conditional stability restriction with respect to the courant number. This temporal discretizing of the Navier-Stokes equations are found in [23].

$$\frac{u_i^{\overline{n+1}} - u_i^n}{\Delta t} = -u_j^n \frac{\partial u_i^{\overline{n+1}}}{\partial x_j} + \nu \frac{\partial^2 u_i^{\overline{n+1}}}{\partial x_j^2} \tag{4.3}$$

The algorithm below shows the entire solver procedure.

---

max pressure steps, tolerance, max simulation time and output interval are given by the user
$n \leftarrow 1$
$j \leftarrow 1$
repeat
  for Internal nodes **do**
    Calculate intermediate velocities $(u, v)^{\overline{n+1}}$
  end for
  for Boundary nodes **do**
    Calculate updated boundary conditions for velocities $(u, v)^{\overline{n+1}}$
  end for
  $k \leftarrow 1$
  repeat
    for Internal nodes **do**
      Calculate updated pressure $p^{n+1,k+1}$ using SOR
    end for
    for Boundary nodes **do**
      Calculate updated boundary conditions for pressure $p^{n+1,k+1}$
    end for
    $k \leftarrow k + 1$
  until abs $\left(p^{n+1,k+1} - p^{n+1,k}\right)$ <tolerance or $k >$ max pressure steps
  for Internal nodes **do**
    Calculate velocities $(u, v)^{n+1}$ using SOR
  end for
  for Boundary nodes **do**
    Calculate boundary conditions for velocities $(u, v)^{n+1}$
  end for

if $j >$ output interval **then**
Write solution $(u, v, p)^{n+1}$ to VTF-file
$j \leftarrow 0$
end if
$j \leftarrow j + 1$
$n \leftarrow n + 1$
until abs $\left((u, v)^{n+1} - (u, v)^n\right)$ <tolerance or simulation time $> n \cdot \Delta t$

As mentioned, successive over-relaxation (SOR) is used to solve the algebraic equations given by the projection method. SOR is a great improvement over the regular Gauss-Seidel iterative method and can be seen as a weighted average of the solution at iteration level k and the Gauss-Seidel solution at level $k + 1$.

$$f^{k+1} = \omega \overline{f^{k+1}} + (1 - \omega) f^k \tag{4.4}$$

ω is the relaxation factor which defines the weighting between the Gauss-Seidel solution and the previous solution in a way that may speed up– or stabilize the convergence.

Table 4.5: Relaxation factors

ω	**Relaxation**
$0 < \omega < 1$	Under
$1 \leq \omega \leq 1$	Gauss-Seidel
$1 < \omega < 2$	Over

Over-relaxation is used in order to increase the numerical convergence speed, and under-relaxation is used to increase the stability of an else diverging solution. The relaxation factor is given in the input file (chapter 4.1) and can be adjusted to speed up– or stabilize the convergence. To illustrate, the PDE from chapter 2.5 have been repeatedly solved on three different node distributions with uniform node spacing h and with relaxation factor ω varying from $0 \rightarrow 2$.

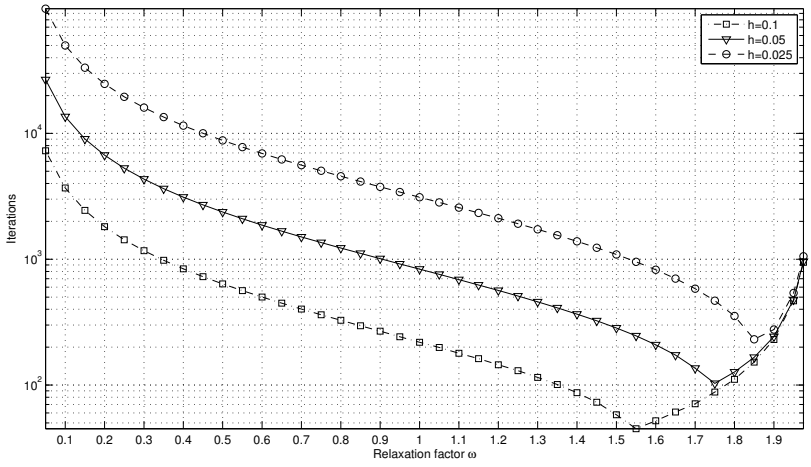


Figure 4.5: Number of iteration steps for varying relaxation factors.

From figure 4.5 it is seen that the solution converges faster for increasing ω up to an optimum relaxation factor. Clearly, this optimum ω differ with the node spacing, and seems to increase for finer distributions. For a Poisson equation on a square domain with regular node spacing h, the optimum ω is given as

$$\omega_{optimum} = \frac{2}{1 + \sqrt{1 - \left(1 - 0.5\,(\pi h)^2\right)^2}} \qquad [32] \qquad (4.5)$$

The optimum ω for each node distribution in figure 4.5 can therefore be calculated using equation (4.5).

Table 4.6: Optimum relaxation factors

h	$\omega_{optimum}$
0.1	1.53
0.05	1.73
0.025	1.85

The analytical relaxation factors in table 4.6 match the numerically calculated optimums in figure 4.5 very well. However, identifying the optimum relaxation factor for a general domain with irregular node spacing is not straight forward. Therefore, in the developed code the relaxation factor has to be given by the user based on trial and error. It is however advisable to initially apply over-relaxation

and in the case of divergence, adjust the parameter. For a given node distribution, set of boundary conditions, and fluid properties there are in fact five parameters that can be adjusted in order to control the numerical convergence and stability.

1. Number of neighbor nodes
2. Relaxation factor
3. Maximum pressure steps
4. Time step
5. Courant number

By increasing the number of neighbor nodes, effective node spacing will increase due to enlarged local domain size. This may lead to faster convergence or better stability conditions. On the other hand, since the local domain size is increased, the spatial accuracy will decrease as described in chapter 2.2. It is hence advisable to keep the number of neighbors as low as possible. In figure 4.6, the PDE from chapter 2.5 have been solved using uniform node spacing $h = 0.05$, relaxation factor $\omega = 1.5$, and tolerance criteria $\epsilon_{tol} = 10^{-10}$ in order to illustrate the effect of increasing number of neighbor nodes.

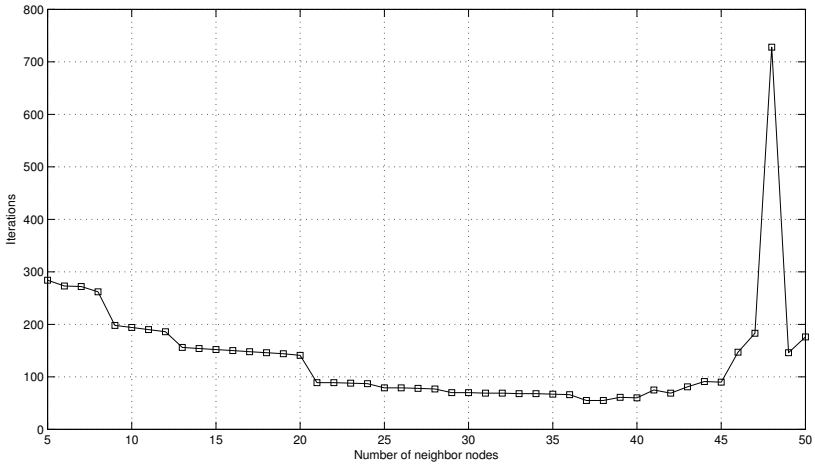


Figure 4.6: Number of iteration steps for varying number of neighbor nodes.

Initially, the pressure-Poisson equation is solved with a given tolerance criteria ϵ_{tol}. However, if steady state solution is required only, the pressure field does not need to be fully converged at every time step. Therefore, by imposing a finite number of iteration steps for the pressure-Poisson equation, the pressure will basically work as a stabilizer for the solution. Increasing number of iteration steps yields a better solution at each time step, giving a more stable solution procedure, though increasing the computational cost at the same time.

The CFL condition requires, as explained in chapter 3.4, Courant number $c < 1$ in order to achieve a stable solution for an explicit method. For a given node distribution with specified boundary conditions, Δt is the only adjustable variable. In the developed code, the user is given the opportunity to specify a minimum– and maximum Courant number together with an initial time step. The program will then automatically optimize Δt at the end of each time step with respect to the given courant number restriction.

Calculate the maximal Courant number $c_{max} = \max\left(\frac{\sqrt{u^2+v^2}\Delta t}{d}\right)$, where d is the local domain size.
if $c_{max} >$ maximum allowed Courant number **then**
  $\Delta t \leftarrow 0.90\Delta t$
else if $c_{max} <$ minimum allowed Courant number **then**
  $\Delta t \leftarrow 1.11\Delta t$
end if

Δt can be kept constant and equal to the initial time step by defining the maximum Courant number $c_{max} = 0$. This is usually preferred in case of transient analysis.

Chapter 5

Analysis

In order to verify the developed code, a series of analysis have been performed. Among them are two cases for which analytical solutions exist, and two cases of more complicated nature without any analytical solution.

5.1 Rotating Couette flow

Couette flows are named after the French physicist Maurice Couette, who performed experiments on fluid flow between fixed and rotating concentric cylinders. The flow regime is caused entirely by the driven wall and is viscous dominated.

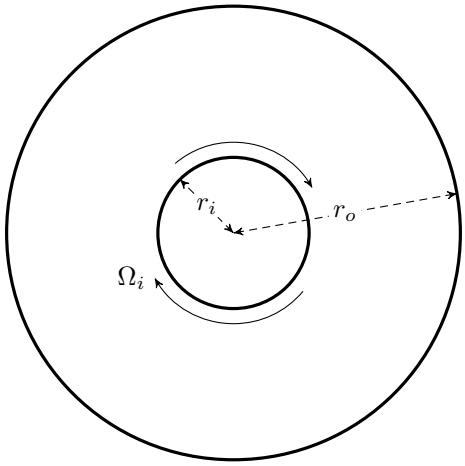


Figure 5.1: Concentric Couette flow

The analytical solution for the problem posed in figure 5.1 is given in [30].

$$u_\theta = \Omega_i r_i \frac{r_o/r - r/r_o}{r_o/r_i - r_i/r_o} \tag{5.1}$$

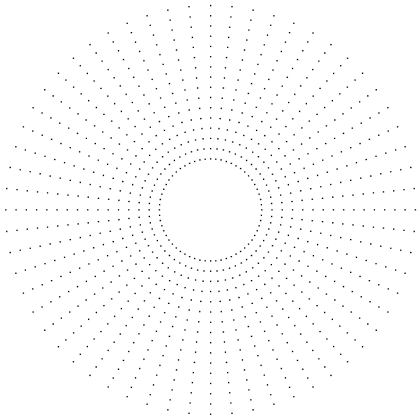
Equation (5.1) is valid for an incompressible flow up to a certain limit, given by the relation between angular velocity Ω_i, inner– and outer radius (r_i and r_o) and kinematic viscosity of the fluid ν. The relation is called the *Taylor number* Ta [30] and is defined as

$$Ta = \frac{r_i \left(r_o - r_i\right)^3 \Omega_i^2}{\nu^2} \tag{5.2}$$

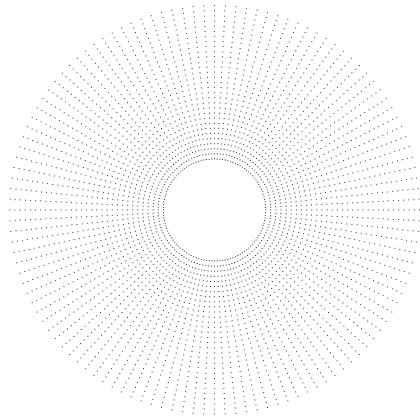
Below the critical value $Ta_{critical} \approx 1700$, the analytical solution is valid. In order to compare the analytical solution with numerical approximations, Ta hence have to be lower than $Ta_{critical}$. For the numerical calculation, a geometry with $r_i = 1\,[\mathrm{m}]$ and $r_o = 4\,[\mathrm{m}]$ have been applied together with a kinematic fluid viscosity of $\nu = 1\,[\mathrm{Pa \cdot s}]$. With these properties, the angular velocity is restricted to

$$\Omega_i = \sqrt{\frac{1700\nu^2}{r_i \left(r_o - r_i\right)^3}} = 7.93\left[\mathrm{s}^{-1}\right] \tag{5.3}$$

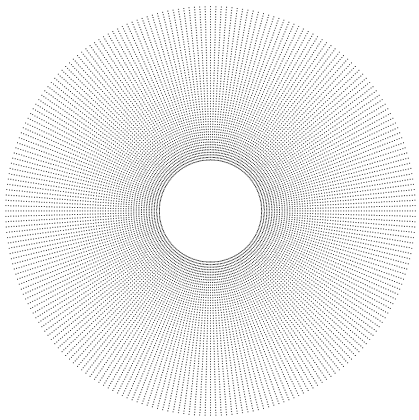
The analysis have therefore been performed with $\Omega_i = 7\left[\mathrm{s}^{-1}\right]$, giving a tangential velocity at the inner wall $U_{\tau,i} = r_i\Omega_i = 7\left[\mathrm{m \cdot s}^{-1}\right]$. The calculation have been done using the four different node distributions shown in figure 5.2. The number of truncated terms are 2, and number of auxiliary nodes are 9. The convergence tolerance ϵ_{tol} is set to 10^{-10} and the courant number is restricted between 0.5 and 0.8. The results in figure 5.3(a) are difficult to interpret directly, though clearly shows that the analysis have successfully captured the physics. The plot in figure 5.3(b) on the other hand, shows the relative errors between the analytical– and numerical solutions in a way that makes it much easier to confirm that increasing node density provides a more accurate solution. The node distribution in figure 5.2(d) however, performs relatively poor compared to the others with respect to total node number. Additionally, for the node distribution in figure 5.2(c), an analysis with various time step have also been executed, for which it is clearly shown in figure 5.3(d) that decreasing time step produce more accurate results. However, when the number of truncation terms are increased (figure 5.3(c)) using the node distribution in figure 5.2(c) and 25 neighbor nodes, 3 truncation terms provide more accurate results than 4. The reason for this is unknown. Additional field plots can be found in Appendix B.



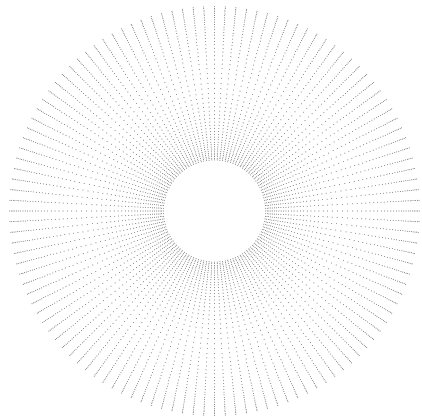
(a) 960 nodes, uniform distribution



(b) 3.720 nodes, uniform distribution

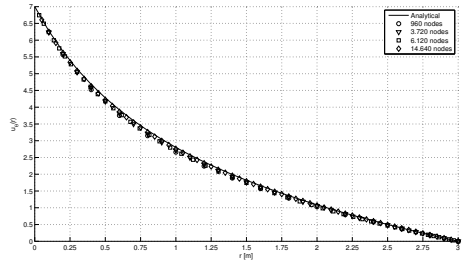


(c) 14.640 nodes, uniform distribution

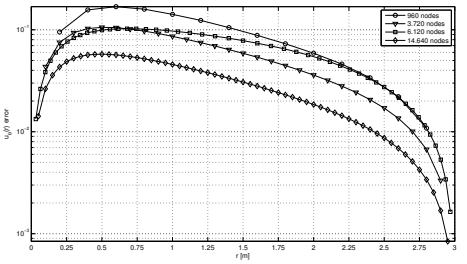


(d) 6.120 nodes, refined uniform distribution

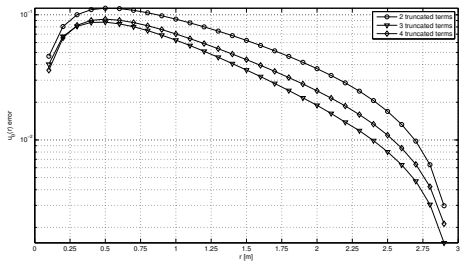
Figure 5.2: Node distribution for the Couette flow problem



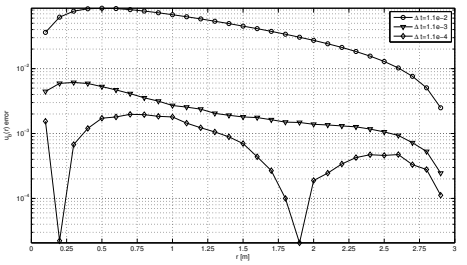
(a) Angular velocity $u_\theta(r)$



(b) $u_\theta(r)$ error with various node distributions



(c) $u_\theta(r)$ error with 3.720 nodes and various truncation terms



(d) $u_\theta(r)$ error with 3.720 nodes and various time step

Figure 5.3: Angular velocity plot from $r_i \to r_o$ for the Couette flow problem

5.2 Poiseuille channel flow

The previous analytical flow case was driven entirely by the rotating cylinder wall. This time, a pressure driven flow shall be considered. The steady Poiseuille flow in a channel is driven by a pressure gradient parallel to the channel walls, and the analytical solution can be derived directly from the Navier-Stokes equations by assuming two dimensions and infinitely long channel. The vertical velocity v will hence vanish, and the horizontal velocity u will be a function of y only.

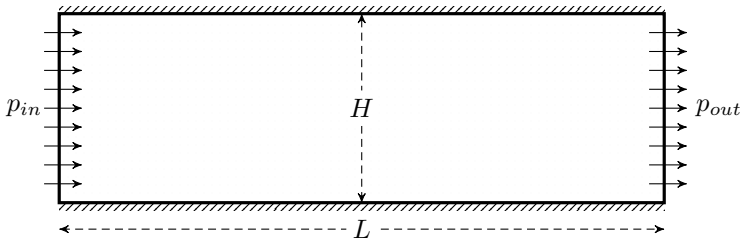


Figure 5.4: Poiseuille flow

The original two dimensional Navier-Stokes equations (3.12) and (3.13) will reduce to an ordinary differential equation.

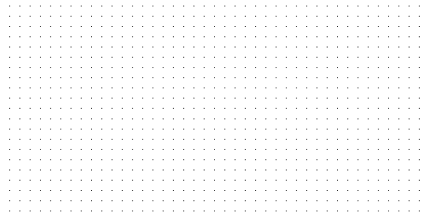
$$\frac{\partial^2 u}{\partial y^2} = \frac{1}{\mu}\frac{\partial p}{\partial x} \tag{5.4}$$

By applying no-slip condition on the top– and bottom of the channel, located at respectively $y = 0$ and $y = H$, the final analytical expression is thus obtained.

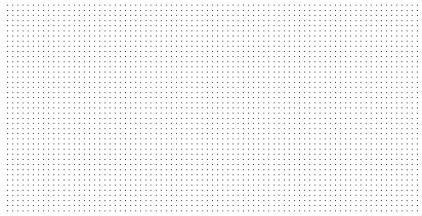
$$u\,(y) = \frac{1}{2\mu}\frac{dp}{dx} y\,(y - H) \tag{5.5}$$

In order to have constant velocity in the horizontal direction, the pressure gradient will have to be constant i.e. linear pressure drop along the channel length. To numerically investigate the problem, a short channel have been modeled as shown in figure 5.4. In the numerical model, $L = 2\,[\mathrm{m}]$ and $H = 1\,[\mathrm{m}]$. Pressure conditions are imposed at the ends and no-slip condition is applied to the top– and bottom. In total, four different node distributions have been considered (figure 5.5), for which three of them are strictly uniform, and the 4th is uniform, though denser near the boundaries. In figure 5.6(a) and 5.6(c), the four node distribution have been tested against each other with a pressure drop of $8\,[\mathrm{Pa}]$, giving an effective pressure gradient of $4\left[\mathrm{Pa}\cdot\mathrm{m}^{-1}\right]$, which can evidently be observed in figure 5.6(b) as well. According to the analytical solution this corresponds to a maximum velocity $u_{max} = 0.5\left[\mathrm{m}\cdot\mathrm{s}^{-1}\right]$. Increasing number of nodes gives, as expected, better accuracy. The node distribution in figure 5.5(d) performs very well near the no-slip boundaries, though less accurate in the center. Figure 5.6(d) shows three calculations, all performed using the node distribution in figure 5.5(b). The simulations have been done using varying pressure drops, though with adjusted viscosity parameter so that the velocity profile shall remain the same. From the figure it is clear that by decreasing the pressure drop and viscosity, a more accurate solution will be obtained. The reason for this could be due to a lower importance of the diffusive term in the Navier-Stokes equations, resulting in reduced truncation error. Additional field plots are given in appendix C.

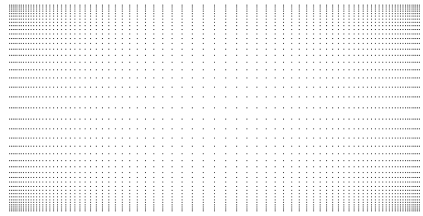
(a) 231 nodes, uniform distribution



(b) 861 nodes, uniform distribution

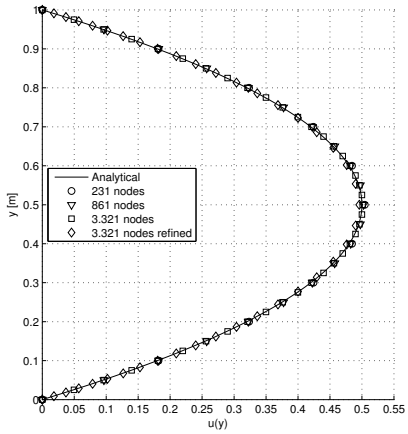


(c) 3.321 nodes, uniform distribution

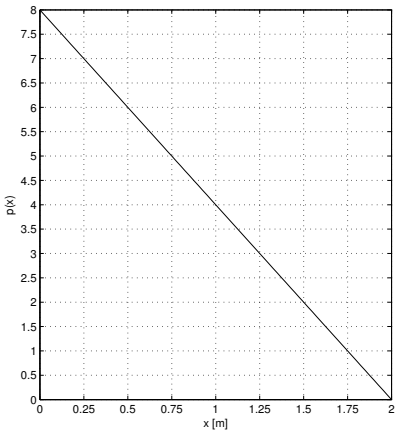


(d) 3.321 nodes, refined uniform distribution

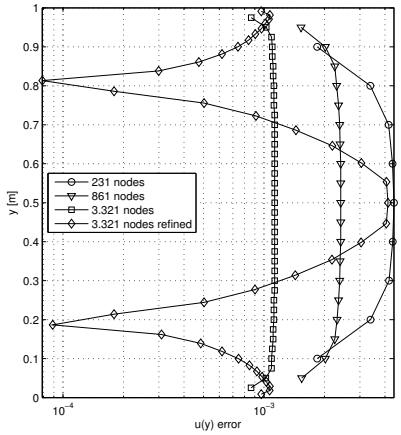
Figure 5.5: Node distribution for the Poiseuille flow problem



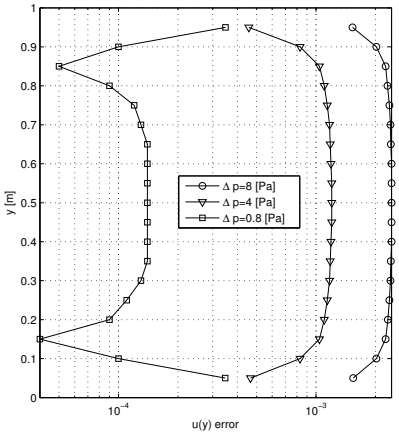
(a) Axial velocity $u(y)$



(b) Pressure drop along channel length



(c) $u(y)$ error with various node distributions



(d) $u(y)$ error with various Δp

Figure 5.6: Axial velocity plot and pressure drop for the Poiseuille flow problem

5.3 Lid-driven cavity flow

The lid-driven cavity flow has been extensively used to test numerical schemes. The geometry shown in figure 5.7 is simple, and the boundary conditions are pretty straight forward to implement as well, though the singularity corners at the top have to be treated carefully. The flow regime, on the other hand is fairly complex with its circulating flow patterns changing with respect to the Reynolds number.

In order to verify the developed code, the work of Ghia et al. [15] and Wan et al. [29] have been used as reference material.

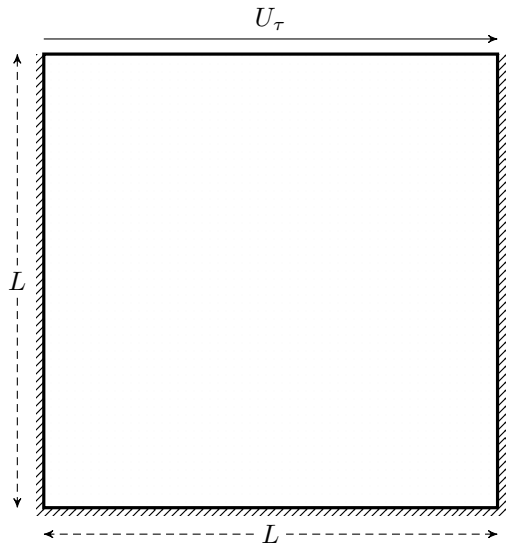


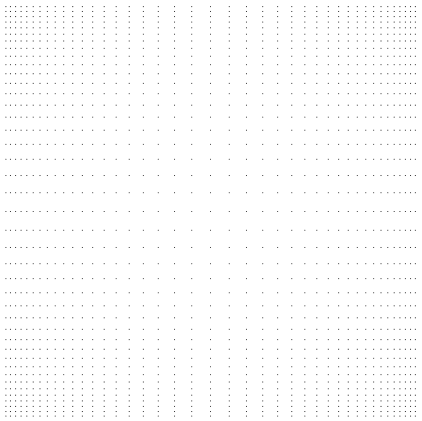
Figure 5.7: Lid-driven cavity flow

Calculations have been performed for Reynolds numbers Re 100, 400, 1000 and 3200. By keeping $U_\tau = 1\,[\mathrm{m \cdot s^{-1}}]$, $L = 1\,[\mathrm{m}]$ and density $\rho = 1\,[\mathrm{kg \cdot m^{-3}}]$, the dynamic viscosity μ has been systematically altered in order to obtain the correct Re.

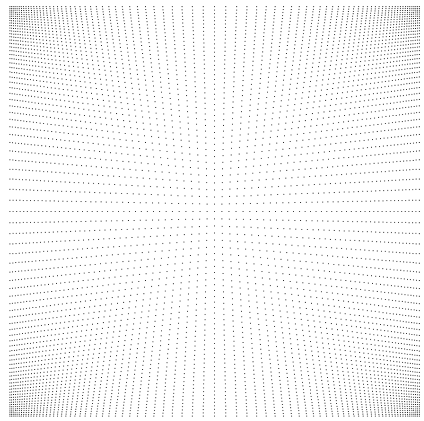
$$\mu = \frac{\rho U_\tau L}{Re} \tag{5.6}$$

Four different node distribution shown in figure 5.8 have been applied to the calculations, though not all of them are used for every Reynolds number. Analysis have been performed using 2 truncated terms and 9 neighbor nodes, with convergence tolerance $\epsilon_{tol} = 10^{-8}$. Increasing the truncation terms quickly led to divergence. The resulting u-velocities are plotted in the y-direction along the geometrical center $x = 0.5\,[\mathrm{m}]$, while the v-velocities are plotted in the x-direction along $y = 0.5\,[\mathrm{m}]$. The velocity plots in figure 5.9 clearly show good comparison with the reference values [15] and [29], especially for low Re numbers. However, in [15] one of the tabulated u-velocities for $Re = 3200$ and v-velocities for $Re = 400$ are believed to be typographical errors. Never the less, they are plotted in figure 5.9. It is clear that for increasing Re-numbers, a finer node distribution is needed in order to capture the flow details entirely. Figure 5.10 shows the vector fields for the four individual Reynolds numbers. In addition to the primary eddy induced by

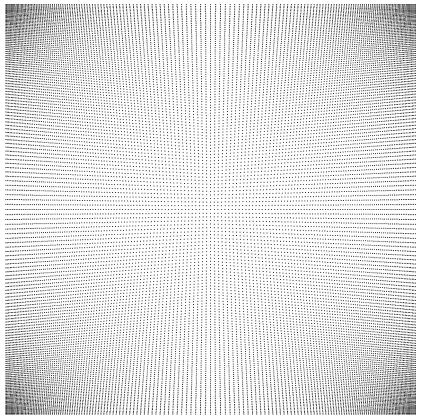
the driven top wall, smaller secondary eddies can be found in the bottom corners. These circulating regions are present in all the calculated cases, but clearly become larger with increasing Reynolds number. In figure 5.10(d), a 3rd eddy has appeared in the upper left corner. The appearance and size of all these eddies corresponds very well to [15].



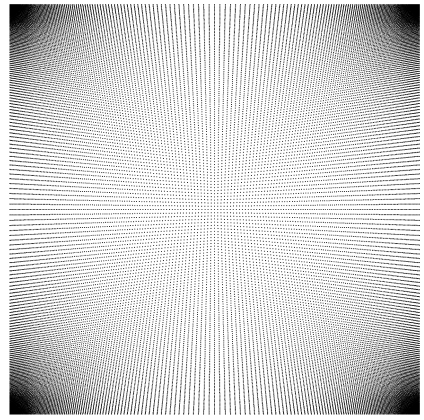
(a) 1.764 nodes, denser distribution near walls



(b) 6.724 nodes, denser distribution near walls

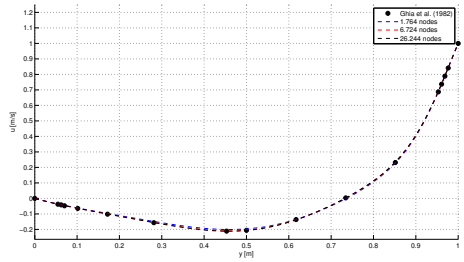


(c) 26.244 nodes, denser distribution near walls

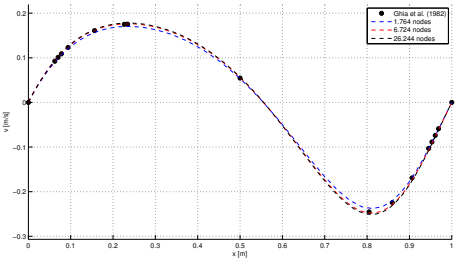


(d) 40.804 nodes, denser distribution near walls

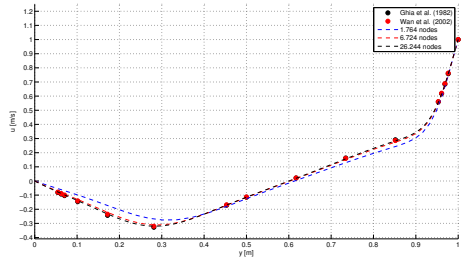
Figure 5.8: Node distribution for the lid-driven cavity flow problem



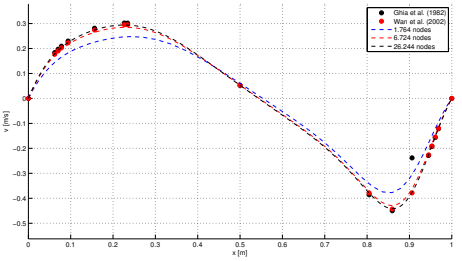
(a) $u\,(x = 0.5, y)$ $Re = 100$



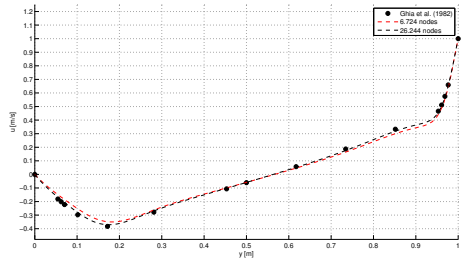
(b) $v\,(x, y = 0.5)$ $Re = 100$



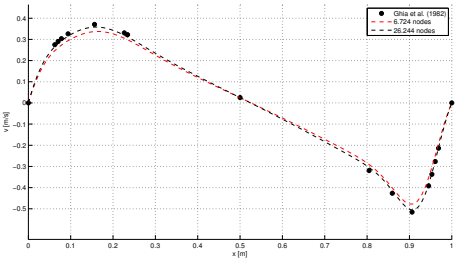
(c) $u\,(x = 0.5, y)$ $Re = 400$



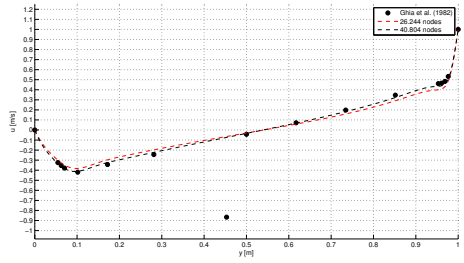
(d) $v\,(x, y = 0.5)$ $Re = 400$



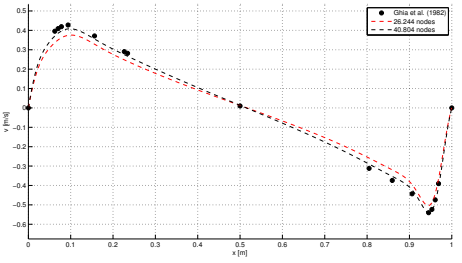
(e) $u\,(x = 0.5, y)$ $Re = 1000$



(f) $v\,(x, y = 0.5)$ $Re = 1000$

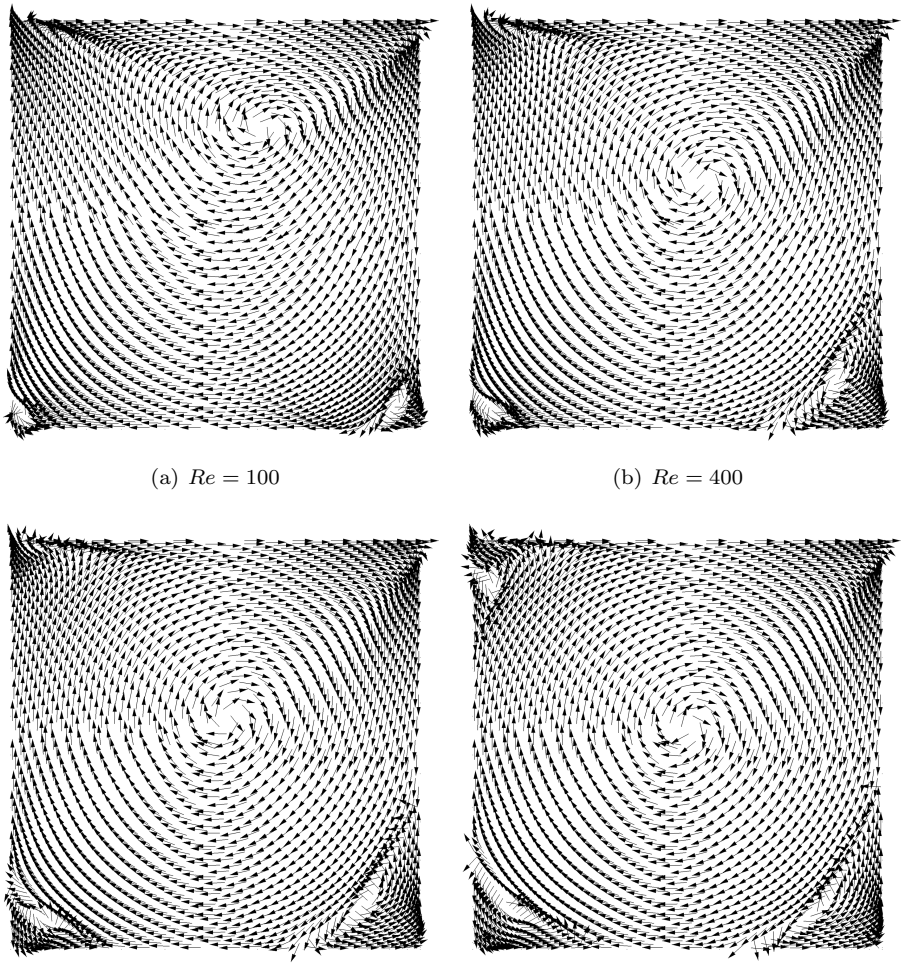


(g) $u\,(x = 0.5, y)$ $Re = 3200$



(h) $v\,(x, y = 0.5)$ $Re = 3200$

Figure 5.9: Velocity comparison for the lid-driven cavity flow problem



(a) $Re = 100$ (b) $Re = 400$

(c) $Re = 1000$ (d) $Re = 3200$

Figure 5.10: Velocity vectors for the lid-driven cavity flow problem

5.4 Viscous flow past a circular cylinder

One of the most popular test cases in CFD is the circular cylinder in uniform incident flow. The flow past slender cylindrical bodies are important in both on– and offshore engineering, especially due to fatigue caused by vortex induced vibrations (VIV). The problem is well documented due to extensive research, both experimentally and numerically, which makes it very attractive as a benchmark case. As a matter of fact, the problem was solved numerically by hand nearly 80 years ago by Thom (1933). One of the major motivations for studying the viscous flow past

blunt bodies is that the potential flow theory does not give satisfactory solutions due to the large viscous effects.

For low Re i.e. viscous dominated flows, any perturbation will be damped out and the flow is considered steady. The pressure distribution over the cylinder surface will clearly show a favorable (negative) gradient up to about 90° from the front stagnation point, and then turn adverse (positive) in the region $90° < \theta < 180°$. The adverse pressure gradient will lead to boundary layer separation somewhere along the rear part of the cylinder body forming a wake downstream, also called the separation bubble. As seen in figure 5.11, the separation bubble reattach at the wake stagnation point downstream of the cylinder. It is widely agreed upon that the flow will remain steady for $Re < 40$. Increasing the Reynolds number further will perturb the steady behavior and lead to transient oscillations in the system.

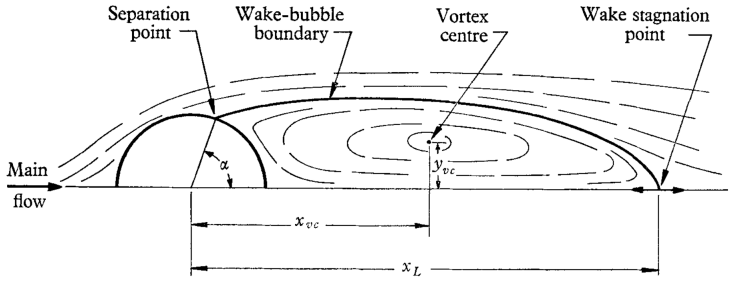


Figure 5.11: Steady flow past a circular cylinder. From Grove et al. [17]

In the unsteady regime, the pressure distribution along the cylinder will no longer be symmetric over the cylinder half shown in figure 5.11. This non-symmetrical distribution will eventually lead to periodic lift– and drag forces, which may cause VIV to initiate in case of slender bodies, i.e. onshore chimneys and offshore pipelines. The steady separation bubble have disappeared and instead a vortex street can be observed behind the cylinder body. The frequency of which these vortexes are shed can be made dimensionless with the *Strouhal number* St defined as

$$St = \frac{f_v D}{U_\infty} \tag{5.7}$$

where f_v is the shedding frequency, D is the cylinder diameter and U_∞ is the uniform incident velocity. It is found that for a large range of Reynolds numbers $200 < Re < 2 \cdot 10^5$, the Strouhal number will remain ~ 0.2.

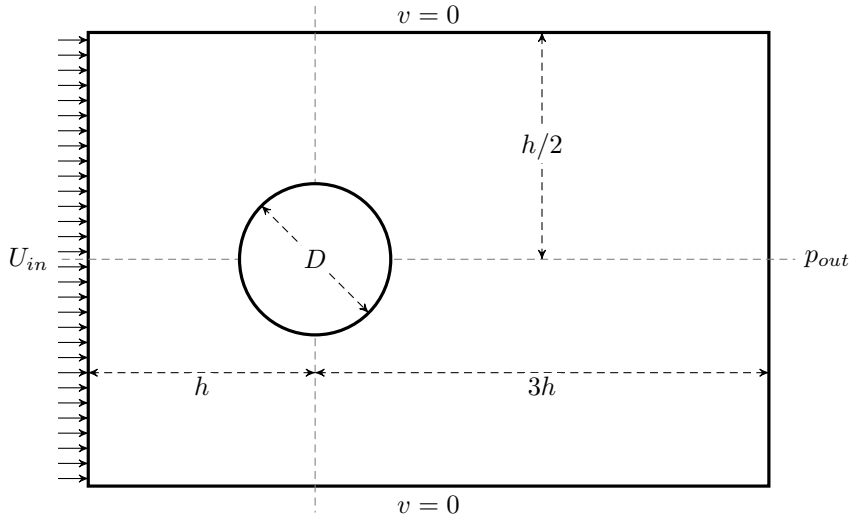
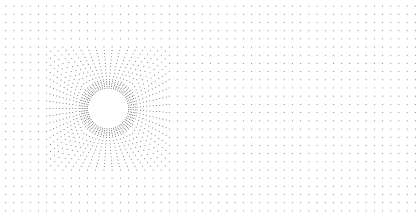
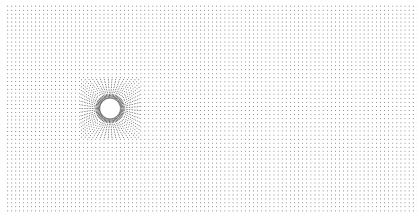


Figure 5.12: Flow past a circular cylinder

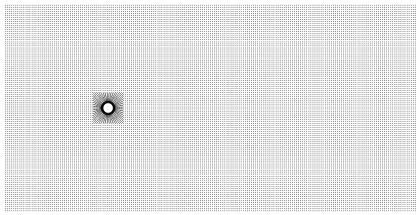
In the current analysis, the solution have been calculated for $Re = 40$ and 100 using the four different node distributions shown in figure 5.13. The effect of blocking, i.e. wall boundaries located near the immersed body, have been examined by varying the factor D/h in figure 5.12. All analysis have been performed using 6 neighbor nodes and 2 truncated terms in order to generate the LSFD coefficients. By keeping the inlet velocity $U_{in} = 1\,[\mathrm{m} \cdot \mathrm{s}^{-1}]$, the diameter $D = 1\,[\mathrm{m}]$ and the fluid density $\rho = 1\,[\mathrm{kg} \cdot \mathrm{m}^{-3}]$, the dynamic viscosity μ has been adjusted to achieve the desired Reynolds number. For the steady analysis at $Re = 40$, a time step of $\Delta t = 0.05\,[\mathrm{s}]$ has been chosen. In order to capture the more complex and oscillating flow field appearing in the transient analysis at $Re = 100$, a time step of $\Delta t = 0.01\,[\mathrm{s}]$ has been adapted. Convergence tolerance $\epsilon_{tol} = 10^{-7}$ has been chosen for the steady-state analysis. For the transient analysis, 100 [s] are simulated using maximum 200 iterations for the pressure-Poisson equations in order to satisfy the incompressible flow requirement. If too few iterations are used, the flow will not become fully incompressible resulting in a slow varying pressure oscillation in the whole domain due to low damping.



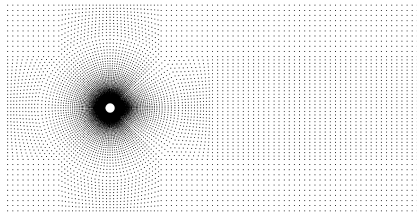
(a) 2.367 nodes, $D/h = 0.20$



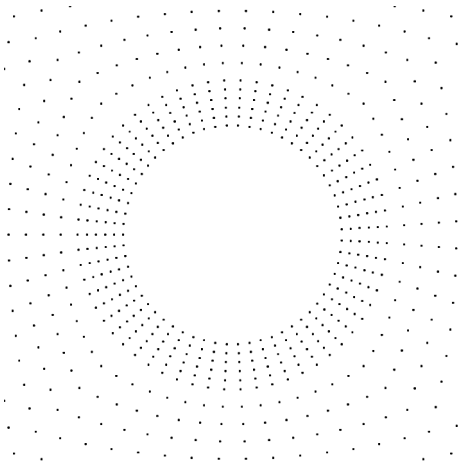
(b) 6.538 nodes, $D/h = 0.10$



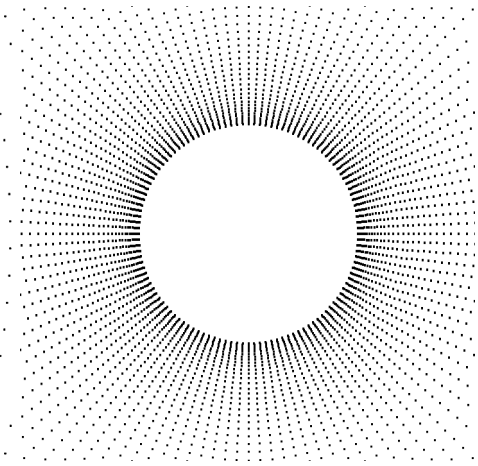
(c) 22.213 nodes, $D/h = 0.05$



(d) 12.329 nodes, $D/h = 0.05$



(e) Close-up on cylinder for 5.13(a), 5.13(b) and 5.13(c).

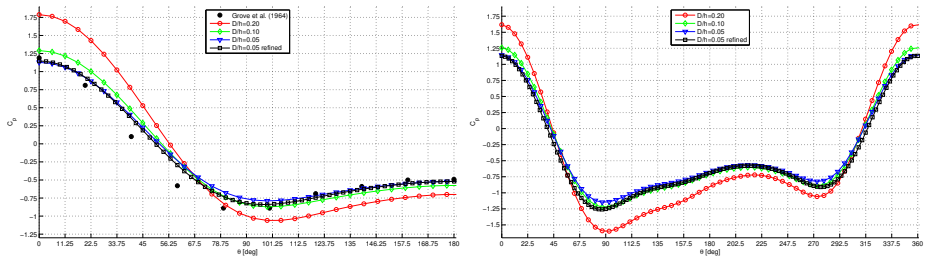


(f) Close-up on cylinder for 5.13(d).

Figure 5.13: Node distribution for the cylinder flow problem

As expected, the calculation at $Re = 40$ converges to a steady-state solution with a symmetrical wake appearing right behind the cylinder body as shown in figure 5.11. The blocking effect is observed by plotting the the pressure distribution along the cylinder surface, starting from the front stagnation point. Due to symmetric behavior in the steady solution, only the upper half of the cylinder is considered in figure 5.14(a). From the figure it is clear that by decreasing the blocking, the pres-

sure distribution will converge to the correct free stream solution. The calculation with blocking factor $D/h = 0.20$ reveals large interaction with the boundary walls, which will result in incorrect drag force and separation point.



(a) $Re = 40$ (b) $Re = 100$

Figure 5.14: Pressure distribution along the cylinder surface for the steady– and unsteady case.

In case of $Re = 100$, the pressure distribution will no longer be steady and symmetrical. In figure 5.14(b), the pressure is shown along the surface at the moment of upper vortex shedding. In figure 5.15, the two extreme distributions are plotted. Vortex shedding 1 represents the pressure distribution at the moment of lower vortex shedding, while Vortex shedding 2 represents the upper. Each of the two distributions are symmetric to the other at $\theta = 180°$.

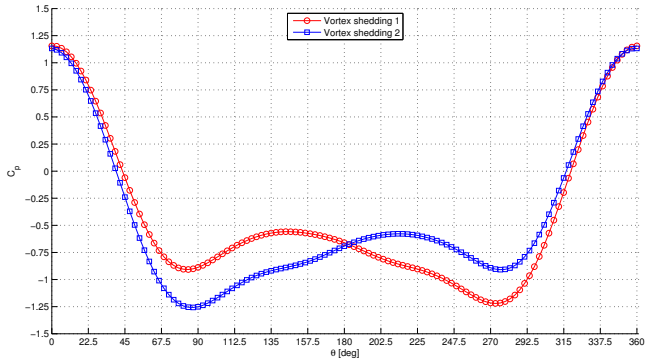


Figure 5.15: Pressure distribution along the cylinder surface for $Re = 100$.

The point of separation has been determined for each simulation and tabulated in table 5.1 together with earlier observations for comparison. For the node distribution in figure 5.13(d) separation occurs much earlier compared to the other simulations and the reference values. The reason for this is however unknown.

Table 5.1: Separation point for the viscous flow past a cylinder.

Re	**Source**	θ_{sep}
40	Present $D/h = 0.20$	51.0
	Present $D/h = 0.10$	51.0
	Present $D/h = 0.05$	69.0
	Ding et al. [7]	53.5
100	Present $D/h = 0.20$	60.0 ± 3.0
	Present $D/h = 0.10$	60.0 ± 3.0
	Present $D/h = 0.05$	67.5 ± 1.5
	Kjell Herfjord [19]	60.0 ± 3.0
	Grove et al. [17] $D/h = 0.20$	57.5

The vector field in figure 5.17(a) clearly shows how the steady separation bubble behind the cylinder at $Re = 40$ is formed by two circulating regions which reattach a couple of diameters downstream the cylinder body. Figure 5.17(b) shows how a vortex is being generated at the lower rear of the cylinder surface. In order to determine the shedding frequency, and hence find the dimensionless Strouhal number, a time series has been extracted from a point downstream the cylinder where the velocities fluctuate with time. The time series has later been transformed into the frequency domain shown in figure 5.16 using fast Fourier transform (FFT) from which the shedding frequency can be directly read out. The Strouhal number converges to $St = 0.16$ already with blocking factor $D/h = 0.10$, and is hence not very sensitive to the boundaries. $St = 0.16$ compares quite well to earlier observations [19]. Additional field plots can be found in appendix E.

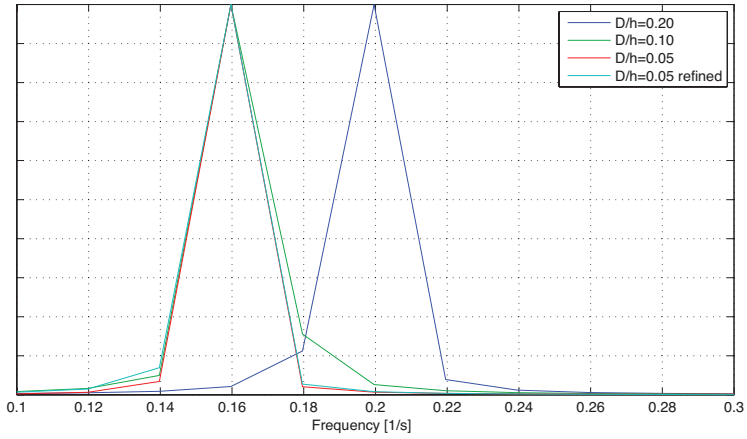
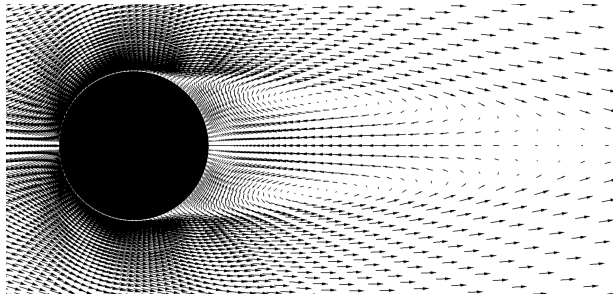
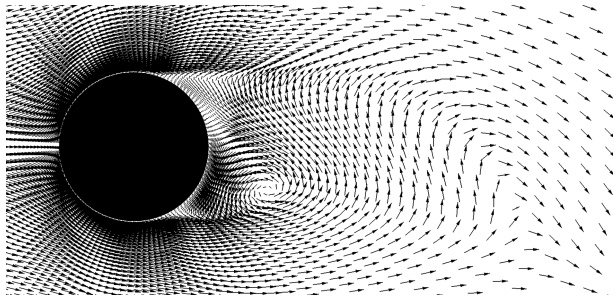


Figure 5.16: Shedding frequency at $Re = 100$



(a) $Re = 40$



(b) $Re = 100$

Figure 5.17: Vector field for the steady– and unsteady cylinder flow.

Chapter 6

Discussion

During this thesis, the meshfree LSFD procedure has been explained throughly and implemented in a code for solving two dimensional incompressible flow problems. The program has been generalized as much as possible within the given restrictions. Nevertheless, a list of important shortcomings can, and shall be highlighted. The results from the analysis part have already been discussed, and the program itself, with its limitations and suggestions for further work, will be subjected for discussion in the following. Despite the drawbacks, the program is entirely capable of solving incompressible two dimensional problems as it initially was meant for, something the analysis in chapter 5 clearly show.

6.1 Neighbor node identification

The neighbor nodes have been determined using the element information given in the geometry input file from MEGA. However, if the procedure is depending on any element information, it is not entirely meshfree. The program should be able to identify the neighbor nodes using the node coordinates only. Efficient methods for this task do exist, i.e. binary tree search and Delaunay triangulation, but have not been the subject of this thesis. However, as for further work these methods should definitely be looked into. With more efficient search algorithms, node adaptivity could also be developed in order to refine the node distribution in regions with large gradients. In fact, this is one of the great advantages with the meshfree concept.

6.2 Three dimensional flows

The present program supports only two dimensional flows, which of course does not exist in reality. However, the LSFD procedure is quite general and it should

definitely be possible to expand the program to three dimensions without too many complications. As a matter of fact, this has already been done for a method closely related to LSFD [8]. When three dimensions are settled, turbulent flows should also be looked further into. Of course, with one more dimension even more care have to be taken in terms of efficient programming since the computational expense will increase tremendously.

6.3 Solution method

When it comes to the solution method, the foundations are settled. However, the method is of 1st order only, which is a huge restriction in terms of accuracy. 2nd order projection methods does exist and should be applied. Though with 2nd order methods, the boundary conditions will become more complicated. As for the boundary conditions in their present form, they are quite restrictive. The program does not allow the end user to specify the x– and y component of the velocity at inlet boundaries for example, only the normal component. Periodic boundary condition have not been treated in this thesis, but is definitely a task subjected for future improvement. Another issue which should be improved, is the solution stability. The semi-implicit projection method applied to the program will give restrictions in terms of the courant number. Especially for steady-state analysis, a fully implicit method would be preferable.

6.4 Graphical user interface

The program in its present form is pure command line code. With no graphical user interface, the program will be less user oriented and quite hard to understand. For the most obvious errors during the calculation process i.e. singular coefficient matrix, wrong input file etc., messages will be printed to the screen in order to inform the user of where the program fails. For further work, a graphical user interface with an intuitive options handler should be developed. Live convergence information should also be implemented in order to give the user an idea of where the analysis might fail.

Bibliography

[1] Numerical solutions of 2-D steady incompressible flow over a backward-facing step, Part 1: High Reynolds number solutions. *Computers & Fluids*, 37(6):633 – 655, July 2008.

[2] John B. Bell, Phillip Colella, and Harland M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 85(2):257 – 283, 1989.

[3] Yongchang Cai and Hehua Zhu. A local search algorithm for natural neighbours in the natural element method. *International Journal of Solids and Structures*, 42(23):6059 – 6070, 2005.

[4] Alexandre Joel Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2(1):12 – 26, 1967.

[5] Filippo Maria Denaro. On the application of the Helmholtz-Hodge decomposition in projection methods for incompressible flows with general boundary conditions. *International Journal for Numerical Methods in Fluids*, 2003.

[6] H. Ding, C. Shu, K. S. Yeo, and D. Xu. Development of least-square-based two-dimensional finite-difference schemes and their application to simulate natural convection in a cavity. *Computers & Fluids*, 33(1):137 – 154, 2004.

[7] H. Ding, C. Shu, K. S. Yeo, and D. Xu. Simulation of incompressible viscous flows past a circular cylinder by hybrid FD scheme and meshless least square-based finite difference method. *Computer Methods in Applied Mechanics and Engineering*, 193(9-11):727 – 744, 2004.

[8] H. Ding, C. Shu, K. S. Yeo, and D. Xu. Numerical computation of three-dimensional incompressible viscous flows in the primitive variable form by local multiquadric differential quadrature method. *Computer Methods in Applied Mechanics and Engineering*, 2006.

[9] H. Ding, C. Shu, K. S. Yeo, and D. Xu. Numerical simulation of flows around two circular cylinders by mesh-free least square-based finite difference methods. *International Journal for Numerical Methods in Fluids*, 53(2):305 – 332, 2007.

[10] Qing dong CAI. Explicit formulations and performance of LSFD method on Cartesian mesh. *Applied Mathematics and Mechanics*, 30(2):183 – 196, 2009.

[11] C. Henry Edwards and David E. Penney. *Calculus*. Prentice-Hall, 6 edition, 2002.

[12] Joel H. Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics*. Springer, 1997.

[13] Ali Rahmani Firozjaee and Mohammad Hadi Afshar. Steady-state solution of incompressible Navier-Stokes equations using discrete least-squares meshless method. *International Journal for Numerical Methods in Fluids*, 2010.

[14] C.A.J. Fletcher. *Computational Techniques for Fluid Dynamics*, volume 1. Springer, 2 edition, 1996.

[15] U. Ghia, K.N. Ghia, and C.T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387 – 411, December 1982.

[16] G.R.Liu and Y.T. Gu. *An Introduction to Meshfree Methods and Their Programming*. Springer, 2005.

[17] A.S. Grove, F.H. Shair, E.E. Petersen, and Andreas Acrivos. An experimental investigation of the steady separated flow past a circular cylinder. *Cambridge University Press*, 1964.

[18] J.L. Guermond, P. Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 2006.

[19] Kjell Herfjord. *A study of two-dimensional separated flow by a combination of the finite element method and Navier-Stokes equations*. PhD thesis, The Norwegian Insitute of Technology, 1995.

[20] Joseph O'Rourke. *Computational geometry in C*. Cambridge University Press, second edition, 1998.

[21] David E. Penney and C. Henry Edwards. *Elementary Linear Algebra*. Prentice-Hall, 1988.

[22] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2 edition, 2002.

[23] Samuel R. Ransau. *Numerical Methods for Flows with Evolving Interfaces*. PhD thesis, Norwegian University of Science and Technology, 2004.

[24] Patrick J. Roache. *Fundamentals of Computational Fluid Dynamics*. Hermosa Publishers, 1998.

[25] C. Shu, H. Ding, and N. Zhao. Numerical comparison of least square-based finite-difference (LSFD) and radial basis function-based finite-difference

(RBFFD) methods. *Computers & Mathematics with Applications*, 51(8):1297 – 1310, 2006.

[26] C. Shu, W.X. Wu, H. Ding, and C.M. Wang. Free vibration analysis of plates using least-square-based finite difference method. *Computer Methods in Applied Mechanics and Engineering*, pages 1330 – 1343, 2007.

[27] T.E. Tezduyar and J. Liou. On the downstream boundary conditions for the vorticity-stream function formulation of two-dimensional incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 1991.

[28] Michael Vine. *C Programming for the Absolute Beginner*. Premier Press, 2002.

[29] D.C. Wan, B.S.V. Patnaik, and G.W. Wei. Discrete singular convolution–finite subdomain method for the solution discrete singular convolution–finite subdomain method for the solution discrete Singular Convolution-Finite Subdomain Method for the Solution of Incompressible Viscous Flows. *Journal of Computational Physics*, pages 229 – 255, 2002.

[30] Frank M. White. *Fluid Mechanics*. McGraw-Hill, 5 edition, 2005.

[31] Frank M. White. *Viscous Fluid Flow*. McGraw-Hill, 3 edition, 2006.

[32] David M. Young. *Iterative Methods for Solving Partial Difference Equations of Elliptic Type*. PhD thesis, Harvard University, 1951.

Appendix A

Program layout

As a supplementary to chapter 4, the input text file and a program flow chart have been included in this appendix. The source code however, has only been included electronically.

```
FILE PROPERTIES
Project name: name
Input file: geometry.vtf

FLUID PROPERTIES
Density: 1.0
Dynamic viscosity:  0.01

NUMERICAL PROPERTIES
Number of auxiliary nodes: 6
Number of truncated terms: 2
Residual error: 1e-7
Relaxation factor: 1.5
Pressure steps: 10

TIME SETUP
Timestep: 0.01
Time duration: 1000
Courant min max: 0.5 0.8
Output interval: 200

BOUNDARY CONDITIONS
1 noslip 0
2 freeslip
3 velocity 1
4 pressure 0
```

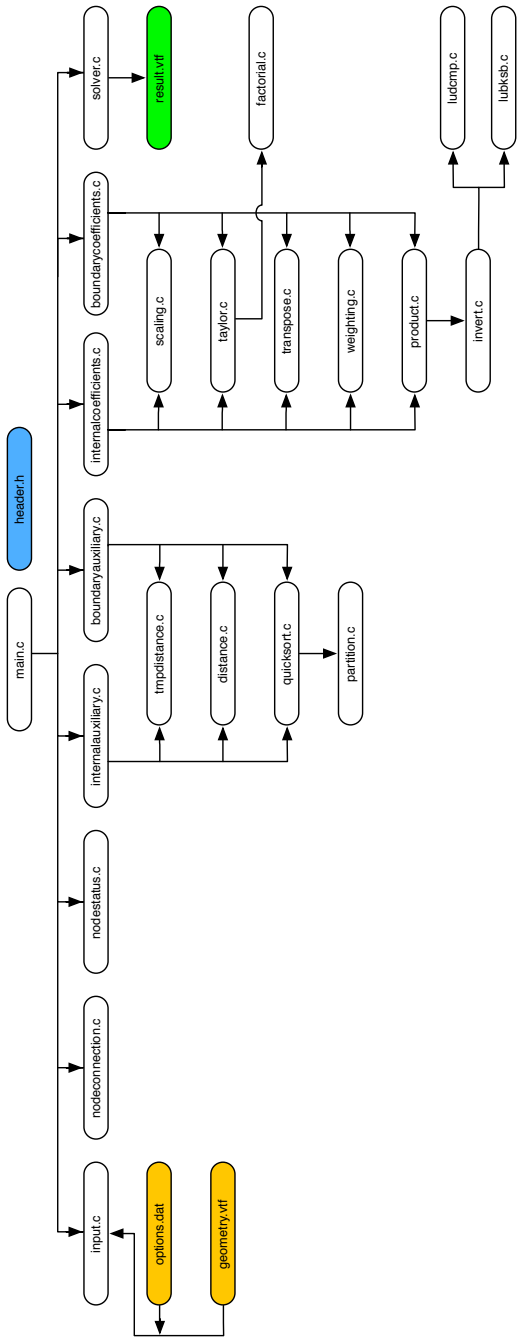
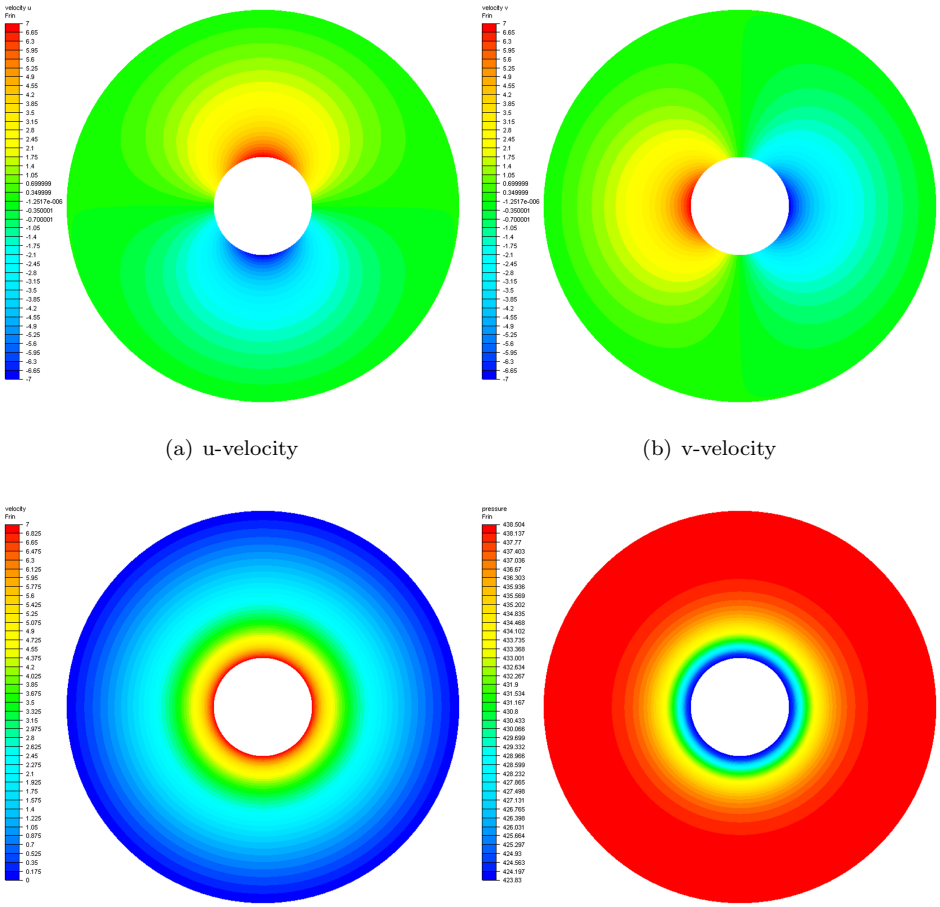


Figure A.1: Program flow chart

Appendix B

Rotating Couette flow



(a) u-velocity

(b) v-velocity

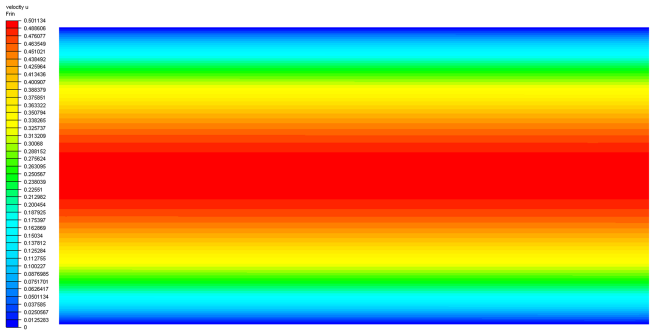
(c) Total velocity

(d) pressure

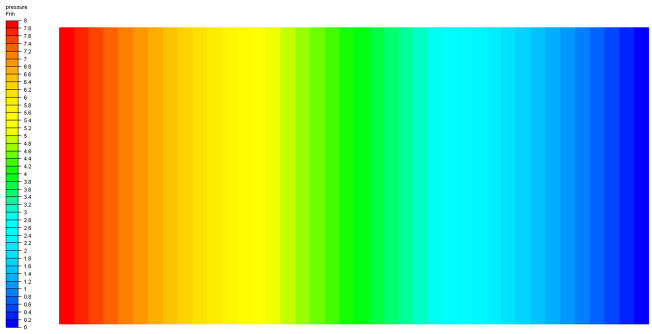
Figure B.1: Velocity field for Couette flow problem

Appendix C

Poiseuille channel flow



(a) u-velocity

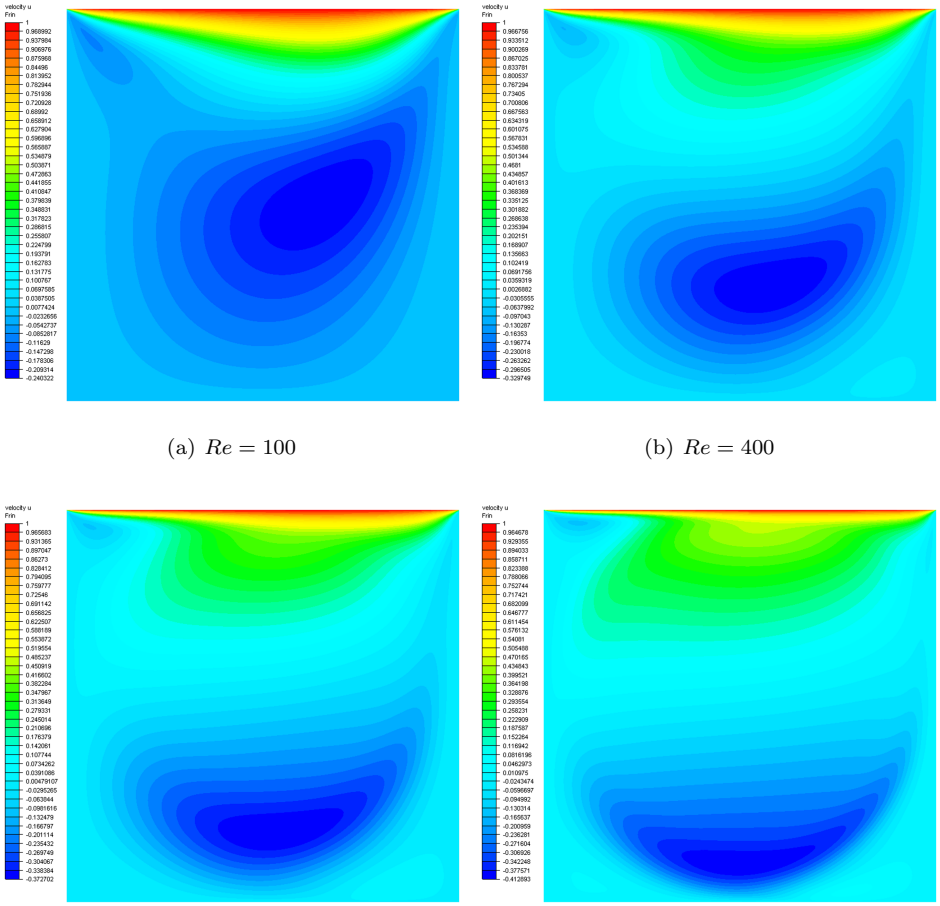


(b) pressure

Figure C.1: Velocity– and pressure field for Poiseuille flow problem

Appendix D

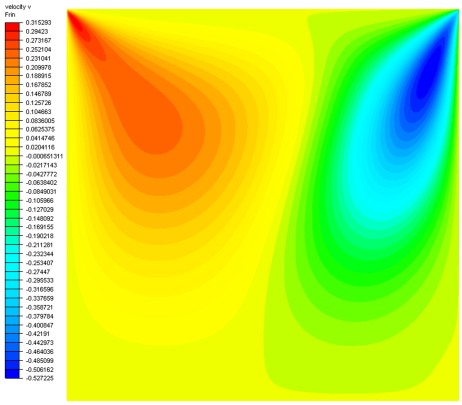
Lid-driven cavity flow



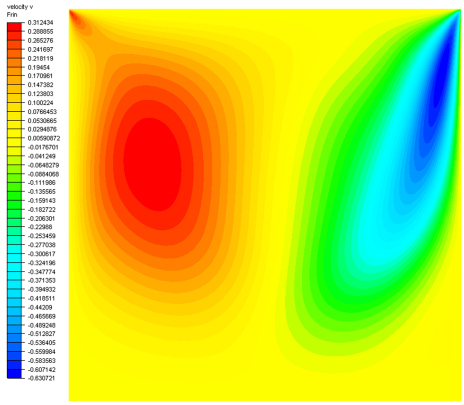
(a) $Re = 100$ (b) $Re = 400$

(c) $Re = 1000$ (d) $Re = 3200$

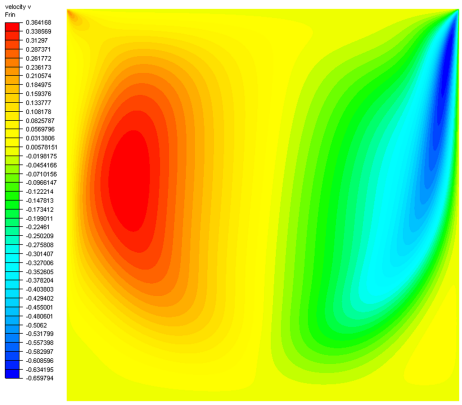
Figure D.1: u-velocity fields for lid-driven cavity flow problem



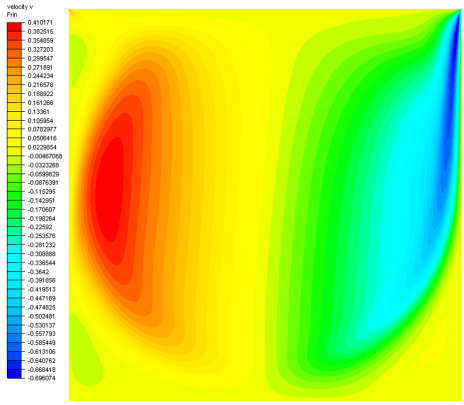
(a) $Re = 100$



(b) $Re = 400$

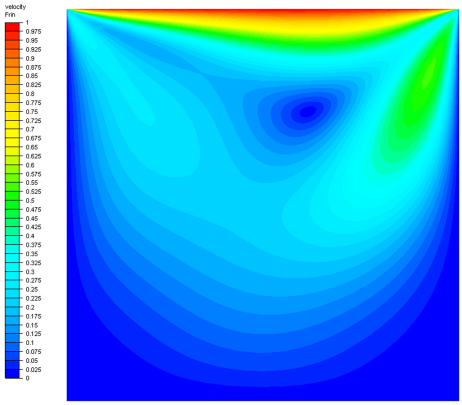


(c) $Re = 1000$

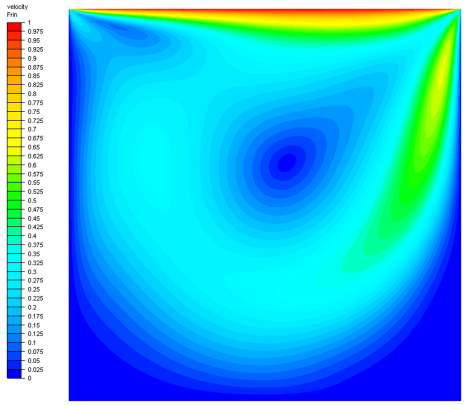


(d) $Re = 3200$

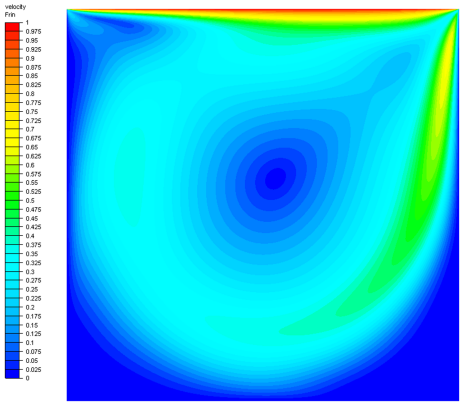
Figure D.2: v-velocity fields for lid-driven cavity flow problem



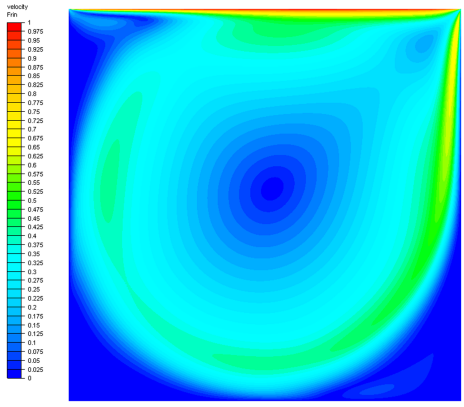
(a) $Re = 100$



(b) $Re = 400$

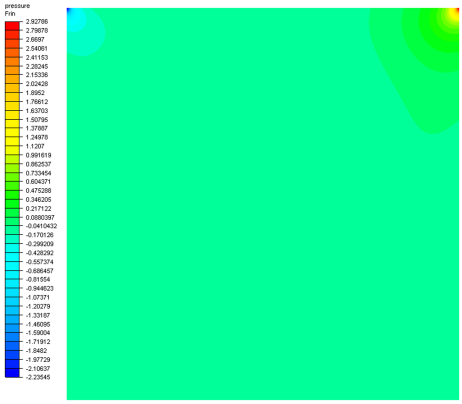


(c) $Re = 1000$

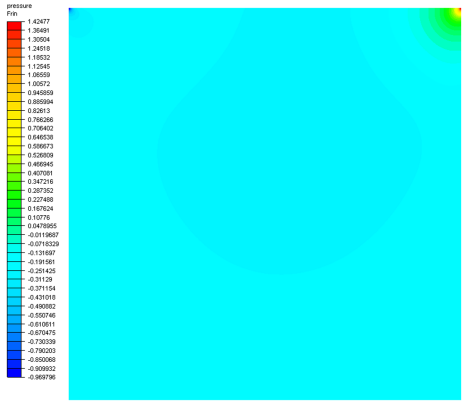


(d) $Re = 3200$

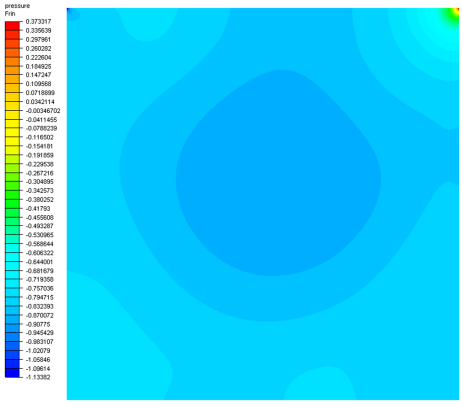
Figure D.3: Total velocity fields for lid-driven cavity flow problem



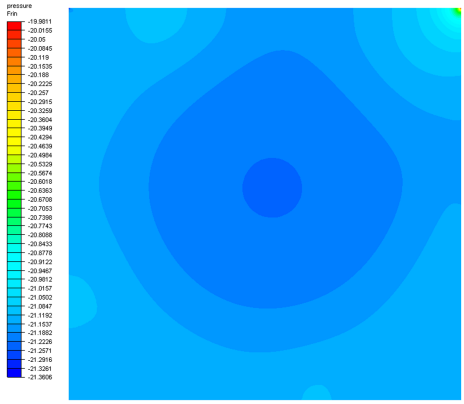
(a) $Re = 100$



(b) $Re = 400$



(c) $Re = 1000$

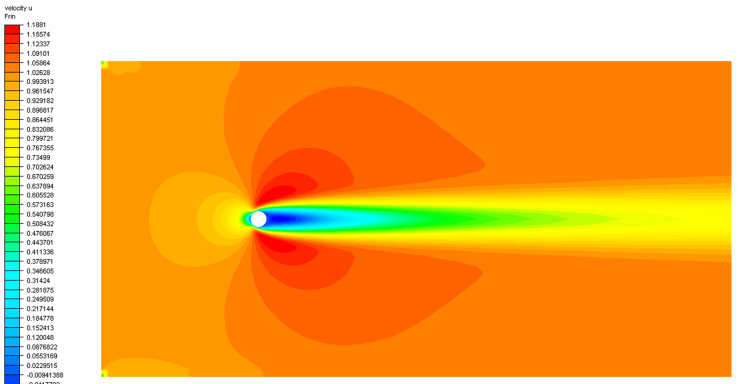


(d) $Re = 3200$

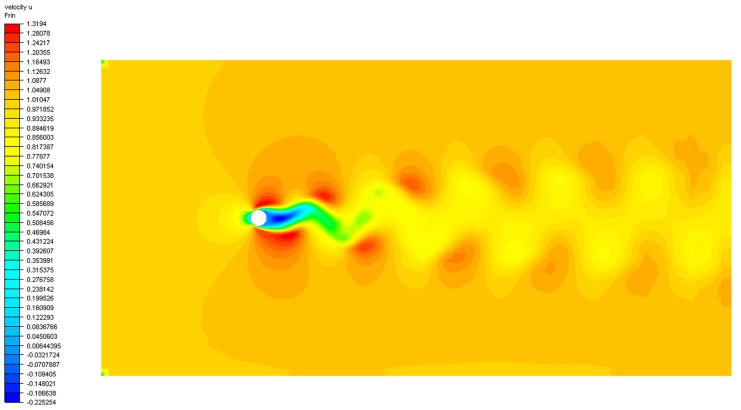
Figure D.4: Pressure fields for lid-driven cavity flow problem

Appendix E

Flow past a circular cylinder

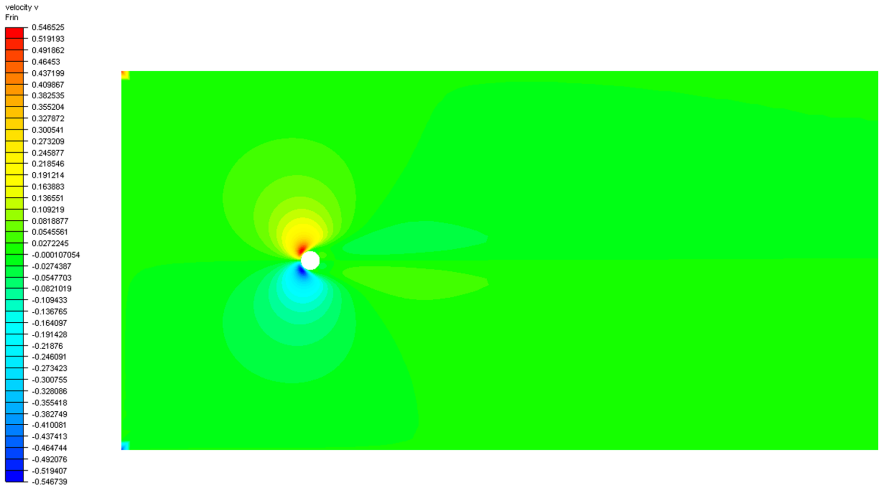


(a) $Re = 40$

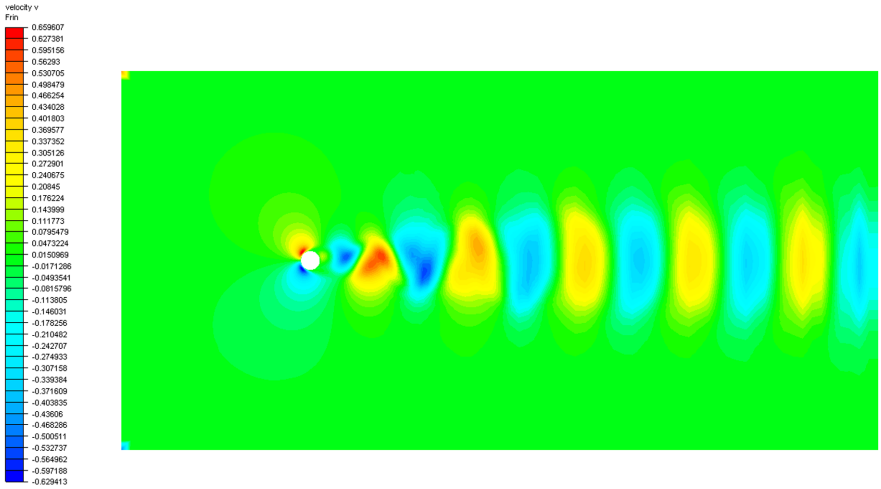


(b) $Re = 100$

Figure E.1: u-velocity fields for the viscous cylinder flow

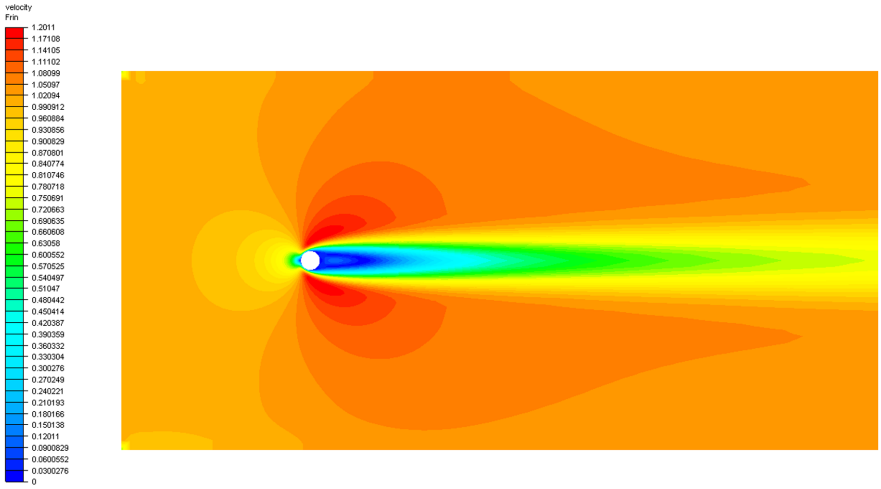


(a) $Re = 40$

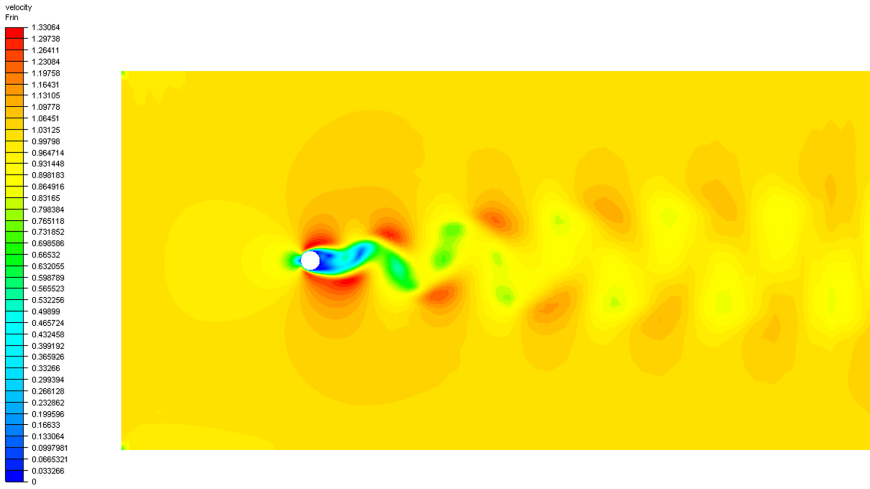


(b) $Re = 100$

Figure E.2: v-velocity fields for the viscous cylinder flow

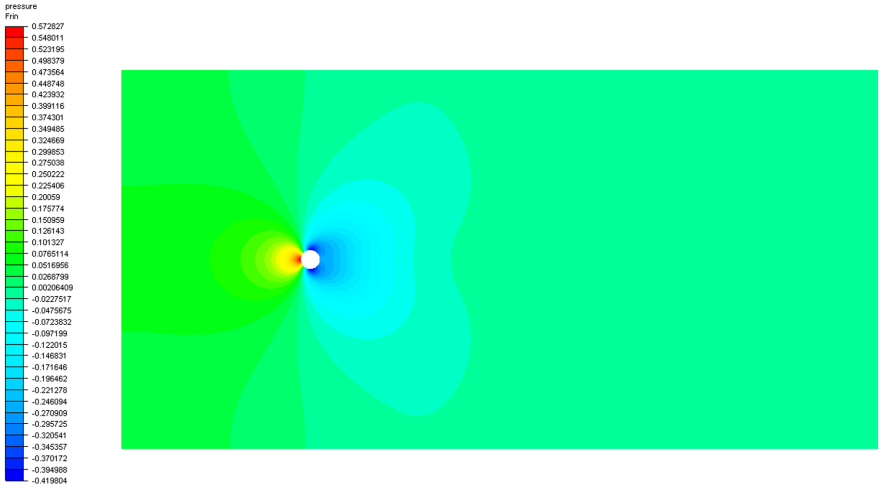


(a) $Re = 40$

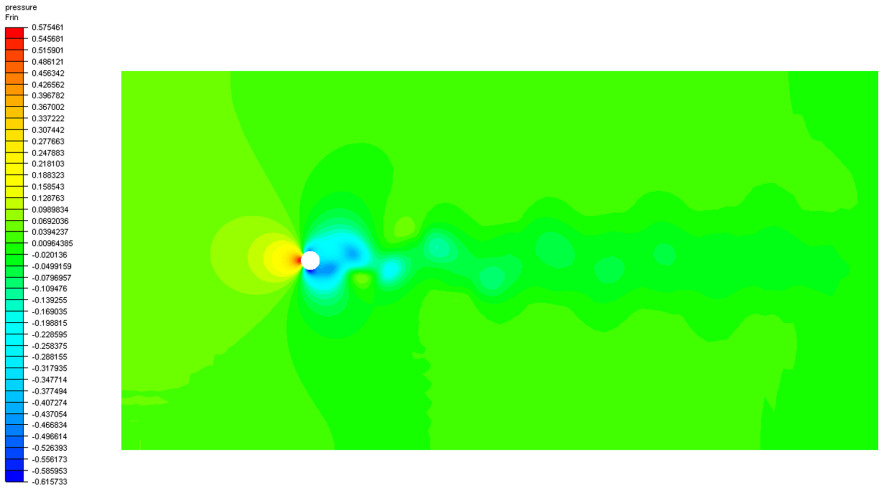


(b) $Re = 100$

Figure E.3: Total velocity fields for the viscous cylinder flow



(a) $Re = 40$



(b) $Re = 100$

Figure E.4: Pressure fields for the viscous cylinder flow