

Discontinuous - Galerkin method for direct numerical simulation of the Navier - Stokes equation: Master Thesis Report

Nirav Vasant Shah,
M.Sc. Candidate, Water Resources Engineering and Management,
Universität Stuttgart,
Stuttgart, Deutschland (Germany)

Supervisor: Prof. Dr. Bernard Haasdonk,
Institute of Applied Analysis and Numerical Simulation,
Universität Stuttgart,
Stuttgart, Deutschland (Germany)

Coadvisor: Prof. Dr. Gianluigi Rozza,
SISSA, Scuola Internazionale Superiore di Studi Avanzati,
Mathematics area, mathlab,
Trieste, Italy

Coadvisor: Dr. Martin Hess,
SISSA, Scuola Internazionale Superiore di Studi Avanzati,
Mathematics area, mathlab,
Trieste, Italy

March 16, 2018

0.1 Acknowledgement

This thesis is a result of the work carried out between September 2017 - March 2018 at the Institute of Applied Analysis and Numerical Simulation, Universität Stuttgart and at the mathematics area, mathlab, SISSA, Scuola Internazionale Superiore di Studi Avanzati, Trieste.

Undoubtedly, this result could only be achieved by the support of a number of people to whom I would like to express my gratitude.

To my supervisor Professor Dr. Bernard Haasdonk: I experienced the freedom that was necessary needed to be creative and continuous support which helped to continue research in case of difficulties. I am thankful for assistance for short term study mission application as home supervisor.

To my coadvisors Dr. Martin Hess and Professor Dr. Gianluigi Rozza: I am thankful for their positive coordination and helping me to develop necessary skills in the field of work. I also thank them for hosting me as visiting student as part of short term study mission.

To COST organisation : I am very much thankful for acceptance of my application for the award of short term study mission grant. The trust shown by them in my skills has been of great importance.

To my colleagues: I would also thank colleagues Mrs. Anne Weiß, Mr. Andreas Schmidt, Mr. Francesco Ballarin, Mr. Matteo Zancanaro, Ms. Maria Strazzullo and Mr. Giacomo Zuccarino for their positive interest in my work and valuable support which created highly conducive environment.

I am really looking forward to continue these valuable working relationships.

To my parents and brother: Last but most importantly, I would like to thank my parents for their continuous encouragement, infusion of new energy and especially support during critical and difficult times.

0.2 Declaration/Eigenständigkeitserklärung

I hereby certify that I have prepared this masters thesis independently, and that only those sources, aids and advisors that are duly noted herein have been used and / or consulted.

Ich versichere hiermit, dass

1. ich meine Arbeit selbständig verfasst habe,
2. ich keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe,
3. die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist,
4. ich die Arbeit weder vollständig noch in Teilen bereits veröffentlicht habe und
5. das elektronische Exemplar mit den anderen Exemplaren übereinstimmt.

Signature/Unterschrift

Place, Date/Ort, Datum

0.3 List of symbols

Following is the list of symbols used throughout the thesis with their respective meaning.

Symbol/Abbreviation	Description
CFD	Computational Fluid Dynamics
Ω	Continuous domain
$\partial\Omega$	Continuous domain boundary
Γ_D	Dirichlet boundary
Γ_N	Neumann boundary
B'	Extensive property under consideration
b'	Intensive property corresponding to B'
t	Neumann value
t'	time
cv	Control volume
cs	Control system
ρ	Density of fluid
u	Velocity
p	Pressure
M	Momentum of fluid flowing through control volume
f	Source/Sink/External force per unit volume
F	External force acting on cv
σ	Stress
ν	Kinematic viscosity
u_D	Velocity at Dirichlet boundary
n	Unit normal vector pointing outward from element
DNS	Direct Numerical Simulation
∇^s	Symmetric tensor, $\frac{1}{2}(\nabla + \nabla^T)$
Re	Reynolds number
L	Characteristic length for Reynolds number
\mathcal{T}	Grid or discretised domain
$\partial\mathcal{T}$	Grid boundaries
Γ	Inter-element or internal boundary of grid
nel	Total number of elements of grid
k	Index of an element, $1 \leq k \leq nel$
n^+	Unit normal vector pointing from element itself to neighbouring element
n^-	Unit normal vector pointing from neighbouring element to element itself
τ_k	k^{th} Triangular element, $\tau_k \in \mathcal{T}$
h_{τ_k}	Diameter of τ_k
θ_k	Smallest angle of τ_k
r	Coordinates of point in barycentric coordinate system
r_1, r_2, r_3	Coordinates of vertices of triangle

$\lambda_1, \lambda_2, \lambda_3$	Weights in barycentric coordinate system (Equation 2.1)
\mathbb{V}	Function space for velocity
\mathbb{Q}	Function space for pressure
P^D	Polynomial of degree D
D	Polynomial degree
ϕ	Velocity basis function in global space
ψ	Pressure basis function in global space
$\hat{\phi}$	Velocity basis function in local space
$\hat{\psi}$	Pressure basis function in local space
N	Truth space dimension
F_k	Mapping from local coordinate system to global coordinate system (Equation (2.10))
\hat{T}	Reference triangle
J_k	Jacobian of τ_k (Equation (2.10))
\hat{X}	Coordinates of point in local coordinate system, $\hat{X} \in \mathbb{R}^d$
X	Coordinates of point in global coordinate system, $X \in \mathbb{R}^d$
C	Translational vector for local to global mapping (Equation (2.10))
g	Function in global space
\hat{g}	Function in local space
g^{up}	Upwind value of function g
det	Determinant of matrix
$\hat{\Gamma}$	An edge of reference Triangle \hat{T}
JIT_k	Jacobian inverse transpose corresponding to τ_k , J_k^{-1}
C_{11}	Penalty parameter
a_{IP}	Term representing poisson operator of the strong form in the weak formulation of the Navier Stoks equation (Equation (2.60))
b	Term representing gradient operator in the weak formulation of the Navier Stoks equation (Equation (2.60))
c	Term representing non linear terms in the weak formulation of the Navier Stoks equation (Equation (2.60))
A	Matrix terms corresponding to a_{IP}
B	Matrix terms in corresponding to b
C	Matrix terms in corresponding to c
l_{IP}	Term representing right hand side of strong form in weak formulation of the Navier Stoks equation (Equation (2.60))
u^{ext}	External trace of velocity (Equation (2.59))
l	Length of an edge on \mathcal{T}
U	Velocity solution vector
P	Pressure solution vector

F_1	Discrete form of right hand side of equation (2.60)
F_2	Discrete form of right hand side of equation (2.36)
u_{ndofs}	Total number of degrees of freedom of velocity
p_{ndofs}	Total number of degrees of freedom of pressure
$bicgstab$	Biconjugate gradients stabilized method
$minres$	Minimum residual method
z	Non zero vector
$\{\cdot\}$	Average operator
$[\cdot]$	Jump operator
(\cdot, \cdot)	L^2 scalar product
S	Schur complement
I	Identity matrix
u_{npe}	Number of degrees of freedom per element for velocity
p_{npe}	Number of degrees of freedom per element for pressure
d	Dimension of problem
κ_e	Coercivity constant
k'	Sobolev space order
w_i	Weight in Gaussian quadrature rule
nop	Number of points

List of Figures

2.1	Element self (+) and neighbouring element (-)	22
2.2	Continuous domain (left) and discretised domain or grid (right)	23
2.3	Finite Element nodes on triangle for polynomials of different degrees	25
2.4	Global geometry (left) to Local geometry (right)	26
3.1	Sparsity pattern of constituents of $([n \otimes \phi], [n \otimes \phi])_{\Gamma \cup \Gamma_D}$	55
3.2	Sparsity pattern of constituents of $(\{\nabla \phi\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D}$	56
3.3	Sparsity pattern of constituents of $(\{\psi\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}$	57
3.4	Sparsity pattern of $(\psi, \nabla \cdot \phi)$	58
3.5	Sparsity pattern of $((u_k \cdot \nabla) \phi, \phi)$	58
3.6	Sparsity pattern of $((u_k \cdot n) \phi, \phi)_{\Gamma_N}$	59
3.7	Sparsity pattern of $(\nabla \phi, \nabla \phi)$	59
3.8	Sparsity pattern of constituents of $((u_k \cdot n) \phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + ((u_k \cdot n) \phi, \phi)_{\partial T \setminus \Gamma_N}$	60
3.9	Sparsity pattern of constituents of $(abs(u_k \cdot n) \phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + (abs(u_k \cdot n) \phi, \phi)_{\partial T \setminus \Gamma_N}$	61
4.2	h -convergence in L^2 norm for the Stokes flow	67
4.4	h -convergence in H_0 norm for the Stokes flow	69
4.5	p -convergence for the Stokes flow	70
4.6	Lid driven cavity problem (<i>bicgstab</i> solver)	71
4.7	Lid driven cavity problem (<i>minres</i> solver)	72
4.8	Lid driven cavity problem (Schur complement method)	73
4.9	Flow over cylinder (<i>bicgstab</i> solver)	74
4.10	Flow over cylinder (<i>minres</i> solver)	75
4.11	Flow over cylinder (Schur complement method)	76
4.12	Effect of penalty parameter on condition number of the stiffness matrix	77
4.13	Flow over cylinder for smaller than minimum allowable penalty parameter	78
4.14	h -convergence for the Navier Stokes flow in L^2 error	80
4.15	h -convergence for the Navier Stokes flow in L^2 error (Logarithmic scale)	81
4.16	h -convergence for the Navier Stokes flow in H_0 error	82
4.17	h -convergence for the Navier Stokes flow in H_0 error (Logarithmic scale)	83
4.18	Lid driven cavity flow (Initial guess by <i>bicgstab</i> solver)	84
4.19	Lid driven cavity flow (Initial guess by <i>minres</i> solver)	85

4.20 Lid driven cavity flow (Initial guess by Schur complement method)	86
4.21 Flow over cylinder (Initial guess by <i>bicgstab</i> solver)	87
4.22 Flow over cylinder (Initial guess by <i>minres</i> solver)	88
4.23 Flow over cylinder (Initial guess by Schur complement method)	89

List of Tables

3.1	Fields of <code>params</code>	40
3.2	Fields of <code>paramsP</code>	40
3.3	Fields of <code>grid</code>	41
3.4	Some other variables	41
3.5	Size and sparsity pattern of different terms	54

Contents

0.1 Acknowledgement	2
0.2 Declaration/Eigenständigkeitserklärung	3
0.3 List of symbols	4
1 Perspective and formulation	17
1.1 Derivation	17
1.1.1 DNS	19
2 Discretisation and functional spaces	21
2.1 Grid geometry	21
2.2 Grid parameters	22
2.3 Discontinuous-Galerkin (DG) method	23
2.4 Basis function	24
2.4.1 Nodal basis functions	24
2.4.2 Orthonormal basis functions	25
2.5 Global and local coordinate system	26
2.6 Jump operator	27
2.7 Average operator	28
2.8 L^2 scalar product	28
2.9 Problem statement	28
2.9.1 Stokes strong, weak and discrete form	28
2.9.2 Properties of the stiffness matrix	30
2.9.3 Upwinding	33
2.9.4 Navier Stokes strong, weak and discrete form	33
2.9.5 Newton method	34
2.9.6 Properties of the stiffness matrix	36
2.9.7 Boundary conditions	36
2.10 Selection of solver	36
2.10.1 Biconjugate gradients stabilized method	36
2.10.2 Minimum residual method	37
2.10.3 Schur complement method	37
3 Implementation aspects	39
3.1 Terminology	39
3.2 Basis functions in RBmatlab	39
3.3 Assembly of average operator	42
3.4 Assembly of jump operator	42
3.5 Matrix assemblies	43

3.5.1	Assembly of $(\nabla\phi, \nabla\phi)$	43
3.5.2	Assembly of $([n \otimes \phi], [n \otimes \phi])_{\Gamma \cup \Gamma_D}$	44
3.5.3	Assembly of $(\{\nabla\phi\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D}$	45
3.5.4	Assembly of $(\{\psi\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}$	45
3.5.5	Assembly of $-\int_{\hat{T}} \psi \nabla \cdot \phi$	46
3.5.6	Assembly of $(t, \phi)_{\Gamma_N}$	47
3.5.7	Assembly of $(u_D, \phi)_{\Gamma_D}$	47
3.5.8	Assembly of $(\psi, n \cdot u_D)_{\Gamma_D}$	48
3.5.9	Assembly of (f, ϕ)	48
3.5.10	Assembly of $(n \otimes u_D, \nabla\phi)_{\Gamma_D}$	49
3.5.11	Assembly of $-((u_k \cdot \nabla)\phi, \phi)$	49
3.5.12	Assembly of $((u_k \cdot n)\phi, \phi)_{\Gamma_N}$	50
3.5.13	Assembly of $((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N}$	50
3.5.14	Assembly of $(u_k \cdot n \phi, \phi^{ext})_{\partial T \setminus \Gamma_N}$	51
3.5.15	Assembly of $((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$	51
3.5.16	Assembly of $((u_k \cdot n \phi, \phi)_{\partial T \setminus \Gamma_N}$	51
3.6	Setting boundary conditions	52
3.7	Setting source term	52
3.8	Program flow	52
3.8.1	Grid preparation	52
3.8.2	Function space formulation	53
3.8.3	Matrix assembly	53
3.8.4	Solving assembled form	53
3.8.5	Post processing	53
3.8.6	Newton method	54
3.9	Sparsity pattern	54
4	Numerical results	63
4.1	Error definitions	63
4.2	Stokes flow	64
4.2.1	Properties of the stiffness matrix	64
4.2.2	Analytical example	64
4.2.3	Lid-driven cavity problem	65
4.2.4	Flow over cylinder	65
4.3	Penalty parameter	65
4.4	Navier-Stokes flow	65
4.4.1	Analytical example	79
4.4.2	Lid-driven cavity problem	79
4.4.3	Flow over cylinder	79
4.5	Solver selection	90
5	Summary, conclusion and outlook	91
5.1	Conclusions	91
5.2	Outlook	92
Appendices		95

CONTENTS	13
A Mathematical preliminaries	95
A.0.1 Cholesky decomposition	95
A.0.2 Saddle point formulation	96
A.0.3 Sobolev spaces	97
A.0.4 Basic definitions	98
A.0.5 Linear forms	99
A.0.6 Bilinear forms	99
A.0.7 Condition number	100
A.0.8 Important inequalities	100
B Online resources	101
C Code access	103

Introduction

The thesis deals with the numerical simulation of the Navier Stokes equations (White F.M. [8]) which are the core of Computational Fluid Dynamics or (*CFD*). The thesis performs numerical simulation of the Navier-Stokes equations through the discontinuous - Galerkin (DG) method.

The goal of the thesis is to derive, discretise and implement the discontinuous Galerkin (Rivière B. [6]) weak form of the Navier-Stokes equations and perform numerical simulation of the Navier-Stokes equations. During the numerical solution we measure the computational efforts. This includes solver performance such as run time as well as accuracy.

In Chapter 1 the Navier Stokes equations are introduced. We first derive the Navier-Stokes equations from the conservation equation or the Reynolds transport theorem. We discuss the flow classification and important issues related to numerical simulation of the Navier-Stokes equations.

In Chapter 2 we bring the infinite dimensional problem to a finite dimensional problem by introducing discretisation. We first introduce grid, constituents of elements and transformation between local and global geometry. Further the function spaces over the grid are discussed in which, the basis functions or the Ansatz functions, function spaces for pressure and velocity are introduced. We further introduce and discuss the weak form of the Navier-Stokes equation, discrete form of the Navier-Stokes equation with boundary conditions and properties of the weak form.

In Chapter 3 we discuss the implementation of the discrete form of the Navier-Stokes equations in RBmatlab, an open-source software developed at the University of Stuttgart and at the University of Münster, for numerical simulation. In this chapter we discuss evaluation of basis functions in RBmatlab, matrix assembly, dimension of assembled matrices and boundary condition implementation. We further discuss the Schur complement method and the Newton method for nonlinearity.

In Chapter 4 we present results from numerical experiments, analyse results from benchmark problems, evaluate solver performance and draw related conclusions.

In Chapter 5 we summarise the conclusions from numerical experiment results and also point towards the possibilities for further developments.

In appendix, we present mathematical preliminaries. Familiarity with concepts presented under this chapter is very helpful for the understanding of the thesis. Readers familiar with the topics can skip this section. Additionally we also provide references to published literature, online resources and implementation used for the current analysis for readers interested in exploring the presented topic further.

Chapter 1

Engineering perspective and mathematical formulation

The subject of mathematical applications in fluid mechanics starts with one of the variants of the Navier-Stokes equations. Almost all processes of fluid mechanics require considerations related to the Navier-Stokes equations. Hence the importance of the Navier-Stokes equations is impossible to be ignored as far as mathematical approaches in fluid mechanics are concerned. The numerical methods for the incompressible Navier-Stokes equations are simpler if compared to the numerical methods for the compressible Navier-Stokes equations. As the incompressible condition is imposed the state variables are constant. This in turn means that the equation of state no longer needs to be solved. Throughout the thesis we discuss only steady state incompressible Navier-Stokes equations and hence, Navier-Stokes equations refer hereafter to steady state incompressible Navier Stokes equation.

1.1 Derivation of the Navier-Stokes equations

Before deriving the Navier-Stokes equations we introduce some notations. The domain is denoted by $\Omega \subseteq \mathbb{R}^d$. The domain boundary is denoted by $\partial\Omega$. The domain boundary is divided into Dirichlet boundary Γ_D and Neumann boundary Γ_N i.e. $\Gamma_D \cup \Gamma_N = \partial\Omega$, $\Gamma_D \cap \Gamma_N = \emptyset$.

The governing equations for the incompressible Navier-Stokes flows are the conservation equations: Mass conservation and Momentum conservation. The conservation equations are derived based on the concept of control volume and the control surface. The control volume is the volume, fixed or moving with constant velocity in space, through which the fluid moves. Extensive quantity is the amount of a quantity contained in the system such as mass. Intensive quantity is the extensive quantity per unit mass of system. Intensive quantity is independent of the mass of the system and is not additive, while extensive quantity does depend on the mass of the system and is additive. The control surface is the surface enclosing the control volume. All equations can be derived from the Reynold's transport equation as presented for example by White F.M. [8]:

$$\frac{dB'}{dt'}|_{cs} = \frac{d}{dt'} \int_{cv} b' \rho dV + \int_{cs} (b' \rho) u \cdot dA \quad (1.1)$$

cv = Control volume,

cs = Control surface,

B' = Extensive quantity under consideration,

b' = Intensive quantity corresponding to B' ,

ρ = Density of fluid,

u = Velocity of fluid.

If in the above equation B' is substituted as momentum M and correspondingly b' as velocity u , we obtain the change in momentum. As per Newton's second law of motion change in momentum is equal to the sum of external forces acting on the system.

$$F = \frac{dM}{dt'} = \frac{d}{dt'} \int_{cv} u \rho dV + \int_{cs} (u \rho) u \cdot dA \quad . \quad (1.2)$$

This sum of forces arises from stresses σ (shear stresses and normal stresses) and external force f such as weight.

$$F = \int_{cs} \sigma \cdot dA + \int_{cv} \rho f dV \quad , \quad (1.3)$$

σ = Viscous stress,

f = External force per unit volume.

Equating external forces with change in momentum i.e. equating (1.2) and (1.3), considering steady conditions only, using definition of viscous stress tensor and with the application of the Gauss divergence theorem we arrive at the Navier Stokes equation,

$$-2\nabla \cdot (\nu \nabla^s u) + (1/\rho) \nabla p + (u \cdot \nabla) u = f \quad \text{in } \Omega. \quad (1.4)$$

The incompressible mass conservation equation can be derived similarly by substituting $B' = \text{Mass of control system}$ and $b' = 1$ in equation (1.1). It can be written as,

$$\nabla \cdot u = 0 \quad \text{in } \Omega. \quad (1.5)$$

The boundary conditions are expressed as,

Dirichlet boundary:

$$u = u_D \quad \text{on } \Gamma_D. \quad (1.6)$$

Neumann boundary:

$$-pn + 2\nu(n \cdot \nabla^s)u = t \quad \text{on } \Gamma_N. \quad (1.7)$$

u = flow velocity and $u : \Omega \rightarrow \mathbb{R}^d$,

p = pressure and $p : \Omega \rightarrow \mathbb{R}$,

ν = kinematic viscosity (fluid property) $\nu : \Omega \rightarrow \mathbb{R}$,

ρ = density (fluid property) $\rho : \Omega \rightarrow \mathbb{R}$,

f = external force $f : \Omega \rightarrow \mathbb{R}^d$,
 u_D = specified flow velocity at Dirichlet boundary $u_D : \Gamma_D \rightarrow \mathbb{R}^d$,
 n = normal unit vector $n : \partial\Omega \rightarrow \mathbb{R}^d$,
 t = specified Neumann flux $t : \Gamma_N \rightarrow \mathbb{R}^d$.

$$\nabla^s = \frac{1}{2}(\nabla + \nabla^T).$$

The equation (1.4) is known as the strong form of the momentum conservation of the Navier Stokes equation.

It can be seen that the steady state Navier Stokes equation is nonlinear and has two unknown variables, pressure p and velocity u . The additional equation, mass conservation equation, is hence necessary to obtain a sufficient number of equations for the number of unknowns.

We also introduce the dimensionless number Reynolds number, Re which is the most characteristic quantity of the flow. The Reynolds number is defined as the ratio of inertial force to the viscous force,

$$Re = uL/\nu. \quad (1.8)$$

Where, L is the characteristic geometrical dimension, such as the diameter of a pipe in case of pipe flow or the span of the wing of an aircraft in case of flow over an aircraft wing.

1.1.1 Direct numerical simulation

We now differentiate between the type of flows, laminar and turbulent.

Laminar flow is characterised by well defined velocity and pressure field and low Reynolds number. This flow has very low velocity fluctuations and pressure fluctuations. The viscous force is balanced by the pressure force and the flow has negligible inertial force. Mathematically, the non linear term in (1.4) is no longer present. This equation is known as Stokes equation (White F.M.[8]).

The strong form of the Stokes equation is as follow,

$$-\nu\Delta u + \nabla p = f \quad \text{in } \Omega, \quad (1.9)$$

$$u = u_D \quad \text{on } \Gamma_D, \quad (1.10)$$

$$-pn + \nu n \cdot \nabla u = t \quad \text{on } \Gamma_N. \quad (1.11)$$

Turbulent flow, in contrast, is characterised by fluctuations in velocity and pressure field and a high Reynolds number. The flow has high velocity and an inertial force is present in addition to a viscous and a pressure force. This inertial force makes the Equation (1.4) non linear. The fluctuations of velocity and pressure are

of the order of the Kolmogrov scale (Kundu et al. [13]).

The method used for solving Equation (1.4) numerically is known as Direct Numerical Simulation, abbreviated as *DNS*. The direct numerical simulation could be computationally very expensive especially in applications such as turbulent flows as the time and space grid size is of the order of the Kolmogrov scale but the time period or space dimension over which the simulation is carried out are very large. In order to avoid such computational expenses alternative models are used replacing the original model. However, the alternative model can not explain the flow physics completely. The prediction of accurate flow physics description requires economical numerical solution of (1.4). It is to be noted that the simulation of turbulent flow is always a choice of balance between accuracy of computation and computational efforts.

We now discuss, in next chapter, about the grid formation, deriving weak formulation from the strong formulation presented in this chapter and solution of the weak formulation.

Chapter 2

Discretisation and functional spaces

2.1 Grid geometry

In numerical analysis a continuous problem is posed over finite number of degrees of freedom. We divide the original domain into smaller subdomains to have finite number of degrees of freedom. The divided domain is called grid. If the original domain is denoted by Ω , we denote the grid by \mathcal{T} . In the present case, we use triangular grid and denote each triangle as τ_k with k as element index. If nel is the total number of elements in the grid, $1 \leq k \leq nel$. We note that $\mathcal{T} = \cup_{k=1}^{nel} \tau_k$. Each triangle is an 'Element' of the grid. The boundary between elements i.e. interelement boundary is denoted by Γ . In case of a grid, the boundary $\partial\mathcal{T}$ comprises of domain boundaries and interelement boundaries i.e. $\partial\mathcal{T} = \Gamma_D \cup \Gamma_N \cup \Gamma$. During discussion on jump operator and average operator we denote the element under consideration as τ_h^+ and neighbouring element as τ_h^- . (Figure 2.1)

We also denote the normal pointing from element itself towards neighbouring element as n^+ and the normal pointing from neighbouring element towards element itself as n^- . Correspondingly every quantity on element itself is denoted by superscript + and on neighbouring element is denoted by -. We denote by h_{τ_k} the diameter of element τ_k such that $h_{\tau_k} = \sup ||x - y||$ where, $(x, y) \in \tau_k$. We also denote by θ_k the smallest angle of the element τ_k .

In case of a 2-dimensional domain the grid could be a triangular grid or a rectangular grid. The triangular grids are useful for irregular geometry and also on regular geometry if the solution is expected to be irregular due to complex flow physics. This flexibility requires additional efforts to define the grid accurately. That is, unlike a structured grid, an unstructured grid needs to define connectivity of vertices, which form edges, which in turn form a face. In case of a 2-dimensional grid we have faces which are 2-dimensional entities, edges which are 1-dimensional entities and points or vertices which are 0-dimensional entities. In case of 3-dimensional grid, these faces constitute tetrahedral elements.

For triangular grids we also consider a barycentric coordinate system. In barycentric coordinate system any point r within a triangle is expressed in terms



Figure 2.1: Element self (+) and neighbouring element (-)

of vertices r_1, r_2, r_3 forming the triangle. This is represented as,

$$r = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3. \quad (2.1)$$

Here, $\lambda_1, \lambda_2, \lambda_3$ are weights. The weights satisfy the criterion,

$$\lambda_1 + \lambda_2 + \lambda_3 = 1. \quad (2.2)$$

Hence, we only need to specify 2 values in 2-dimensional plane in order to fully define the position of point.

For example, the centroid of triangle will have $\lambda_1 = 1/3, \lambda_2 = 1/3$. By equation (2.2) we have $\lambda_3 = 1/3$.

2.2 Grid parameters

We refer to 'Grid parameters' as the geometrical parameters which are dependent on the geometry of the problem or the grid or both. These parameters do not depend upon the mathematical formulation but are supplementary to the mathematical formulation. On the triangular grid we have 3 entities: faces, edges, vertices as explained above. From faces we have the area (equivalent to volume of element in case of a 3-dimensional grid) and the Jacobian. As explained later in the weak form of the Stokes and Navier-Stokes Discontinuous-Galerkin formulation and transformation between local and global geometry, the area of element is useful for volume integral terms and the Jacobian is useful for transformation between local and global geometry. From edges we derive the edge length l which is useful for boundary integral terms and normal vector n which is useful for flux calculation. The normal vector is the unit vector normal to the edge pointing outward from the element. Every element has 3 neighbouring elements and the element shares each

of his edge with one of its neighbour. From vertices we derive the vertex index which helps to define the connectivity of the vertices which is useful especially in case of unstructured grid. In order to give clear visualization of continuous domain and grid, we refer to Figure 2.2.

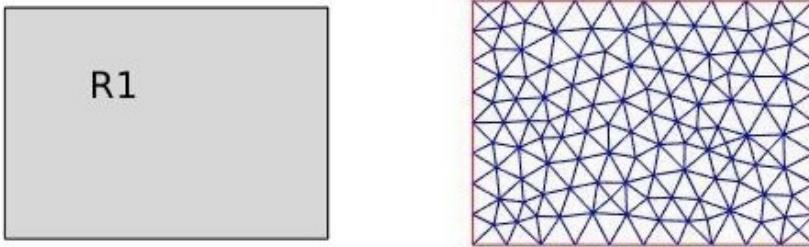


Figure 2.2: Continuous domain (left) and discretised domain or grid (right)

2.3 Discontinuous-Galerkin (DG) method

In the context of the discontinuous Galerkin method we introduce the function spaces $V(\mathcal{T})$ and $Q(\mathcal{T})$ for analytical solution of velocity and analytical solution of pressure respectively. The space containing high fidelity solution (in this case discontinuous Galerkin) is called truth space denoted by \mathbb{V} for velocity and \mathbb{Q} for pressure. d_u and d_p represent the dimension of velocity and pressure respectively.

$$\mathbb{V} = \{\phi \in (L^2(\mathcal{T}))^{d_u} \mid \phi \in (P^D(\tau_k))^{d_u} \quad \forall \quad \tau_k \in \mathcal{T}\} \quad (2.3)$$

$$\mathbb{Q} = \{\psi \in (L^2(\mathcal{T}))^{d_p} \mid \psi \in (P^{D-1}(\tau_k))^{d_p} \quad \forall \quad \tau_k \in \mathcal{T}\} \quad (2.4)$$

Here, $P^D(\tau_k)$ denotes space of polynomials of degree at most D over τ_k .

We apply a similar procedure as in the finite element method i.e. multiplying the partial differential equation by a test function and integration by parts (Section 2.9.1). However, we note that our test function is not continuous on the interface. Hence, we require flux approximations and jumps at the interface. These requirements have given rise to different discontinuous Galerkin methods. For explanation of each method we refer to literatures such as by Persson et al. [14] for local discontinuous Galerkin method and by Montlaur et al. [2] for the Compact discontinuous Galerkin method and the Interior penalty method.

Discontinuous-Galerkin methods for the Navier Stokes equation were compared by Montlaur et al. [2]. The local discontinuous Galerkin (LDG) method extends

the computational stencil beyond immediate neighbours whereas compact discontinuous Galerkin (CDG) and interior penalty method (IPM) only connect to neighbouring elements. The CDG method provides more flexibility with respect to the stabilisation constant at the cost of additional simulation effort related to computation of the lifting operator, while the IPM method requires restrictions on penalty parameter in order to maintain coercivity of bilinear form. However, implementation of IPM is simpler as compared to implementation of CDG. Both methods, CDG and IPM, have almost similar convergence rates.

2.4 Nodal basis function and Orthonormal basis function

The "Basis functions" are also known as "Ansatz functions". There are two kinds of basis function, nodal basis function and orthonormal basis function, which are used in the application of continuous Galerkin method or discontinuous Galerkin method.

The number of degrees of freedom per element npe can be calculated as,

$$u_{npe} = d_u \frac{(D+1)(D+2)}{2} \quad \text{for velocity,} \quad (2.5)$$

$$p_{npe} = d_p \frac{(D)(D+1)}{2} \quad \text{for pressure.} \quad (2.6)$$

2.4.1 Nodal basis functions

Nodal basis functions are also known as "Shape function". Such a basis function has value of 1 at its respective node and 0 at other nodes. At all other points it is approximated based on the degree of the basis function.

In order to define a polynomial of given degree completely, we need to calculate its co-efficients. A polynomial of degree D in a 1-dimensional domain has $D+1$ coefficients. In case of a 2-dimensional domain the number of coefficients become $(D+1)(D+2)/2$. To define these coefficients the known values of function at a number of points equal to the number of coefficients is required. These points are known as nodes. In case of triangular element, the nodes are located as,

1. For polynomials of degree 1 i.e. $D = 1$, the nodes are located at vertices of an element.
2. For polynomials of degree 2 i.e. $D = 2$, the nodes are located at the vertices of an element and mid points of edges.

For higher order polynomials the nodes are located as shown in Figure 2.3. (Source:http://hplgit.github.io/INF5620/doc/pub/sphinx-fem/_main_fem009.html)

For example, Nodal basis function of degree $D = 1$ in 1 dimensional domain can be represented as (To higher dimension and higher polynomial degree, the definition can be extended similarly),

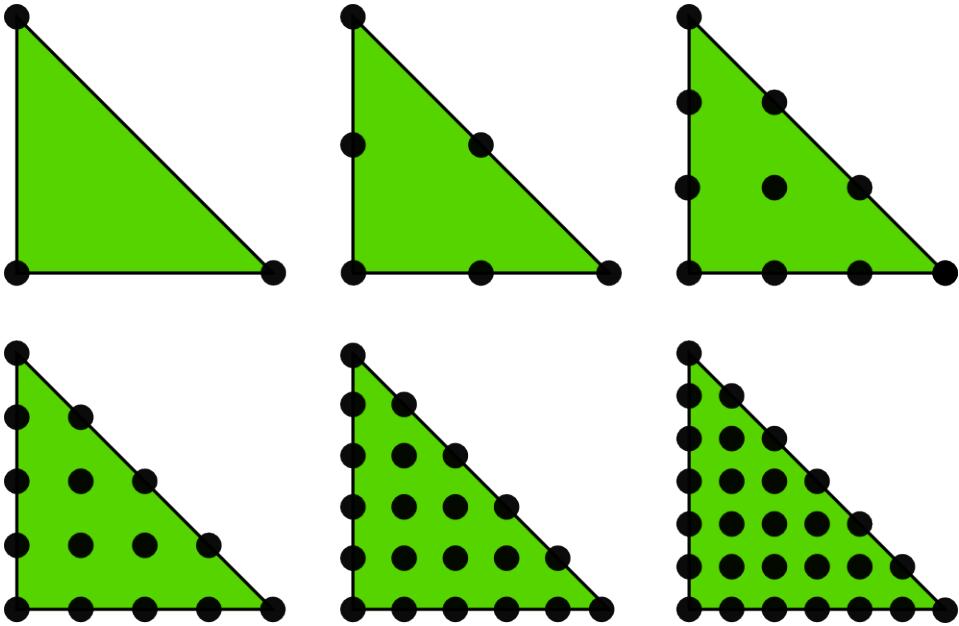


Figure 2.3: Finite Element nodes on triangle for polynomials of different degrees

$$\begin{aligned}\hat{\phi}_i &= \frac{x - x_{i-1}}{x_i - x_{i-1}} \quad \text{for } x_{i-1} \leq x \leq x_i, \\ \hat{\phi}_i &= \frac{x_{i+1} - x}{x_{i+1} - x_i} \quad \text{for } x_i \leq x \leq x_{i+1}, \\ \hat{\phi}_i &= 0 \text{ else.}\end{aligned}\tag{2.7}$$

We do not use Nodal basis function in our analysis, instead we use orthonormal basis function.

2.4.2 Orthonormal basis functions

Orthonormal basis functions are the basis functions defined in such a way that all basis functions are orthonormal to each other with respect to suitable inner product. The number of orthonormal basis functions for a given element is the same as the number of nodal basis functions.

$$\begin{aligned}(\hat{\phi}_i, \hat{\phi}_j) &= \int_{\hat{T}} \hat{\phi}_i \hat{\phi}_j = 1 \quad \text{if } i = j, \\ (\hat{\phi}_i, \hat{\phi}_j) &= \int_{\hat{T}} \hat{\phi}_i \hat{\phi}_j = 0 \quad \text{if } i \neq j.\end{aligned}\tag{2.8}$$

In the present analysis, (\cdot, \cdot) represents L^2 scalar product and \hat{T} is reference triangle (Section 2.5).

2.5 Global and local coordinate system

Integral terms are evaluated on a reference triangle instead of the element itself. Accordingly, a coordinate transformation between reference triangle and element is performed for evaluating integrals. The coordinate system in which the reference triangle lies is called reference or local coordinate system and the coordinate system in which element itself lies is called global coordinate system. The reference triangle has vertices $(0, 0), (1, 0), (0, 1) \in \mathbb{R}^2$ in order. The element is defined by vertex indices in order forming triangle. The transformation from local coordinate \hat{X} to global coordinate X is defined by the mapping,

$$F_k : \hat{X} \mapsto X \quad \forall \quad \hat{X} \in \hat{T} \quad \text{and} \quad X \in \mathcal{T}. \quad (2.9)$$

This mapping function is defined as,

$$F_k(\hat{X}) : X = J_k \hat{X} + C. \quad (2.10)$$

Here,

J_k = Jacobian matrix of element τ_k for transformation from local coordinate system to global coordinate system, $J_k \in \mathbb{R}^{d \times d}$,

C = Translational vector for transformation from local coordinate system to global coordinate system, $C \in \mathbb{R}^d$,

X = Coordinates of a point in Global coordinate system, $X \in \mathcal{T}$,

\hat{X} = Coordinates of a point in local coordinate system, $\hat{X} \in \hat{T}$.

We represent the image of a global function space or grid constituent on reference triangle by superscript $\hat{\cdot}$. This function space is known as local basis function space.

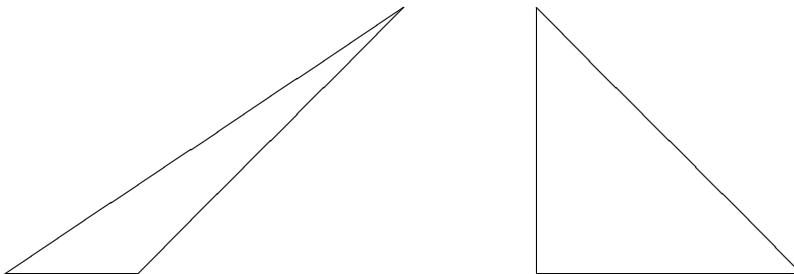


Figure 2.4: Global geometry (left) to Local geometry (right)

The volume integral of a function $g(x)$ in global coordinates is related to volume integral on reference geometry as,

$$\int_{\Omega} g(x) dx = \sum_{k=1}^{nel} \int_{\tau_k} g(x) dx = \sum_{k=1}^{nel} \int_{\hat{T}} \hat{g}(\hat{x}) |\det(J_k)| d\hat{x}. \quad (2.11)$$

The linear boundary integral of a function $g(x)$ on global coordinates is related to boundary integral on reference geometry as,

$$\int_{\Gamma} g(x) ds = \int_{\hat{\Gamma}} \hat{g}(\hat{x}) l d\hat{s}. \quad (2.12)$$

Also, the following holds,

$$\nabla g = JIT_k \hat{\nabla} \hat{g}, \quad (2.13)$$

Here,

l = length of an edge on \mathcal{T} ,

g = A function in global coordinate system, $g : \Omega \mapsto \mathbb{R}$,

\hat{g} = A function in local coordinate system corresponding to function g in global coordinate system, $\hat{g} : \hat{T} \mapsto \mathbb{R}$,

$JIT_k = J_k^{-1}$, Jacobian inverse transpose of element k .

Here g and \hat{g} satisfy,

$$g(x) = \hat{g}(\hat{x}) \quad \text{for } x = F_k(\hat{x}) \quad \text{according to equation (2.10)} . \quad (2.14)$$

2.6 Jump operator

The jump operator of a quantity $[u]$ at an internal boundary is defined as,

$$[u] = u^+ \cdot n^+ + u^- \cdot n^-, \quad (2.15)$$

where n is the unit normal to an edge of an element pointing outward from the element.

As pointed out by Montlaur et al. [1] this jump representation has two disadvantages.

1. The function space of the quantity itself and the function space of the jump are different that is, the jump of a vector is scalar and jump of a scalar is vector.
2. The use of this definition camouflages the presence of a normal.

To overcome these disadvantages Montlaur et al. [1] modified jump representation as below :

1. For scalar quantity p ,

$$\begin{aligned} [pn] &= p^+ n^+ + p^- n^- \quad \text{on } \Gamma, \\ [pn] &= pn \quad \text{on } \Gamma_D. \end{aligned} \quad (2.16)$$

2. For vector quantity u ($n \otimes u = u_i n_j$, $1 \leq i \leq d_u$, $1 \leq j \leq d$),

$$\begin{aligned} [n \otimes u] &= n^+ \otimes u^+ + n^- \otimes u^- \quad \text{on } \Gamma, \\ [n \otimes u] &= n \otimes u \quad \text{on } \Gamma_D, \\ &\quad \text{or} \\ [n \cdot u] &= n^+ \cdot u^+ + n^- \cdot u^- \quad \text{on } \Gamma, \\ [n \cdot u] &= n \cdot u \quad \text{on } \Gamma_D. \end{aligned} \quad (2.17)$$

As can be seen the quantity and its jump are now in same space i.e. jump of vector is vector and jump of scalar is scalar.

2.7 Average operator

The average operator is defined as :

$$\{u\} = \frac{u^+ + u^-}{2}. \quad (2.18)$$

As can be seen definition of average operator does not involve normal and hence, is simpler as compared to the Jump operator.

Also, u and $\{u\}$ are in same function space i.e. average of a vector is vector and average of scalar is scalar.

2.8 L^2 scalar product

We denote the L^2 scalar product of p and q by (p, q) which refers to,

If p and q are scalars,

$$(p, q) = \int_{\Omega} pq. \quad (2.19)$$

If p and q are vectors,

$$(p, q) = \int_{\Omega} p \cdot q. \quad (2.20)$$

If p and q are tensors,

$$(p, q) = \int_{\Omega} p : q \quad \text{where} \quad p : q = \text{Tr}(pq^T). \quad (2.21)$$

2.9 Problem statement

With the above background, we are now ready to derive weak formulation and define problem in weak form. Following the approach presented by Montlaur et al. [2] and Montlaur et al. [1] we arrive at the weak form of the Stokes equations and the Navier-Stokes equations interior penalty approximation.

2.9.1 Stokes strong, weak and discrete form

The strong form of the Stokes equation is as follow,

$$-\nu \Delta u + \nabla p = f \quad \text{in } \Omega. \quad (2.22)$$

$$u = u_D \quad \text{on } \Gamma_D. \quad (2.23)$$

$$-pn + \nu n \cdot \nabla u = t \quad \text{on } \Gamma_N. \quad (2.24)$$

Derivation of the weak form of the Stokes equation

We multiply the Equation 2.22 by ϕ and integrate over \mathcal{T} ,

$$\int_{\mathcal{T}} (-\phi \nu \Delta u + \phi \cdot \nabla p) = \int_{\mathcal{T}} \phi \cdot f. \quad (2.25)$$

By applying Gauss divergence theorem,

$$\int_{\mathcal{T}} (-\nabla(\phi \nu \nabla u) + (\nu \nabla \phi : \nabla u) + \nabla \cdot (p\phi) - (p \nabla \cdot \phi)) = \int_{\mathcal{T}} \phi \cdot f, \quad (2.26)$$

$$\int_{\Gamma \cup \Gamma_D \cup \Gamma_N} (-(\phi \nu \nabla u) + \int_{\mathcal{T}} (\nu \nabla \phi : \nabla u) + \int_{\Gamma \cup \Gamma_D \cup \Gamma_N} [p\phi] - \int_{\mathcal{T}} (p \nabla \cdot \phi)) = \int_{\mathcal{T}} \phi \cdot f. \quad (2.27)$$

Following symmetric formulation for jump operator proposed by Peraire et al. [10],

$$\int_{\Gamma \cup \Gamma_D \cup \Gamma_N} [(\phi \nu \nabla u)] = \int_{\Gamma} (\{\phi\} : [n \otimes \nabla u]) + \int_{\Gamma} (\{\nabla u\} : [n \otimes \phi]) + \int_{\Gamma_D \cup \Gamma_N} (\phi \cdot u), \quad (2.28)$$

and

$$\int_{\Gamma \cup \Gamma_D \cup \Gamma_N} [p\phi] = \int_{\Gamma} (\{p\} [n \cdot \phi]) + \int_{\Gamma} (\{\phi\} \cdot [pn]) + \int_{\Gamma_D \cup \Gamma_N} (p(n \cdot \phi)) . \quad (2.29)$$

It is to be noted that analytical solution is continuous and hence $[u] = 0$. Therefore we add the term $([u], [\phi])$ which helps to maintain the coercivity of discontinuous approximation.

Collecting all Neumann boundary integral terms, the weak form of Stokes equation is as follow,

$$a_{IP}(u, \phi) + b(\phi, p) + (\{p\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D} = l_{IP}(\phi). \quad (2.30)$$

$$a_{IP}(u, \phi) = (\nabla u, \nabla \phi) + C_{11}([n \otimes u], [n \otimes \phi])_{\Gamma \cup \Gamma_D} - \nu(\{\nabla u\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D} - \nu([n \otimes u], \{\nabla \phi\})_{\Gamma \cup \Gamma_D}. \quad (2.31)$$

It is to be noted that penalty parameter C_{11} is to be kept large enough to maintain coercivity of bilinear form.

$$b(\phi, \psi) = - \int_{\mathcal{T}} \psi \nabla \cdot \phi, \quad (2.32)$$

$$l_{IP}(\phi) = (f, \phi) + (t, \phi)_{\Gamma_N} + C_{11}(u_D, \phi)_{\Gamma_D} - (n \otimes u_D, \nu \nabla \phi)_{\Gamma_D}. \quad (2.33)$$

The discrete form of Stokes equation is written as,

$$AU + BP = F_1. \quad (2.34)$$

Matrix A and matrix B are calculated as per equation 2.39 and equation 2.41 respectively.

The strong form of continuity equation is as follow,

$$\nabla \cdot u = 0 \quad \text{in } \Omega. \quad (2.35)$$

and the weak for of continuity equation is as follow,

$$b(u, \psi) + (\{\psi\}, [n \cdot u])_{\Gamma \cup \Gamma_D} = (q, n \cdot u_D)_{\Gamma_D}. \quad (2.36)$$

The discrete form of continuity equation is written as,

$$B^T U = F_2. \quad (2.37)$$

Discrete form of equations can be written in Matrix form as,

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}. \quad (2.38)$$

Stiffness matrix Solution vector Right hand side (Known)

Here, (\cdot, \cdot) is L^2 inner product, $\{\cdot\}$ is average operator, $[\cdot]$ is jump operator.

2.9.2 Properties of the stiffness matrix

We now write each element of matrix A . We represent components of Unit normal vector as $n = [n_1 \dots n_d]$

$$\begin{aligned} A_{ij} &= \sum_{k=1}^d \left(\frac{\partial \phi_i}{\partial x_k}, \frac{\partial \phi_j}{\partial x_k} \right) + C_{11} \sum_{k=1}^d ([\phi_i n_k], [\phi_j n_k])_{\Gamma \cup \Gamma_D} \\ &\quad - \nu \sum_{k=1}^d ([\phi_i n_k], \{\frac{\partial \phi_j}{\partial x_k}\})_{\Gamma \cup \Gamma_D} - \nu \sum_{k=1}^d (\{\frac{\partial \phi_i}{\partial x_k}\}, [\phi_j n_k])_{\Gamma \cup \Gamma_D}. \end{aligned} \quad (2.39)$$

We can see following properties of A :

1. $A_{ij} = A_{ji} \implies$ Symmetric,
2. A is positive definite : As the penalty parameter C_{11} is adjusted to ensure coercivity of A (Section A.0.6),
 $\exists c' > 0$ and for any non zero vector z ,

$$z^T A(\phi, \phi) z \geq c' \|z\|^2 \implies z^T A(\phi, \phi) z > 0. \quad (2.40)$$

3. Size of matrix A : $A \in \mathbb{R}^{u_{ndofs} \times u_{ndofs}}$ (u_{ndofs} is the total number of degrees of freedom of velocity).

Each element of B can be represented as,

$$B_{ij} = - \int_{\tau} \frac{\partial \phi_i}{\partial x_i} \psi_j + (\{\psi_j\}, [n \cdot \phi_i])_{\Gamma \cup \Gamma_D}. \quad (2.41)$$

We notice that, Size of matrix B : $B \in \mathbb{R}^{u_{ndofs} \times p_{ndofs}}$ (p_{ndofs} is total number of degrees of freedom of pressure)

u_{ndofs} and p_{ndofs} i.e. total number of degrees of freedom of velocity and pressure respectively on triangular grid and taylor-hood pressure velocity basis function can be calculated as below. In present analysis we have $d_u = 2$ and $d_p = 1$ and we represent total number of elements as nel .

$$u_{ndofs} = 2 \left(\frac{(D+1)(D+2)}{2} \right) nel \quad (2.42)$$

$$p_{ndofs} = \left(\frac{D(D+1)}{2} \right) nel \quad (2.43)$$

With above considerations we arrive at following conclusions,

1. Stiffness matrix is symmetric,
2. Stiffness matrix $\in \mathbb{R}^{(u_{ndofs}+p_{ndofs}) \times (u_{ndofs}+p_{ndofs})}$,
3. The number of positive eigenvalues of stiffness matrix is equal to number of velocity degrees of freedom and number of negative eigenvalues of stiffness matrix is equal to number of pressure degrees of freedom.

Proof :

From equation (A.9) we see that the congruent matrix of the stiffness matrix for Stokes equation is,

$$\begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix}. \quad (2.44)$$

We look at the the eigenvalues of S as the number of positive and negative eigenvalues of congruent matrix and stiffness matrix are same.

We see that $S \in \mathbb{R}^{p_{ndofs} \times p_{ndofs}}$, $S = -B^T A^{-1} B$.

For any non zero vector z , $z^T S z < 0$ i.e. S is symmetric negative definite and hence, all eigenvalues of S are negative.

Coercivity constant for the equation for the Stokes flow

We define a lower and upper bound for the kinematic viscosity such that

$$\nu_0 \leq \nu \leq \nu_1. \quad (2.45)$$

For neighbouring elements τ_{k_1} and τ_{k_2} sharing the edge e , using Cauchy Schwarz inequality,

$$\begin{aligned} \sum_{e \in \Gamma \cup \Gamma_D} \int_e \nu[n \otimes \phi] &\leq \sum_{e \in \Gamma \cup \Gamma_D} \|\nu \nabla \phi \cdot n\|_{L^2(e)} \quad |[[\phi]]|_{L^2(e)} \\ &\leq \sum_{e \in \Gamma \cup \Gamma_D} \|\nu \nabla \phi \cdot n\|_{L^2(e)} \left(\frac{1}{l}\right)^{(1/2-1/2)} \quad |[[\phi]]|_{L^2(e)}. \end{aligned} \quad (2.46)$$

Based on the Trace inequatlity and the lower and upper bound for viscocity,

$$\begin{aligned} \|\{\nu \nabla \phi \cdot n\}\|_{L^2(e)} &\leq \frac{C_t \nu_1}{2} h_{\tau_{k_1}}^{-1/2} \|\nabla \phi\|_{L^2(\tau_{k_1})} \\ &\quad + \frac{C_t \nu_1}{2} h_{\tau_{k_2}}^{-1/2} \|\nabla \phi\|_{L^2(\tau_{k_2})}. \end{aligned} \quad (2.47)$$

Again based on the trace inequality, for $e \in \Gamma \cup \Gamma_D$

$$\begin{aligned} \int_e \{\nu \nabla \phi\}[n \otimes \phi] &\leq \frac{C_t \nu_1}{2} \left(h_{(\tau_{k_1})}^{-\frac{1}{2}} \|\nabla \phi\|_{L^2(\tau_{k_1})} + h_{(\tau_{k_2})}^{-\frac{1}{2}} \|\nabla \phi\|_{L^2(\tau_{k_2})} \right) \\ &\quad l^{\frac{\beta_0}{2}} \left(\frac{1}{l} \right)^{\frac{\beta_0}{2}} |[[\phi]]|_{L^2(e)}. \end{aligned} \quad (2.48)$$

For $h_{\tau_k} \leq 1$ and the $\beta_0(d-1) \geq 1$ we obtain a similar bound for the boundary edges.

If n_0 denotes maximum number of neighbours ($n_0 = 3$ for triangles),

$$\begin{aligned} \int_e \{\nu \nabla \phi\}[n \otimes \phi] &\leq C_t \nu_1 \left(\sum_{e \in \Gamma_h \cup \Gamma_D} \frac{1}{l_0^\beta} |[[\phi]]|_{L^2(e)}^2 \right)^{1/2} \\ &\quad \times \left(\sum_{e \in \Gamma_h} \|\nabla \phi\|_{L^2(\tau_{k_1})}^2 + \|\nabla \phi\|_{L^2(\tau_{k_2})}^2 + \sum_{e \in \Gamma_D} \|\nabla \phi\|_{0,\tau_{k_1}}^2 \right) \end{aligned} \quad (2.49)$$

and

$$\begin{aligned} \int_e \{\nu \nabla \phi\}[n \otimes \phi] &\leq C_t \nu_1 \sqrt{n_0} \left(\sum_{e \in \Gamma_h \cup \Gamma_D} \frac{1}{l_0^\beta} |[[\phi]]|_{L^2(e)}^2 \right)^{1/2} \\ &\quad \left(\sum_{e \in \Gamma_h \cup \Gamma_D} \|\nabla \phi\|_{L^2(\tau)}^2 \right). \end{aligned} \quad (2.50)$$

Using Young's inequality for $\delta > 0$

$$\sum_{e \in \Gamma \cup \Gamma_D} \int_e \{\nu \nabla \phi\}[n \otimes \phi] \leq \frac{\delta}{2} \sum_{\tau \in \mathcal{T}} \|\nu^{1/2} \nabla \phi\|_{L^2(\tau)}^2 + \frac{C_t^2 \nu_1^2 n_0}{2\delta \nu_0} \sum_{e \in \Gamma \cup \Gamma_D} \frac{1}{l^{\beta_0}} |[[\phi]]|_{L^2(e)}^2, \quad (2.51)$$

$$a_\epsilon(\phi, \phi) \geq \left(1 - \frac{\delta}{2}|1 - \epsilon|\right) \sum_{\tau \in \mathcal{T}} \|\nu^{1/2} \nabla v\|_{L^2(\tau)}^2 + \sum_{e \in \Gamma_h \cup \Gamma_D} \frac{\sigma_e^0 - \frac{C_t^2 \nu_1^2 n_0}{2\delta K_0} |1 - \epsilon|}{l^{\beta_0}} \| [v] \|_{L^2(e)}^2, \quad (2.52)$$

Here, where constant σ_e^0 is related to penalty parameter C_{11} as,

$$C_{11} = \frac{\sigma_e^0}{l}. \quad (2.53)$$

We can obtain exact expression for minimum value of σ_e^0 , such that bilinear form is coercive, for triangular mesh and Symmetric interior penalty discontinuous Galerkin formulation i.e. $\epsilon = -1$ as,

$$\begin{aligned} \sigma_e^0 &= \frac{3(\nu_1 \tau_{k_1})^2}{2\nu_0 \tau_{k_1}} D(D+1) l^{\beta_0-1} \cot \theta^{\tau_{k_1}} \\ &\quad + \frac{3(\nu_1 \tau_{k_2})^2}{2\nu_0 \tau_{k_2}} D(D+1) l^{\beta_0-1} \cot \theta^{\tau_{k_2}} . \end{aligned} \quad (2.54)$$

2.9.3 Upwinding

Upwinding is the method used to discretise the convective term. In the case of discontinuous Galerkin method we define the upwinding as follow.

If n_τ is the unit normal from τ_1 to τ_2 and if we denote the upwind value of function g as g^{up} [6],

$$\begin{aligned} g^{up} &= g|_{\tau_1} && \text{if } g \cdot n_\tau \geq 0, \\ g^{up} &= g|_{\tau_2} && \text{if } g \cdot n_\tau < 0. \end{aligned} \quad (2.55)$$

In our analysis this is explicitly defined in the term $c(u, u, \phi)$ in section 2.9.4. The formulation is such that,

$$\begin{aligned} u^{up} &= u && \text{if } u \cdot n \geq 0, \\ u^{up} &= u^{ext} && \text{if } u \cdot n < 0. \end{aligned} \quad (2.56)$$

2.9.4 Navier Stokes strong, weak and discrete form

The Stokes flow is an example of the Navier Stokes flow with low Reynolds number, Re . In case of high Reynolds number the inertial force can no longer be neglected and hence we need to add inertial forces in the Stokes flow.

The strong form of the Navier Stokes equation can be written as,

$$-\nu \Delta u + \nabla p + (u \cdot \nabla) u = f \quad \text{in } \Omega. \quad (2.57)$$

with Dirichlet and Neumann boundary condition as per section 2.9.1. Also the continuity equation as mentioned in section 2.9.1 is valid.

The inertial forces term i.e. $(u \cdot \nabla) u$ in weak form with upwinding (section 2.9.3) is represented as below,

$$\begin{aligned} c(g; u, \phi) = \sum_{i=1}^{nel} \int_{\partial\Omega_i \setminus \Gamma_N} \frac{1}{2} [[(g \cdot n_i)(u^{ext} + u) - |g \cdot n_i|(u^{ext} - u)] \cdot \phi \\ + \int_{\Gamma_N} (g \cdot n) u \cdot \phi - ((g \cdot \nabla)\phi, u) . \end{aligned} \quad (2.58)$$

$$u^{ext} = \lim_{\epsilon \rightarrow 0} u(x + \epsilon n_i) \quad \text{for } x \in \partial\mathcal{T}_i. \quad (2.59)$$

Hence, the Navier Stokes equation can be written as,

$$a_{IP}(u, \phi) + c(u; u, \phi) + b(\phi, p) + (\{p\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D} = l_{IP}(\phi). \quad (2.60)$$

Here, we can see that the $c(u, u, \phi)$ is non linear term.

In discrete form this equation can be written as,

$$AU + C(U)U + BP = F. \quad (2.61)$$

Here, $C(U)$ is a matrix which is dependent on solution vector U and hence making the system of equation non linear.

In matrix form the Navier Stokes equation with the continuity equation can be written as,

$$\begin{pmatrix} A + C(U) & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}, \quad (2.62)$$

Stiffness matrix , Solution vector , Right hand side (Known) .

In the present analysis, we use Newton method to solve nonlinear system of equations. We present the Newton scheme in section 2.9.5.

Our problem statement now reduces to, find $(u, p) \in (\mathbb{V}, \mathbb{Q})$ such that $\forall(\phi, \psi) \in (\mathbb{V}, \mathbb{Q})$ equation 2.30 for the Stokes flow and equation 2.60 for the Navier Stokes flow is satisfied.

2.9.5 Newton method

We derive Newton method as per method presented by Haasdonk B. [5] for solving nonlinear system of equations arising out of the discrete form of the Navier Stokes equation. For $u, \phi, h \in \mathbb{V}$ and $p, \psi, h' \in \mathbb{Q}$

$$S(u) = a(u, \phi) + b(\phi, p) + (\{p\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D} - l_{IP}(\phi), \quad (2.63)$$

$$\begin{aligned} S(u + h) - S(u) = (a(u + \delta h, \phi) + c(u + \delta h; u + \delta h, \phi) \\ + b(\phi, p + \delta h') + (\{p + \delta h'\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D} - l_{IP}(\phi)) - (a(u, \phi) \\ + c(u, u, \phi) + b(\phi, p) + (\{p\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D} - l_{IP}(\phi)), \end{aligned} \quad (2.64)$$

$$\begin{aligned} S(u + h) - S(u) &= 2\delta c(u, h, \cdot) + \delta^2 c(h, h, \cdot) + \delta a(h, \cdot) \\ &\quad + \delta b(h', \cdot) + \delta(\{h'\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}, \end{aligned} \quad (2.65)$$

$$DS(u) = \lim_{\delta \rightarrow 0} \frac{S(u + h) - S(u)}{\delta}, \quad (2.66)$$

$$DS(u) = 2c(u, h, \cdot) + a(h, \cdot) + b(h', \cdot) + (\{h'\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}. \quad (2.67)$$

Following similar procedure we write for continuity equation:

$$S'(u) = b(u, \psi) + (\{\psi\}, [n \cdot u])_{\Gamma \cup \Gamma_D} - (\psi, n \cdot u_D)_{\Gamma_D}, \quad (2.68)$$

$$S'(u + \delta h) = b(u + \delta h, \psi) + (\{\psi\}, [n \cdot u + \delta h])_{\Gamma \cup \Gamma_D} - (\psi, n \cdot u_D)_{\Gamma_D}, \quad (2.69)$$

$$DS'(u) = b(\delta h, \psi) + (\{\psi\}, [n \cdot \delta h])_{\Gamma \cup \Gamma_D}. \quad (2.70)$$

Algorithm for the Newton method is as follow [5] :

1. Select $u^{iter} \in \mathbb{V}$ at iteration $iter$;
2. Verify $DS_{u^{iter}}(h^{iter}) = -S(u^{iter})$;
3. Set $u^{iter+1} := u^{iter} + h^{iter}$ till $\|u^{iter+1} - u^{iter}\| < tol$ where tol is specified tolerance.

In discrete form the Newton method means, solving the equation (at iteration $= iter$)

$$\begin{array}{ccc} \left(\begin{array}{cc} A + C(U^{iter}) & B \\ B^T & 0 \end{array} \right) & \left(\begin{array}{c} U^{iter+1} \\ P^{iter+1} \end{array} \right) & = \left(\begin{array}{c} F_1 \\ F_2 \end{array} \right) \\ \text{Stiffness matrix}^{iter} & \text{Solution vector}^{iter+1} & \text{Right hand side (Known function)} . \end{array} \quad (2.71)$$

to reach convergence in suitable norm i.e. $\|U^{iter+1} - U^{iter}\| < tol$, some specified tolerance.

For success of the Newton method it is important to have good initial guess. For the Newton method we use solution of the Stokes equation as initial guess.

2.9.6 Properties of the stiffness matrix

Each element of C , at newton iteration $iter + 1$, can be represented as,

$$\begin{aligned} C_{IJ} = \sum_{i=1}^{nel} \int_{\partial\mathcal{T}_i \setminus \Gamma_N} & \frac{1}{2} [[(U^{iter} \cdot n_i)(\phi^{ext}_J + \phi_J) - |U^{iter} \cdot n_i|(\phi^{ext}_J - \phi_J)] \cdot \phi_I \\ & + \int_{\Gamma_N} (U^{iter} \cdot n) \phi_J \cdot \phi_I - ((U^{iter} \cdot \nabla) \phi_J, \phi_I) . \end{aligned} \quad (2.72)$$

Size of matrix : $C \in \mathbb{R}^{u_{ndofs} \times u_{ndofs}}$.

We now can see that whereas the stiffness matrix of the Stokes equation is symmetric and has symmetric positive definite part A , the stiffness matrix of the Navier-Stokes equations is non symmetric.

2.9.7 Boundary conditions

We impose in our analysis boundary conditions weakly. This is done by the linear terms on the right hand side of equation $(t, \phi)_{\Gamma_N}$, $(u_D, \phi)_{\Gamma_D}$, $(n \otimes u_D, \nu \nabla \phi)_{\Gamma_D}$ in equation (2.60) and equation (2.30).

2.10 Selection of solver

In order to solve the variation form of the Stokes equation we use Biconjugate gradients stabilized method popularly known as *bicgstab*, Minimum residual method better known as *minres* and Schur complement method. The *bicgstab* and *minres* are in built solvers of MATLAB. Schur complement method (Section 2.10.3) is implemented separately based on Cholesky decomposition (Section A.0.1).

2.10.1 Biconjugate gradients stabilized method

The *bicgstab* works to minimise residual of linear equation of the form:

$$KX = B, \quad (2.73)$$

K = Coefficient matrix,

X = Vector of unknowns,

B = vector of value of known function.

The coefficient matrix A need not be symmetric. We refer to [9] for details about the method.

2.10.2 Minimum residual method

The *minres* method is a special kind of conjugate gradient method but does not require LU decomposition. We refer to [3] for details about the method. It also solves the equations 2.73. However, the coefficient matrix K must be symmetric.

As the coefficient matrix in case discrete form of Stokes equation is symmetric, the *minres* is suitable solver. Moreover, it has shown to converge to required convergence level, provided sufficient number of iterations, when *bicgstab* is not able to reach required level of convergence.

2.10.3 Schur complement method

We refer to methods by Fritzen F. [7]. We subdivide the matrix form of the Stokes equation (A.2) into smaller dimension system by Schur complement method. We also note that matrix A is symmetric positive definite and matrix K is symmetric. Here, $A \in \mathbb{R}^{undofs \times pndofs}$; $B \in \mathbb{R}^{undofs \times pndofs}$; $U, F_1 \in \mathbb{R}^{undofs}$; $P, F_2 \in \mathbb{R}^{pndofs}$.

We solve equation (A.2) in following steps,

STEP 1:

$$U = A^{-1}(F_1 - BP) \quad (2.74)$$

The matrix A is inverted by Cholesky decomposition (section A.0.1).

STEP 2 :

We substitute now equation (2.74) into equation (2.37) and (2.34)

$$-B^T A^{-1} B P = F_2 - B^T A^{-1} F_1 \quad (2.75)$$

STEP 3 :

We now back substitute P in equation (2.74) and compute U in order to eventually obtain the solution vector.

As pointed by Fritzen F. [7], the success of this method primarily depends upon the sparsity pattern of B and efforts required for inverting A . The Cholesky decomposition provides faster approach for inverting A due to symmetric positive definite nature of A .

In the present analysis, we find that the Schur complement method is much faster and in fact, for low flow velocities accurate method. Also Cholesky decomposition provides error message in case A is not symmetric positive definite, indicating improper choice of penalty parameter.

Chapter 3

Implementation aspects

We discuss now the implementation of the discrete formulation of the Navier-Stokes discontinuous-Galerkin weak formulation in RBmatlab, A MATLAB library containing all our numerical simulation approaches for linear and nonlinear, affine or arbitrarily parameter dependent evolution problems with finite element, finite volume or local discontinuous Galerkin discretizations. We refer to the literature by Haasdonk B.[4] and RBmatlab website:

<http://www.ians.uni-stuttgart.de/MoRePaS/software/rbmatlab/1.16.09/index.html>.

Before we discuss details of the implementation it is important to understand some frequently used terminologies and the data type of basis Function and derivative of basis function in RBmatlab.

3.1 Terminology

A. `params` and `paramsP` : These are structures corresponding to velocity and pressure respectively and primarily containing fields related to basis function. (Table 3.1 and Table 3.2)

B. `grid` : It is the structure containing fields relevant to information stored in grid. (Table 3.3)

C. We also define some other variables. (Table 3.4)

3.2 Basis functions in RBmatlab

`ldg_evaluate_basis` routine of RBmatlab evaluates the orthonormal basis functions at a given point.

We approximate the solution at any point in element τ_k from the solution of degrees of freedom of τ_k in orthonormal basis. In matrix formulation this means basis function is matrix of the size :

Table 3.1: Fields of `params`

Fieldname	Representation	Size	Description
<code>dimrange</code>	d_u	\mathbb{N}	User defined field
<code>pdeg</code>	D	\mathbb{N}	User defined field
<code>ndofs_per_element</code>	u_{npe}	\mathbb{N}	Calculated as per equation 2.5
<code>ndofs</code>	u_{ndofs}	\mathbb{N}	Calculated as per equation 2.42
<code>dofs</code>	U	$\mathbb{R}^{u_{ndofs}}$	Calculated after solving 2.60 and 2.30
<code>show_sparsity</code>	-	Bool variable	User defined switch to plot sparsity pattern

Table 3.2: Fields of `paramsP`

Fieldname	Representation	Size	Description
<code>dimrange</code>	d_p	\mathbb{N}	User defined field
<code>pdeg</code>	$D - 1$	\mathbb{N}	Calculated as taylor hood element
<code>ndofs_per_element</code>	p_{npe}	\mathbb{N}	Calculated as per equation 2.6
<code>ndofs</code>	p_{ndofs}	\mathbb{N}	Calculated as per equation 2.43
<code>dofs</code>	P	$\mathbb{R}^{p_{ndofs}}$	Calculated after solving 2.60 and 2.30
<code>show_sparsity</code>	-	Bool variable	User defined switch to plot sparsity pattern

Table 3.3: Fields of **grid**

Fieldname	Representation	Size	Description
nelement	nel	\mathbb{N}	User defined or generated field
NBI	—	$\mathbb{R}^{nel \times n_0}$	$NBI(i, j)$ is the j^{th} neighbour of element i
NX	n_x	$\mathbb{R}^{nel \times d}$	x -component of unit normal vector
NY	n_y	$\mathbb{R}^{nel \times d}$	y -component of unit normal vector
A	Ar	\mathbb{R}^{nel}	Area of element
EL	l	$\mathbb{R}^{nel \times n_0}$	edge length
JIT	JIT	$\mathbb{R}^{nel \times n_0 \times d}$	Jacobian inverse transpose

Table 3.4: Some other variables

Name	Representation	Size	Description
k	k	\mathbb{N}	element number, $1 \leq k \leq nel$
ids	—	\mathbb{R}^{npe}	indices of degrees of freedom in global vector

$$\phi \in \mathbb{R}^{u_{npe} \times d_u}, \quad (3.1)$$

and

$$\psi \in \mathbb{R}^{p_{npe} \times d_p}. \quad (3.2)$$

This representation creates many zeros in matrix, however, it does not require new evaluation for each component of vector quantity. Due to the zeros in the basis function, derivative of basis functions also has many zeros.

`ldg_evaluate_basis_derivative` is the routine which evaluates the derivative of the basis functions. The derivative of basis function $(\phi)_i$, where $1 \leq i \leq u_{npe}$ is a cell containing matrix $\nabla(\phi)_i$ of size,

$$\nabla(\phi)_i \in \mathbb{R}^{d_u \times d}. \quad (3.3)$$

The columns of the matrix correspond to $\nabla_x(\phi)_i$ and $\nabla_y(\phi)_i$.

The derivative of basis function $(\psi)_i$, where $1 \leq i \leq p_{npe}$ is a cell containing matrix $\nabla(\psi)_i$ of size,

$$\nabla(\psi)_i \in \mathbb{R}^{d_p \times d}. \quad (3.4)$$

The columns of the matrix correspond to $\nabla_x(\psi)_i$ and $\nabla_y(\psi)_i$.

3.3 Assembly of average operator

Average of quantity A_h in discrete form is assembled as

$$\{A_h\} = \frac{(A_h^+ + A_h^-)}{2}. \quad (3.5)$$

The assembly of the average operator is relatively simple as compared to the jump operator which is explained in the section 3.5.

3.4 Assembly of jump operator

The jump of the quantity A_h in discrete form is assembled in line with explanation as per section 2.6. For example $[A_h \cdot n]$ is assembled as

$$[A_h \cdot n] = A_h^+ n^+ + A_h^- n^-. \quad (3.6)$$

In case of terms such as $[A_h \cdot n]$, $[B_h \cdot n]$ we assemble matrices as, for internal edges Γ ,

$$[A_h \cdot n], [B_h \cdot n] = A_h^+ n^+ B_h^+ n^+ + A_h^+ n^+ B_h^- n^- + A_h^- n^- B_h^+ n^+ + A_h^- n^- B_h^- n^-, \quad (3.7)$$

and for Dirichlet edges Γ_D

$$[A_h \cdot n], [B_h \cdot n] = A_h^+ n^+ B_h^+ n^+. \quad (3.8)$$

3.5 Matrix assemblies

It is to be noted that $\hat{\phi}$ is evaluated at local coordinate corresponding to global coordinate in accordance with the equation (2.14).

The matrices from weak form of the Navier Stokes equation have been assembled in 3 steps :

1. Evaluating function at vertex of local element and transform to global geometry (Equation (2.14)).
2. Transform function evaluation from step 1 to global geometry (if not done in step 1) and performing integral of function over local element (Equation (2.12) and (2.13)).
3. Performing a loop over all elements and allocate integral at position in global matrix (for bilinear terms)/global vector (for linear terms) according to indices of element degrees of freedom in global degree of freedom vector.

We also perform numerical integration over domain Ω as,

$$\int_{\Omega} f(x) = \sum_{i=1}^{nop} f(x_i) w_i, \quad (3.9)$$

x_i = Location of function evaluation,

w_i = Weight at corresponding point,

nop = Number of points.

The location of function evaluation, number of points and weights are based on Gaussian quadrature rule.

Also the determinant of the Jacobian is twice the area of triangle.

$$\det J(k) = 2Ar(k). \quad (3.10)$$

With this preliminary informations we discuss now the assembly of matrices.

3.5.1 Assembly of $(\nabla\phi, \nabla\phi)$

Step 1: Function evaluation

Through `ldg_evaluate_basis_derivative` and `grid` structure, we first evaluate the derivative of $\hat{\phi}$ and $JIT(k, :, :)$ at point x_i . We also perform elementary operation so as to receive one global basis function in each row. The matrix transformation from local derivative to global derivative is based on the equation (2.13).

We than assemble the matrix,

$$res_1[i, j] = \nabla\phi_i \nabla\phi_j^T \quad \text{for } 1 \leq i, j \leq u_{npe}. \quad (3.11)$$

Step 2: Performing integration in global coordinate system

We perform the numerical integration as per equation (3.9)

$$res_2 = \int_{\hat{T}} (res_1)(2Ar(k)). \quad (3.12)$$

Step 3: Degrees of freedom allocation in global matrix

Looping over each element and performing following operation in each loop,

$$res_3[ids_velocity, ids_velocity] = res_2. \quad (3.13)$$

3.5.2 Assembly of $([n \otimes \phi], [n \otimes \phi])_{\Gamma \cup \Gamma_D}$

Step 1: Function evaluation

Following matrices are evaluated in first step.

$$\begin{aligned} res_1^{++} &= (n \otimes \hat{\phi})^+ (n \otimes \hat{\phi})^+, \\ res_1^{+-} &= (n \otimes \hat{\phi})^+ (n \otimes \hat{\phi})^-, \\ res_1^{-+} &= (n \otimes \hat{\phi})^- (n \otimes \hat{\phi})^+, \\ res_1^{--} &= (n \otimes \hat{\phi})^- (n \otimes \hat{\phi})^-. \end{aligned} \quad (3.14)$$

Please note that $\hat{\phi}_h$ is evaluated at local coordinate corresponding to global coordinate.

Step 2: In step 2 we perform following integration

We perform the numerical integration as per equation (3.9).

$$\begin{aligned} res_2^{++} &= \int_{\Gamma} res_1^{++} EL(i, j), \\ res_2^{+-} &= \int_{\Gamma} res_1^{+-} EL(i, j), \\ res_2^{-+} &= \int_{\Gamma} res_1^{-+} EL(i, j), \\ res_2^{--} &= \int_{\Gamma} res_1^{--} EL(i, j). \end{aligned} \quad (3.15)$$

Step 3: Degrees of freedom allocation in global matrix

Loop over all elements and perform following operation in each loop.

$$\begin{aligned} res_3^{++}[ids_velocity_self, ids_velocity_self] &= res_2^{++}, \\ res_3^{+-}[ids_velocity_self, ids_velocity_neighbour] &= res_2^{+-}, \\ res_3^{-+}[ids_velocity_neighbour, ids_velocity_self] &= res_2^{-+}, \\ res_3^{--}[ids_velocity_neighbour, ids_velocity_neighbour] &= res_2^{--}. \end{aligned} \quad (3.16)$$

Finally,

$$res_3 = res_3^{++} + res_3^{+-} + res_3^{-+} + res_3^{--}. \quad (3.17)$$

It is to be noted that on dirichlet boundary only $res_1^{++}, res_2^{++}, res_3^{++}$ is evaluated as all other terms are zero.

3.5.3 Assembly of $(\{\nabla\phi\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D}$

Step 1: Function evaluation

We evaluate following matrices,

$$\begin{aligned} res_1^{++} &= (\nabla\phi)^+(n \otimes \hat{\phi})^+, \\ res_1^{+-} &= (\nabla\phi)^+(n \otimes \hat{\phi})^-, \\ res_1^{-+} &= (\nabla\phi)^-(n \otimes \hat{\phi})^+ \\ &\quad \text{textrm,} \\ res_1^{--} &= (\nabla\phi)^-(n \otimes \hat{\phi})^-. \end{aligned} \quad (3.18)$$

Step 2: Integration of evaluated function

We perform the numerical integration as per equation (3.9).

$$\begin{aligned} res_2^{++} &= \int_{\Gamma} res_1^{++} EL(i, j), \\ res_2^{+-} &= \int_{\Gamma} res_1^{+-} EL(i, j), \\ res_2^{-+} &= \int_{\Gamma} res_1^{-+} EL(i, j), \\ res_2^{--} &= \int_{\Gamma} res_1^{--} EL(i, j). \end{aligned} \quad (3.19)$$

Step 3: Allocation in global degrees of freedom matrix

Loop over all elements, define the global assembly matrix as zero matrix and perform following operation in each loop,

$$\begin{aligned} res_3^{++}[ids_velocity_self, ids_velocity_self] &= res_2^{++}, \\ res_3^{+-}[ids_velocity_self, ids_velocity_neighbour] &= res_2^{+-}, \\ res_3^{-+}[ids_velocity_neighbour, ids_velocity_self] &= res_2^{-+}, \\ res_3^{--}[ids_velocity_neighbour, ids_velocity_neighbour] &= res_2^{--}. \end{aligned} \quad (3.20)$$

Finally,

$$res_3 = res_3^{++} + res_3^{+-} + res_3^{-+} + res_3^{--}. \quad (3.21)$$

It is to be noted that on Dirichlet boundary only $res_1^{++}, res_2^{++}, res_3^{++}$ is evaluated as all other terms are zero.

3.5.4 Assembly of $(\{\psi\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}$

Step 1: Function evaluation

We evaluate following matrices,

$$\begin{aligned}
res_1^{++} &= (\psi)^+(n \cdot \hat{\phi})^+, \\
res_1^{+-} &= (\psi)^+(n \cdot \hat{\phi})^-, \\
res_1^{-+} &= (\psi)^-(n \cdot \hat{\phi})^+, \\
res_1^{--} &= (\psi)^-(n \cdot \hat{\phi})^-.
\end{aligned} \tag{3.22}$$

Please note that $\hat{\phi}$ is evaluated at local coordinate corresponding to global coordinate in accordance with (2.14).

Step 2: Integration of evaluated function

In step 2, we perform the numerical integration as per equation (3.9).

$$\begin{aligned}
res_2^{++} &= \int_{\Gamma} res_1^{++} EL(i, j), \\
res_2^{+-} &= \int_{\Gamma} res_1^{+-} EL(i, j), \\
res_2^{-+} &= \int_{\Gamma} res_1^{-+} EL(i, j), \\
res_2^{--} &= \int_{\Gamma} res_1^{--} EL(i, j).
\end{aligned} \tag{3.23}$$

Step 3: Allocation of matrices in global degrees of freedom matrix

Loop over all elements, define the global assembly matrix as zero matrix and perform following operation in each loop,

$$\begin{aligned}
res_3^{++}[ids_velocity_self, ids_velocity_self] &= res_2^{++}, \\
res_3^{+-}[ids_velocity_self, ids_velocity_neighbour] &= res_2^{+-}, \\
res_3^{-+}[ids_velocity_neighbour, ids_velocity_self] &= res_2^{-+}, \\
res_3^{--}[ids_velocity_neighbour, ids_velocity_neighbour] &= res_2^{--}.
\end{aligned} \tag{3.24}$$

Finally,

$$res_3 = res_3^{++} + res_3^{+-} + res_3^{-+} + res_3^{--}. \tag{3.25}$$

It is to be noted that on Dirichlet boundary only res_1^{++} , res_2^{++} , res_3^{++} is evaluated as all other terms are zero.

3.5.5 Assembly of $-\int_{\hat{T}} \psi \nabla \cdot \phi$

We note that, in accordance with equation (2.13)

$$\nabla \phi = JIT \hat{\nabla} \hat{\phi}, \tag{3.26}$$

and in accordance with equation (2.14)

$$\psi(x) = \hat{\psi}(\hat{x}). \tag{3.27}$$

Step 1: Function evaluation

We first evaluate JIT , $\hat{\nabla}\hat{\phi}$ and ψ and assemble following local matrix

$$res_1 = \hat{\psi}_i \hat{\nabla} \cdot \hat{\phi}_j. \quad (3.28)$$

Step 2: Integration of evaluated function

We integrate the evaluated function over domain

$$res_2 = - \int_{\hat{T}} (res_1)(2Ar(k)). \quad (3.29)$$

Step 3: Allocation in global matrix

Assemble the global matrix

$$res_3[ids_pressure, ids_velocity] = res_2. \quad (3.30)$$

3.5.6 Assembly of $(t, \phi)_{\Gamma_N}$ **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = \hat{\phi}_i \cdot t, \quad (3.31)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over the Neumann edge.

$$res_2 = \int_{\Gamma_N} res_1 EL(i, j). \quad (3.32)$$

Step 3: Allocation in global degrees of freedom vector

Loop over all Neumann edges and perform following operation in each loop.

$$res_3[ids] = res_2. \quad (3.33)$$

3.5.7 Assembly of $(u_D, \phi)_{\Gamma_D}$ **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = \hat{\phi}_i u_D, \quad (3.34)$$

in accordance with the equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over an element

$$res_2 = \int_{\Gamma_D} res_1 EL(i, j). \quad (3.35)$$

Step 3: Allocation in global degrees of freedom vector

Loop over element having Dirichlet boundary and perform following operation in each loop.

$$res_3[ids] = res_2. \quad (3.36)$$

3.5.8 Assembly of $(\psi, n \cdot u_D)_{\Gamma_D}$ **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = \hat{\psi} n \cdot u_D, \quad (3.37)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over an element

$$res_2 = \int_{\Gamma_D} res_1 EL(i, j). \quad (3.38)$$

Step 3: Allocation in global degrees of freedom vector

Loop over all Dirichlet edges and perform following operation in each loop.

$$res_3[ids] = res_2. \quad (3.39)$$

3.5.9 Assembly of (f, ϕ) **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = \hat{\phi} f, \quad (3.40)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over an element.

$$res_2 = \int_{\hat{\Omega}} res_1 2Ar(k). \quad (3.41)$$

Step 3: Allocation in global degrees of freedom vector

Loop over element having Dirichlet boundary and perform following operation in each loop.

$$res_3[ids] = res_2. \quad (3.42)$$

3.5.10 Assembly of $(n \otimes u_D, \nabla \phi)_{\Gamma_D}$ **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = n \otimes u_D \nabla \phi, \quad (3.43)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over a Dirichlet edge :

$$res_2 = \int_{\Gamma_D} res_1 EL(i, j). \quad (3.44)$$

Step 3: Allocation in global matrix

Loop over all Dirichlet edges and perform following operation in each loop.

$$res_3[ids] = res_2. \quad (3.45)$$

We discuss now assembly of non linear terms. We now introduce initial guess u_k which will be iterated further.

3.5.11 Assembly of $-((u_k \cdot \nabla) \phi, \phi)$ **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = (u_k \cdot \nabla \hat{\phi}_i \hat{\phi}_j), \quad (3.46)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over an element

$$res_2 = - \int_{\hat{T}} (res_1)(2Ar(k)). \quad (3.47)$$

Step 3: Allocation in global matrix

Loop over all elements and perform following operation in each loop

$$res_3[ids, ids] = res_2. \quad (3.48)$$

3.5.12 Assembly of $((u_k \cdot n)\phi, \phi)_{\Gamma_N}$ **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = (u_k \cdot n)\hat{\phi}_i \hat{\phi}_j, \quad (3.49)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over a Neumann edge.

$$res_2 = \frac{1}{2} \int_{\Gamma_N} res_1 EL(i, j). \quad (3.50)$$

Step 3: Allocation in global matrix

Loop over all Neumann edges and perform following operation in each loop.

$$res_3[ids, ids] = res_2. \quad (3.51)$$

3.5.13 Assembly of $((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N}$ **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = (u_k \cdot n)\hat{\phi}_i \hat{\phi}_j^{ext}, \quad (3.52)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over an edge

$$res_2 = \frac{1}{2} \int_{\Gamma} res_1 EL(i, j). \quad (3.53)$$

Step 3: Allocation in global matrix

Loop over all internal edges and Dirichlet edges and perform following operation in each loop.

$$res_3[ids, ids^{ext}] = res_2. \quad (3.54)$$

3.5.14 Assembly of $(|u_k \cdot n| \phi, \phi^{ext})_{\partial T \setminus \Gamma_N}$ **Step 1: Function evaluation**

On local element following function is evaluated.

$$res_1 = (|u_k \cdot n|) \hat{\phi}_i \hat{\phi}_j^{ext}, \quad (3.55)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over an edge as,

$$res_2 = \frac{1}{2} \int_{\Gamma} res_1 EL(i, j). \quad (3.56)$$

Step 3: Allocation in global matrix

Loop over all internal edges and Dirichlet edges and perform following operation in each loop.

$$res_3[ids, ids^{ext}] = res_2. \quad (3.57)$$

3.5.15 Assembly of $((u_k \cdot n) \phi, \phi)_{\partial T \setminus \Gamma_N}$ **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = (u_k \cdot n) \hat{\phi}_i \hat{\phi}_j, \quad (3.58)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over an edge as,

$$res_2 = \frac{1}{2} \int_{\Gamma} res_1 EL(i, j). \quad (3.59)$$

Step 3: Allocation in global matrix

Loop over all internal edges and Dirichlet edges and perform following operation in each loop.

$$res_3[ids, ids] = res_2. \quad (3.60)$$

3.5.16 Assembly of $((|u_k \cdot n| \phi, \phi)_{\partial T \setminus \Gamma_N}$ **Step 1: Function evaluation**

On local element following function is evaluated

$$res_1 = (|u_k \cdot n|) \hat{\phi}_i \hat{\phi}_j, \quad (3.61)$$

in accordance with equation (2.14).

Step 2: Integration of evaluated function

An integral is performed over an edge

$$res_2 = \frac{1}{2} \int_{\Gamma} res_1 EL(i, j). \quad (3.62)$$

Step 3: Allocation in global matrix

Loop over all internal edges and Dirichlet edges and perform following operation in each loop.

$$res_3[ids, ids] = res_2. \quad (3.63)$$

3.6 Setting boundary conditions

We extract the Dirichlet edges and Neumann edges by routine `tria_edge_index_Dirichlet` and `tria_edge_index_Neumann` respectively.

We set the Dirichlet and Neumann boundary values in routine `Dirichlet_values` and `Neumann_values` respectively as column vectors.

3.7 Setting source term

We set the source term values in routine `func_rhs` as column vectors. This routine is called during the process of assembling vector term (f, ϕ) of right hand side of equation (2.60) and equation (2.30).

3.8 Program flow

The main script of the program is `mymodel`. This script is equivalent to `main.cc` file in C++.

3.8.1 Grid preparation

We use `pdegrid` tool to generate grid. We generate mesh and export parameters `point(p)`, `edge(e)` and `triangle(t)`. `params.bnd_rect_corner1` marks the lower corner of to be marked boundary and `params.bnd_rect_corner2` marks the upper corner of to be marked boundary. `params.bnd_rect_index` marks the type of boundary. (-1) represents Dirichlet boundary and (-2) represents Neumann boundary. `params.gridtype` defines the type of grid.

`construct_grid(params)` constructs the grid as struct containing necessary fields for grid. In case of creating rectangular grid without using `pdegrid`, we define fields `xrange`, `yrange`, `xnumintervals` and `ynumintervals` defining range of x co-ordinates of domain, range of y co-ordinates of domain, number of intervals for x division and number of intervals for y division respectively and call routine `construct_grid`.

3.8.2 Function space formulation

We now define fields for constructing function spaces for pressure and velocity. We now use two structures `params` for velocity and `paramsP` for pressure as explained previously (Table 3.1 and Table 3.2). These structures contain fields (as relevant to function space formulation) `pdeg` and `dimrange`. `pdeg` represents polynomial degree of basis function and `dimrange` represents dimension of quantity i.e. 2 for velocity and 1 for pressure. The field `paramsP.pdeg` is calculated as `paramsP.pdeg = params.pdeg - 1` in accordance with Taylor hood element. Based on these fields `ndofs_per_element`, `ndofs` are calculated. Number of elements in grid can be read from `grid.nelements`. We also define degree for integration `qdeg` and kinematic viscosity in `params.kinematic_viscosity`. Variable `mu` also holds the value of kinematic viscosity. We define penalty parameter in variable C_{11} . Bool variable `show_sparsity` plots sparsity pattern of each matrix if set to true.

3.8.3 Matrix assembly

We now assemble all the matrices as explained during matrix assembly (routine `assemble_stiffness_matrix`). The individual matrices are assembled and stored in struct `params`. `params.bilinear_side` contains A or assembly of $a(u, \phi)$. `params.bilinear_side_pressure_terms` contains assembly of B or assembly of $b(\phi, \psi)$. `params.lhs_continuity` contains assembly of B^T . `rhs` is the right hand side matrix [$F_1; F_2$] as [`params.linear_side; params.rhs_continuity`].

3.8.4 Solving assembled form

We now define the solver specific variables. `required_residual_tol` specifies the required accuracy from solver in solution and `max_iter` specifies the maximum number of iterations that is used to stop the solver in case the solver does not converge. We call the routine `solve_plot_solution` for `bicgstab,minres` (The solver is specified in `solve_plot_solution`). In case of Schur complement method, routine `solve_plot_solution_schur` is used. The output variable `achieved_residual_tol` contains residual value, `params` and `paramsP` are given new values `dofs` which contain degree of freedom. `actual_iter` is the value of number of iterations at the end of iterations process. `flag` specifies the criteria for end of iteration process.

3.8.5 Post processing

We now enter into `stiffness_matrix_test`, to check properties of assembled stiffness matrix as explained in section 2.9.2. Then we enter into error measurement (`error_l2_norm_assembly` and `error_h0_norm_assembly`) which measures the error in L^2 or H_0 norm. `params.dof_analytical`, `paramsP.dof_analytical` contain analytical expression against which numerical solution is to be compared. `params.dof_derivative_analytical` and `paramsP.dof_derivative_analytical` contain analytical expression for derivative of the solution to be used for H_0 norm.

We plot the solution using `ldg_plot` implemented for plotting discontinuous functions.

3.8.6 Newton method

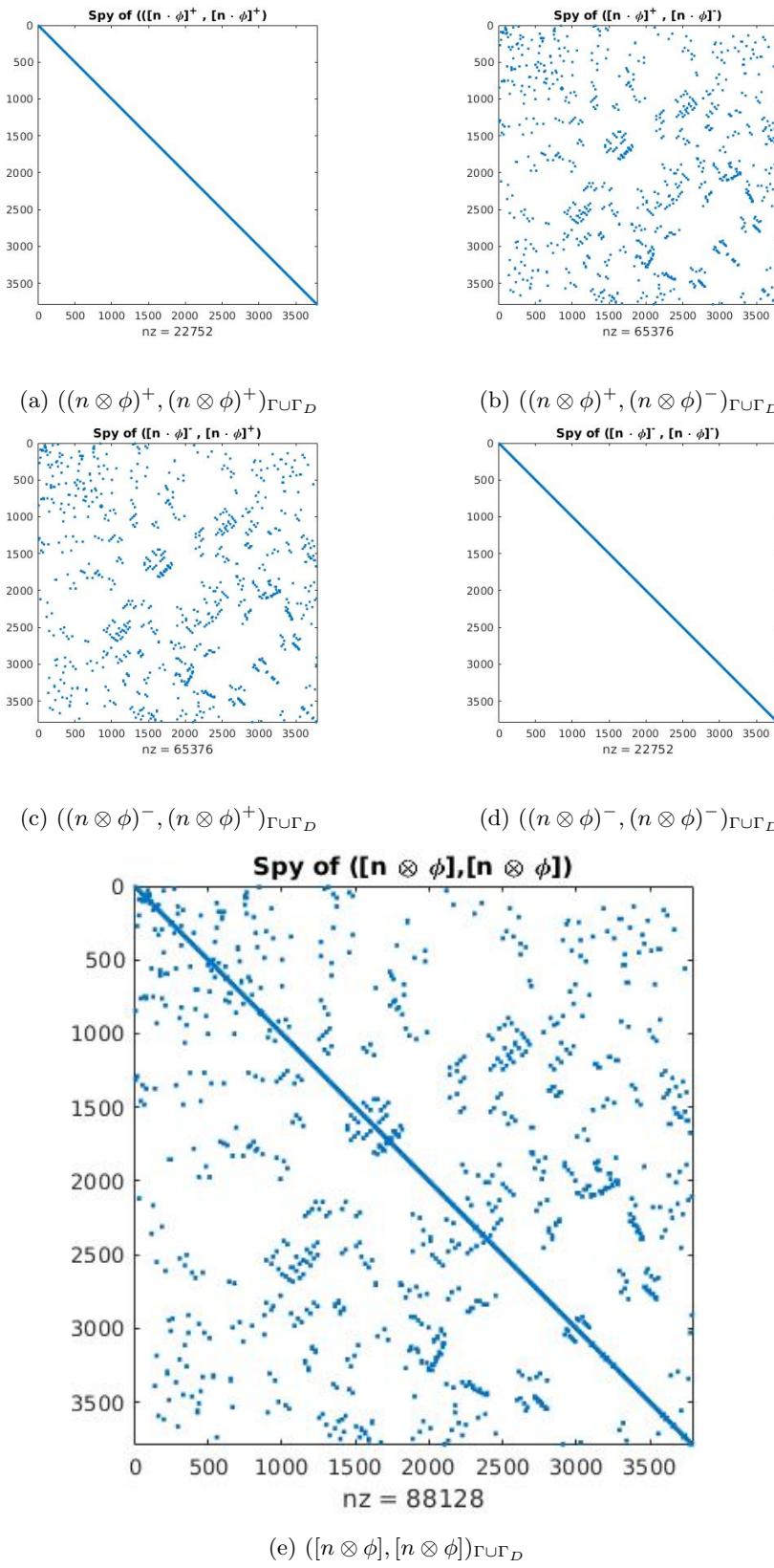
We now enter into Newton method (section 2.9.5) implemented in `newton_script`. We define again solver specific variables `tol_newton`, `max_iter_newton`, `tol_solver`, `max_iter_solver`. `tol_newton`, `max_iter_newton` are variables for ending newton method and `tol_solver`, `max_iter_solver` are variables for solver to solve h^{iter} in each newton loop. We again measure the error in L^2 and H_0 norm.

3.9 Sparsity pattern

It is the connectivity of a node with neighbouring nodes that gives rise to different discontinuous Galerkin formulations and generates different sparsity patterns. In general the flux terms are responsible for connecting to other nodes. This is also demonstrated in assembly process in section 3.5.

Table 3.5: Size and sparsity pattern of different terms

Matrix term	Size	Sparsity pattern
$(\nabla\phi, \nabla\phi)$	$\mathbb{R}^{undofs \times undofs}$	Figure 3.7
$([n \otimes \phi], [n \otimes \phi])_{\Gamma \cup \Gamma_D}$	$\mathbb{R}^{undofs \times undofs}$	Figure 3.1
$(\{\nabla\phi\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D}$	$\mathbb{R}^{undofs \times undofs}$	Figure 3.2
$(\{\psi\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}$	$\mathbb{R}^{pndofs \times undofs}$	Figure 3.3
$(-\int_T \psi \nabla \cdot \phi)$	$\mathbb{R}^{pndofs \times undofs}$	Figure 3.4
$-((u_k \cdot \nabla)\phi, \phi)$	$\mathbb{R}^{undofs \times undofs}$	Figures 3.5
$((u_k \cdot n)\phi, \phi)_{\Gamma_N}$	$\mathbb{R}^{undofs \times undofs}$	Figures 3.6
$((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N}$	$\mathbb{R}^{undofs \times undofs}$	Figures 3.8a
$(u_k \cdot n \phi, \phi^{ext})_{\partial T \setminus \Gamma_N}$	$\mathbb{R}^{undofs \times undofs}$	Figures 3.9a
$((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$	$\mathbb{R}^{undofs \times undofs}$	Figures 3.8b
$(u_k \cdot n \phi, \phi)_{\partial T \setminus \Gamma_N}$	$\mathbb{R}^{undofs \times undofs}$	Figures 3.9b

Figure 3.1: Sparsity pattern of constituents of $([n \otimes \phi], [n \otimes \phi])_{\Gamma \cup \Gamma_D}$

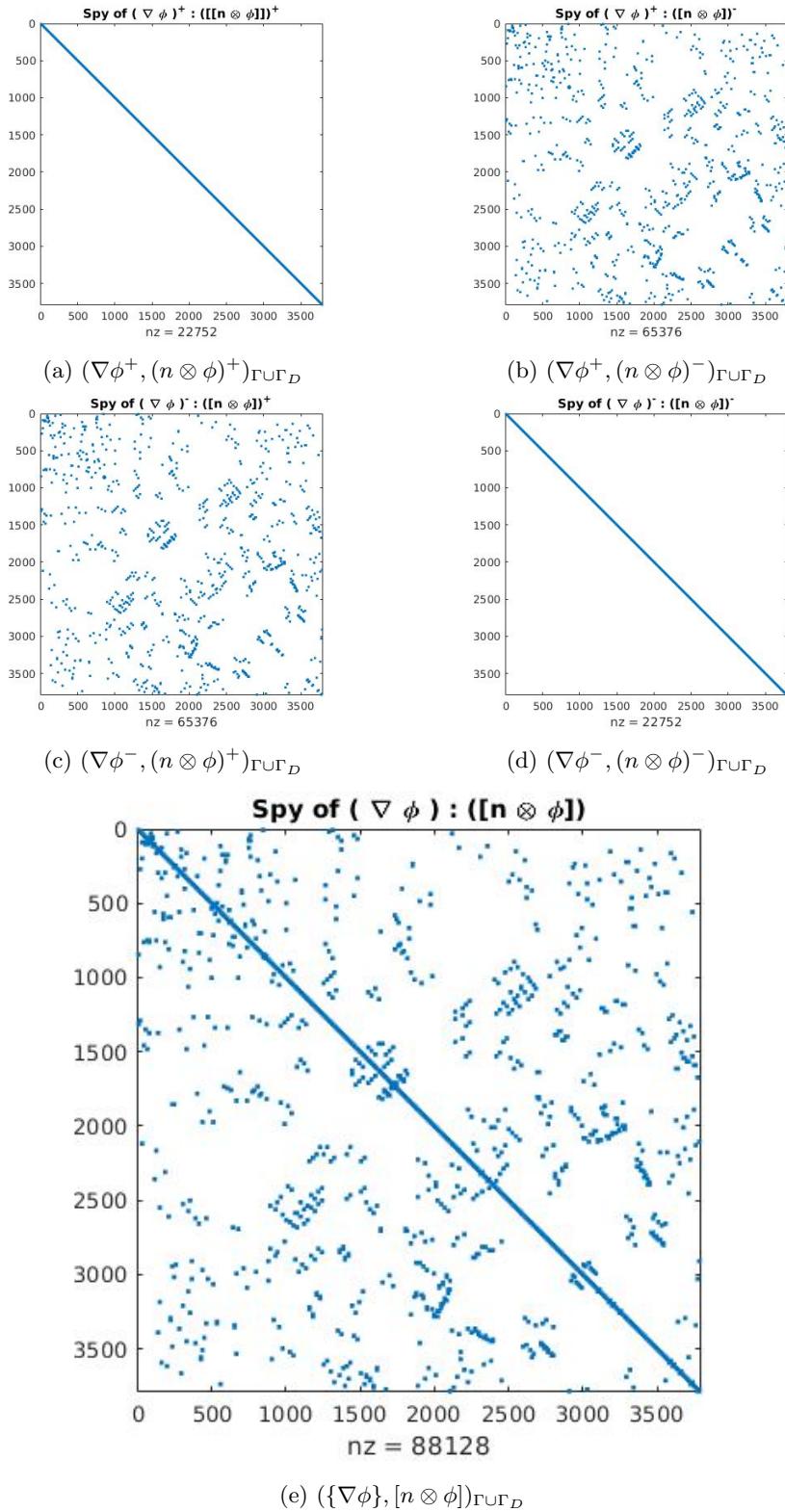
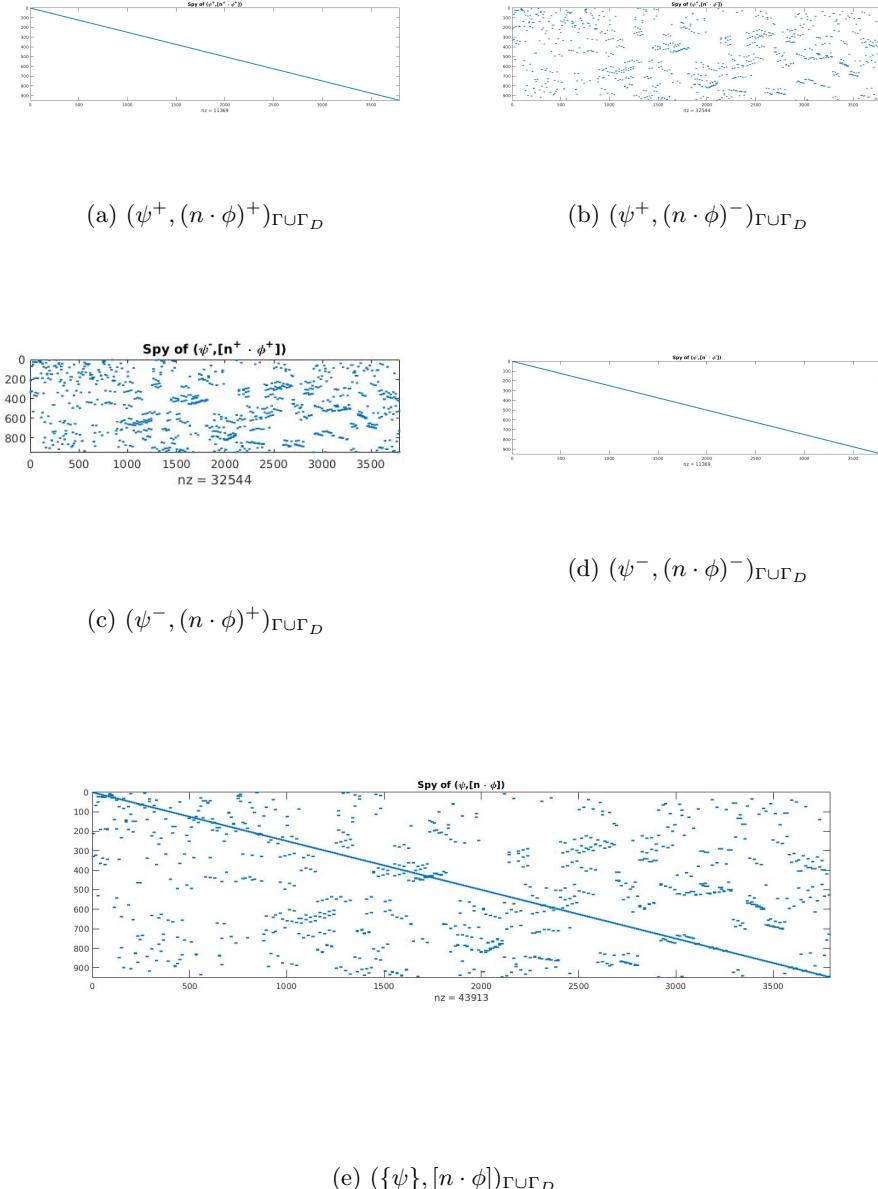


Figure 3.2: Sparsity pattern of constituents of $(\{\nabla\phi\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D}$

Figure 3.3: Sparsity pattern of constituents of $(\{\psi\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}$

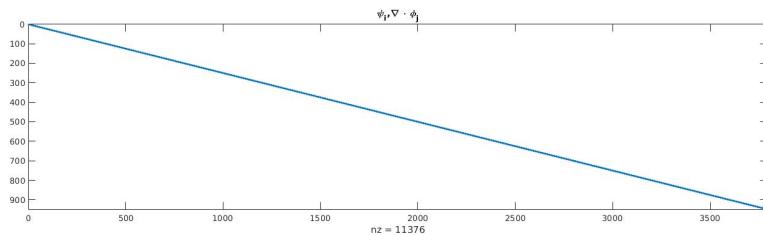


Figure 3.4: Sparsity pattern of $(\psi, \nabla \cdot \phi)$

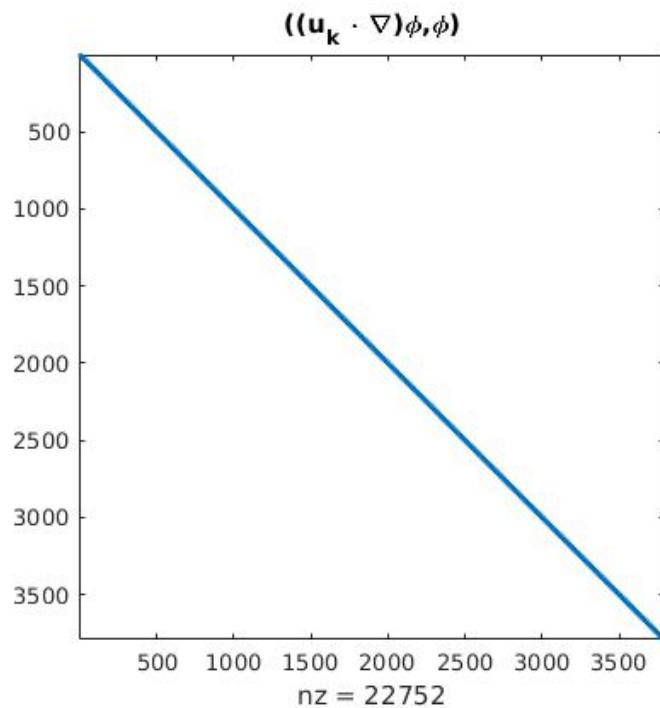
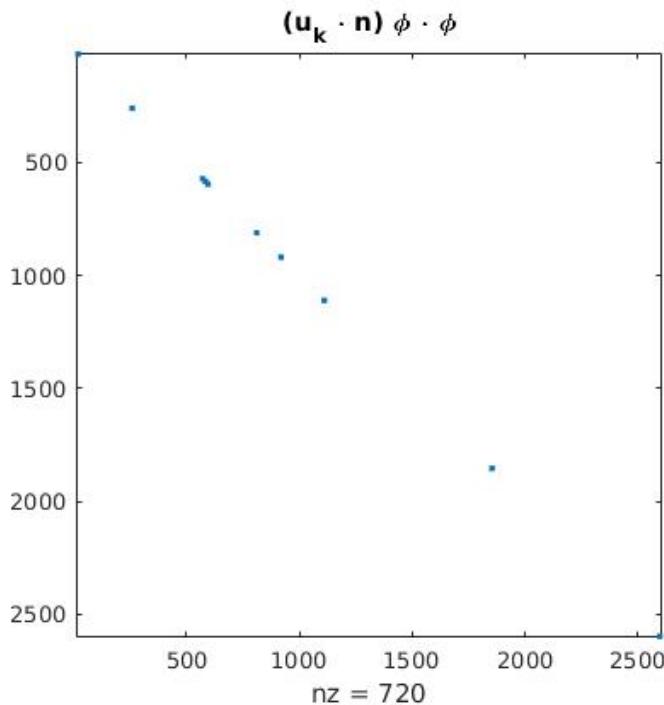
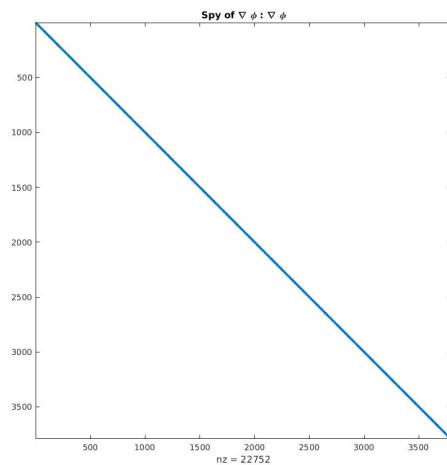
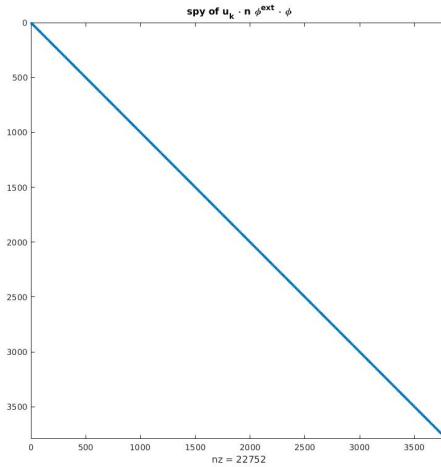
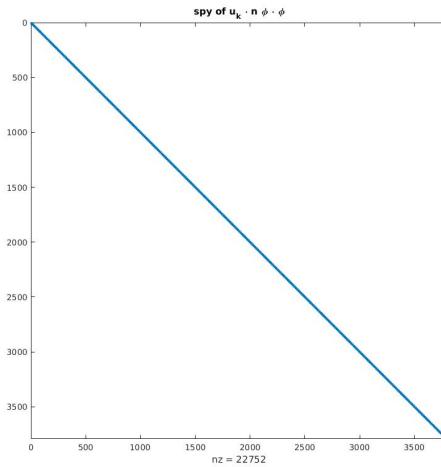
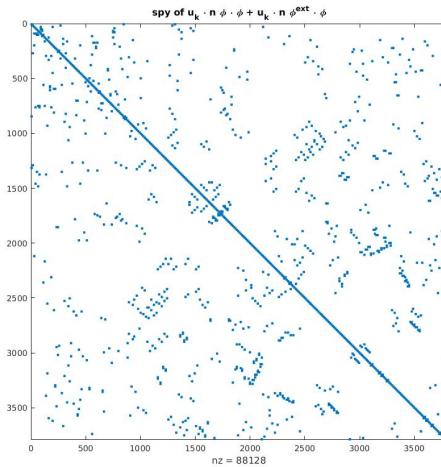
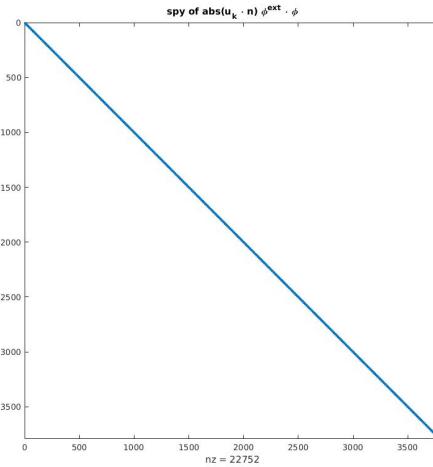
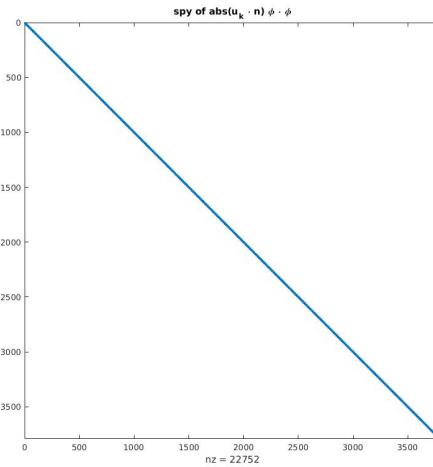
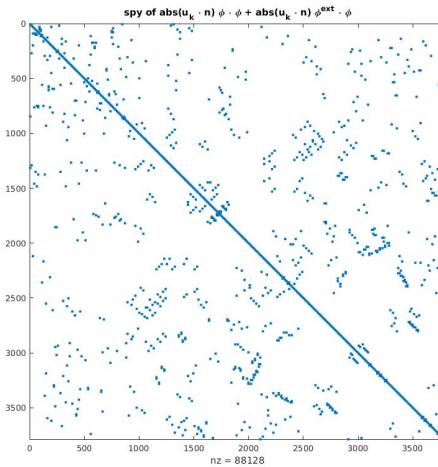


Figure 3.5: Sparsity pattern of $((u_k \cdot \nabla) \phi, \phi)$

Figure 3.6: Sparsity pattern of $((u_k \cdot n)\phi, \phi)_{\Gamma_N}$ Figure 3.7: Sparsity pattern of $(\nabla \phi, \nabla \phi)$

(a) $((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N}$ (b) $((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$ (c) $((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + ((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$ Figure 3.8: Sparsity pattern of constituents of $((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + ((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$

(a) $((\text{abs}(\mathbf{u}_k \cdot \mathbf{n})\phi, \phi^{\text{ext}})_{\partial T \setminus \Gamma_N})$ (b) $((\text{abs}(\mathbf{u}_k \cdot \mathbf{n})\phi, \phi)_{\partial T \setminus \Gamma_N})$ (c) $(\text{abs}(\mathbf{u}_k \cdot \mathbf{n})\phi, \phi^{\text{ext}})_{\partial T \setminus \Gamma_N} + (\text{abs}(\mathbf{u}_k \cdot \mathbf{n})\phi, \phi)_{\partial T \setminus \Gamma_N}$ Figure 3.9: Sparsity pattern of constituents of $(\text{abs}(\mathbf{u}_k \cdot \mathbf{n})\phi, \phi^{\text{ext}})_{\partial T \setminus \Gamma_N} + (\text{abs}(\mathbf{u}_k \cdot \mathbf{n})\phi, \phi)_{\partial T \setminus \Gamma_N}$

Chapter 4

Numerical results

This chapter discusses results obtained by performing the numerical experiments on the discontinuous Galerkin formulation of the Stokes equation and the Navier-Stokes equations.

4.1 Error definitions

The error is the difference, measured in a suitable norm, between the true solution and an approximated or computed solution. It is also a measure of how closely the implemented scheme simulates the physical nature of the problem. A correct numerical scheme should converge to the actual solution when the number of degrees of freedom are increased. The degrees of freedom can be increased either by discretizing the domain further (h -convergence) or by increasing the degree of the basis functions (p -convergence). If P_h is the computed solution and P is the true solution, the error in the W -norm is defined as,

$$P_{error,W} = \|P - P_h\|_W. \quad (4.1)$$

In the present analysis, we measure the error in the L^2 norm and the H_0 semi norm and present the results.

The L^2 norm of the error is defined as,

$$P_{error,L^2} = \int_{\Omega} |P - P_h|^2. \quad (4.2)$$

The H_0 semi norm of the error is defined as,

$$P_{error,H_0} = \sum_{k=1}^{nel} \int_{\tau_k} |\nabla P - \nabla P_h|^2. \quad (4.3)$$

We use the notations from the section 2.9.4 and the section 2.9.1.

4.2 Stokes flow

4.2.1 Properties of the stiffness matrix

We recall now some conclusions from the section 2.9.2 and the section 2.9.6. The present code provides a routine `stiffness_matrix_test` which,

1. checks whether the coefficient matrix K is symmetric and the number of non positive eigenvalues. The number of non positive eigenvalues should be same as the number of pressure degrees of freedom (section 2.9.2). The routine also provides eigenvalues and eigenvectors as output,
2. calculates the condition number of the coefficient matrix K ,
3. determines the rank of coefficient matrix K .

`stiffness_matrix_test` can also be used for the matrix A by giving the matrix A as input. In this case the matrix should be symmetric and all eigenvalues should be positive.

We consider stiffness matrix K symmetric, if $\|K - K^T\|_2 \leq tol$, where tol is some specified tolerance. Due to round off error, $tol > 0$.

4.2.2 Analytical example

The domain considered for this example is the unit square $[0,1] \times [0,1]$ in the $x-y$ plane. The boundary $x = 0$ is dirichlet boundary with inflow velocity at point $(0, y)$ as $u = (y(1-y), 0)$. The boundaries $y = 0$ and $y = 1$ are Dirichlet boundaries with no slip or zero velocity condition. The boundary $x = 1$ is a Neumann boundary with zero Neumann value i.e. $t = (0, 0)$. The source term is $f = (2\nu - 1, 0)$. The analytical solution for pressure and velocity reads as,

$$p = (1 - x), \quad (4.4)$$

$$u = (y(1 - y), 0). \quad (4.5)$$

The results of an h -convergence test with velocity polynomial degree $D = 2$ and pressure polynomial degree $D - 1 = 2$ in the L^2 norm is presented in Figures 4.1a, 4.2a, 4.1b and 4.2b and in the H_0 semi norm is presented in Figures 4.3a, 4.4a, 4.3b and 4.4b. We see almost linear convergence in L^2 norm.

We also present results of p -convergence test in L^2 norm and H_0 semi norm for velocity and in L^2 norm for pressure (Figures 4.5a, 4.5b, 4.5c). As can be seen the higher polynomial degree does not necessarily mean more accurate solution. However, the convergence rate increases with polynomial degree and below certain step size, the higher polynomial degree provides more accurate solution.

We now present additional examples and check whether the implementation of the Stokes flow is capable of reproducing the physics of the problem.

4.2.3 Lid-driven cavity problem

We next present a benchmark *CFD* problem, the Lid-driven cavity flow [1]. We solve the Stokes flow on the unit square $[0,1] \times [0,1]$ in the $x - y$ plane. On boundaries $x = 0, x = 1$ and $y = 0$, we impose no slip or zero velocity Dirichlet condition. On $y = 1$, we impose Dirichlet condition with Dirichlet velocity,

$$\begin{aligned} u &= (10x, 0) \quad \text{for } 0 \leq x \leq 0.1, \\ u &= (1, 0) \quad \text{for } 0.1 \leq x \leq 0.9, \\ u &= (10 - 10x, 0) \quad \text{for } 0.9 \leq x \leq 1. \end{aligned} \tag{4.6}$$

The results are shown in Figures 4.6, 4.7 and 4.8. The results are found to be in agreement with literature i.e. boundary layer formation at the no slip boundaries and shape of streamline.

4.2.4 Flow over cylinder

The domain considered for this example is the unit square $[0,1] \times [0,1]$ with a cut out cylinder of diameter 0.2 centered at $(0.5, 0.5)$ i.e. the center of the cylinder coincides with the center of the square in the $x - y$ plane. The boundary $x = 0$ is Dirichlet boundary with inflow velocity at point $(0, y)$ as $u = (y(1 - y), 0)$. The boundaries $y = 0$ and $y = 1$ are Dirichlet boundaries with no slip or zero velocity condition. The boundary $x = 1$ is a Neumann boundary with zero Neumann value i.e. $t = (0, 0)$. The source term is $f = (0, 0)$. Figures 4.9, 4.10 and 4.11 give physically relevant result for example, low pressure zone after cylinder, high pressure zone before cylinder and wake zone after cylinder for velocity.

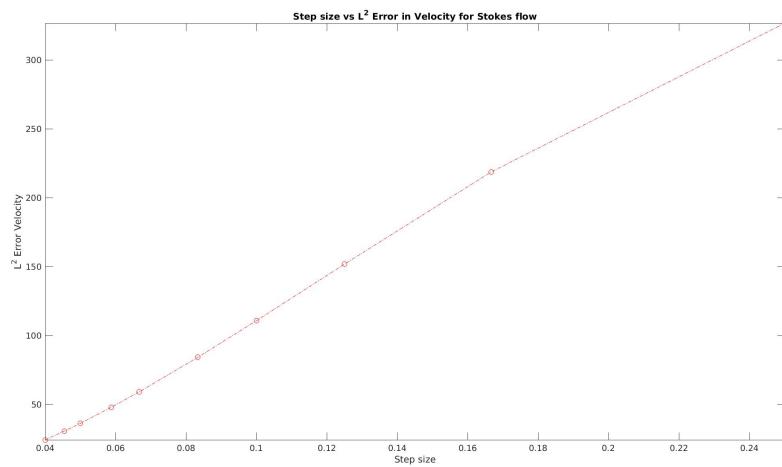
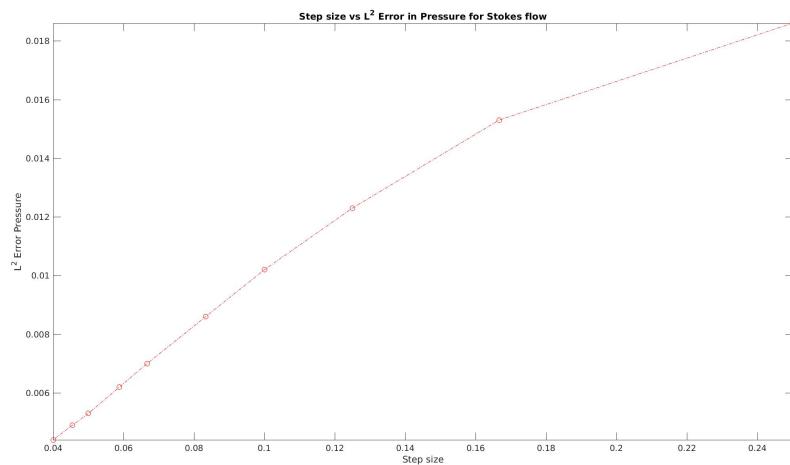
4.3 Penalty parameter

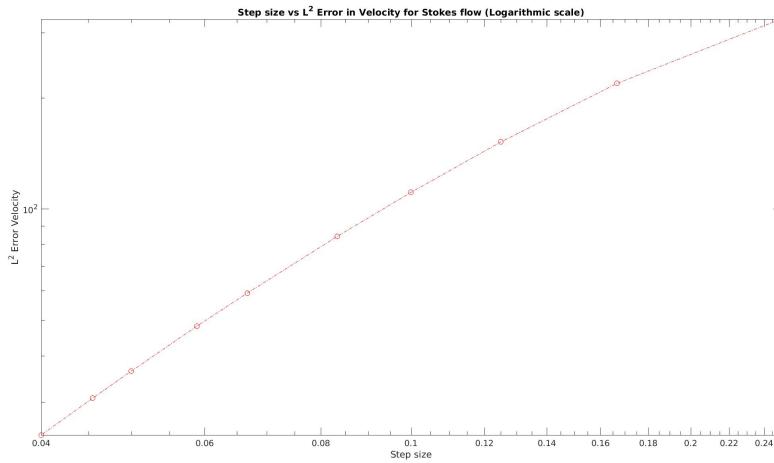
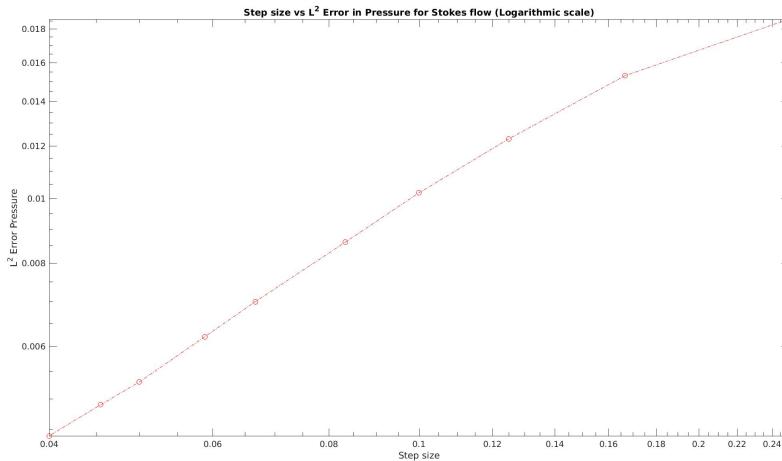
We now measure the effect of the penalty parameter on the condition number of the matrix A . While coercivity provides a lower limit for the penalty parameter, the upper limit is based on an affordable condition number of the matrix A . As shown in Figure 4.12b, the condition number of the matrix A increases with increasing penalty parameter. The condition number is measured on a unit square $[0,1] \times [0,1]$ in the $x - y$ plane with the number of intervals 10×10 . We see a linear increase in the condition number of the stiffness matrix with respect to the penalty parameter (Figure 4.12). The linear increase is in agreement with results of Montlaur et al. [2].

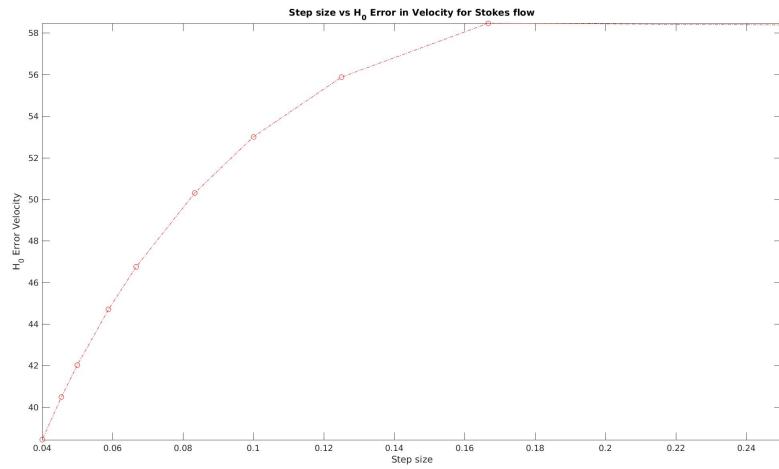
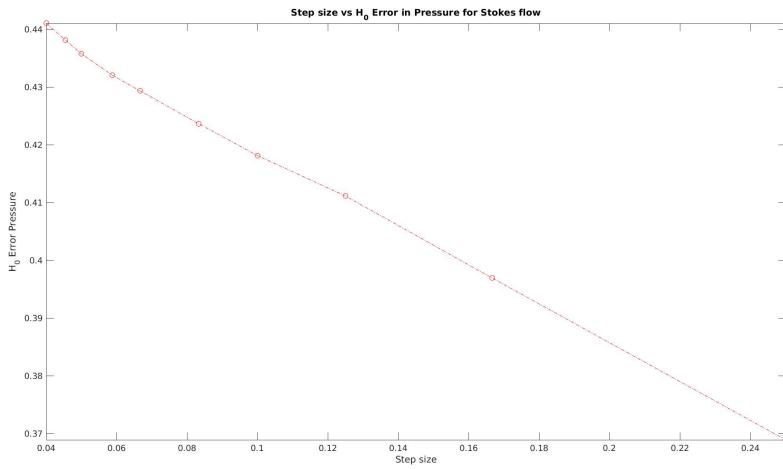
We also solve flow over cylinder problem from the section 4.2.4 with penalty parameter smaller than required to maintain coercivity. The results as can be seen in Figure 4.13 is unable to produce physically relevant flow profile.

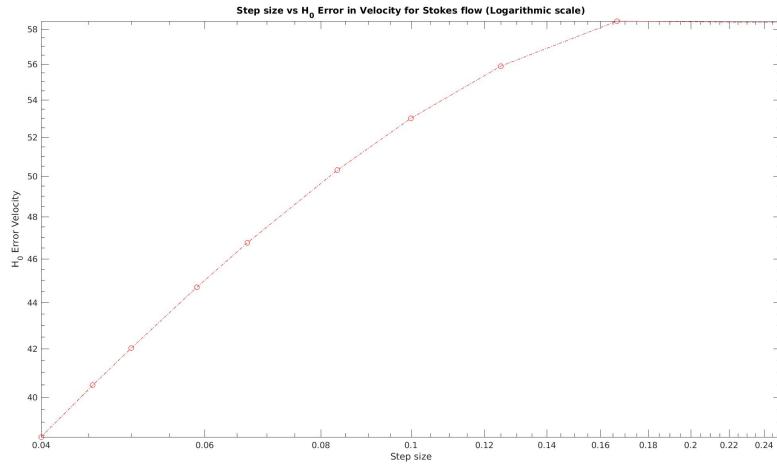
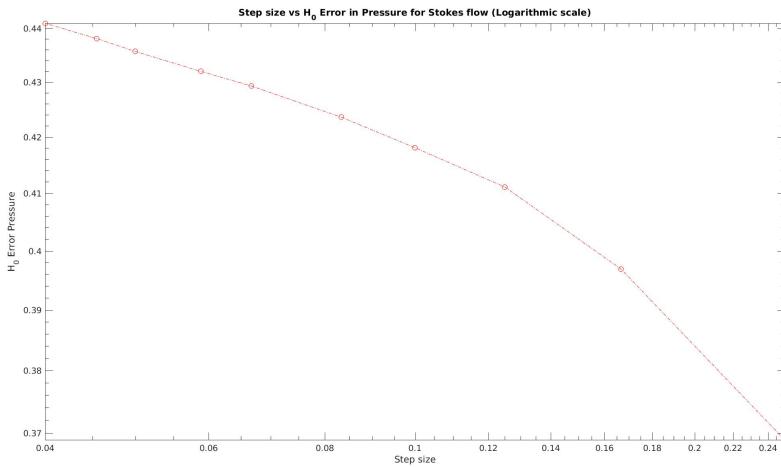
4.4 Navier-Stokes flow

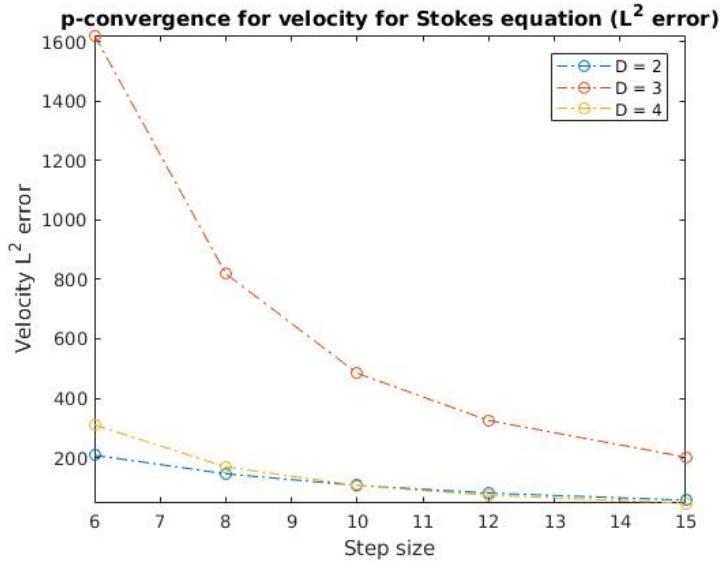
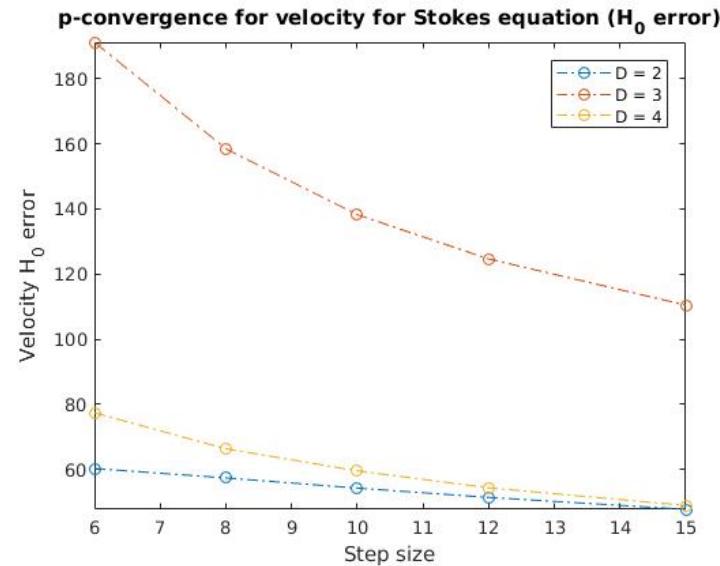
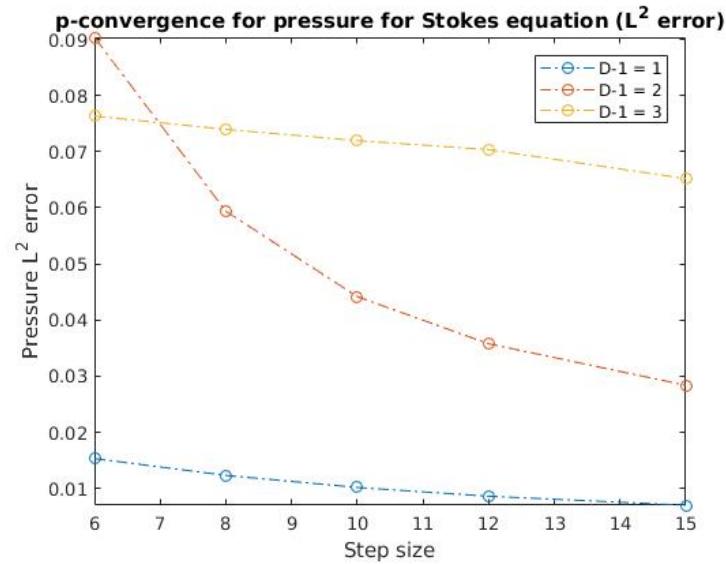
We recall that we use the initial guess from the Stokes equation and the Newton method presented in the section 2.9.5. We also note that the stiffness matrix in case of the Navier-Stokes equations is non symmetric and therefore, solvers applicable only for symmetric matrices can not be used.

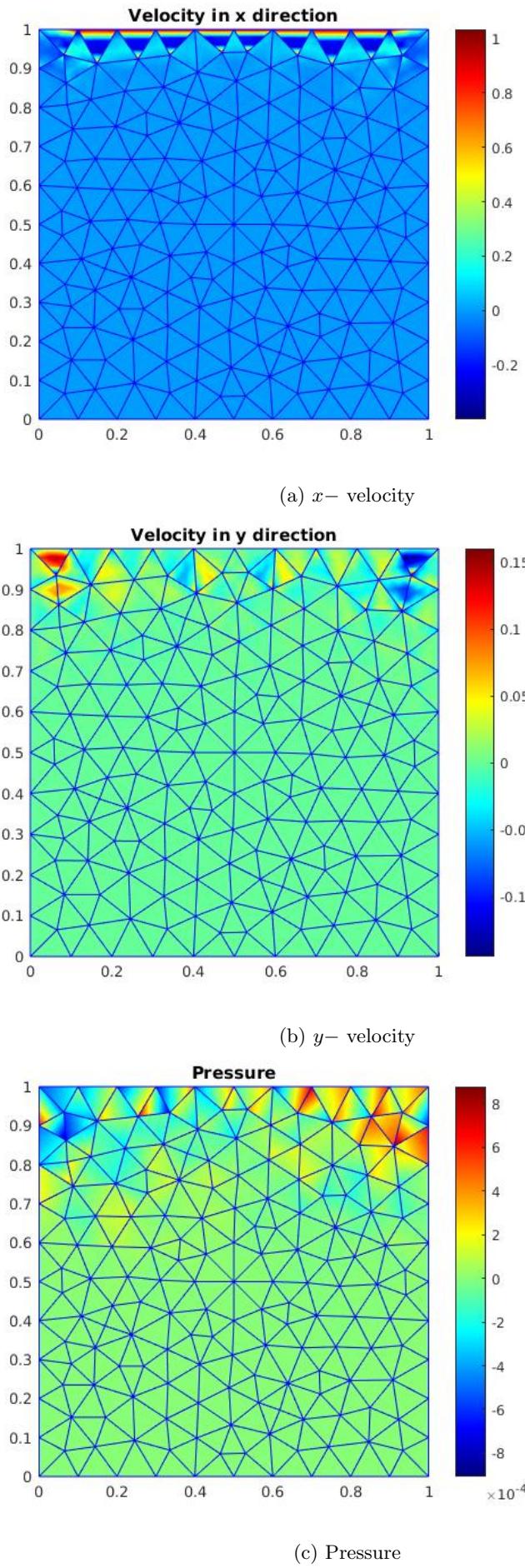
(a) h -convergence test for velocity L^2 error(b) h -convergence test for pressure in L^2 error

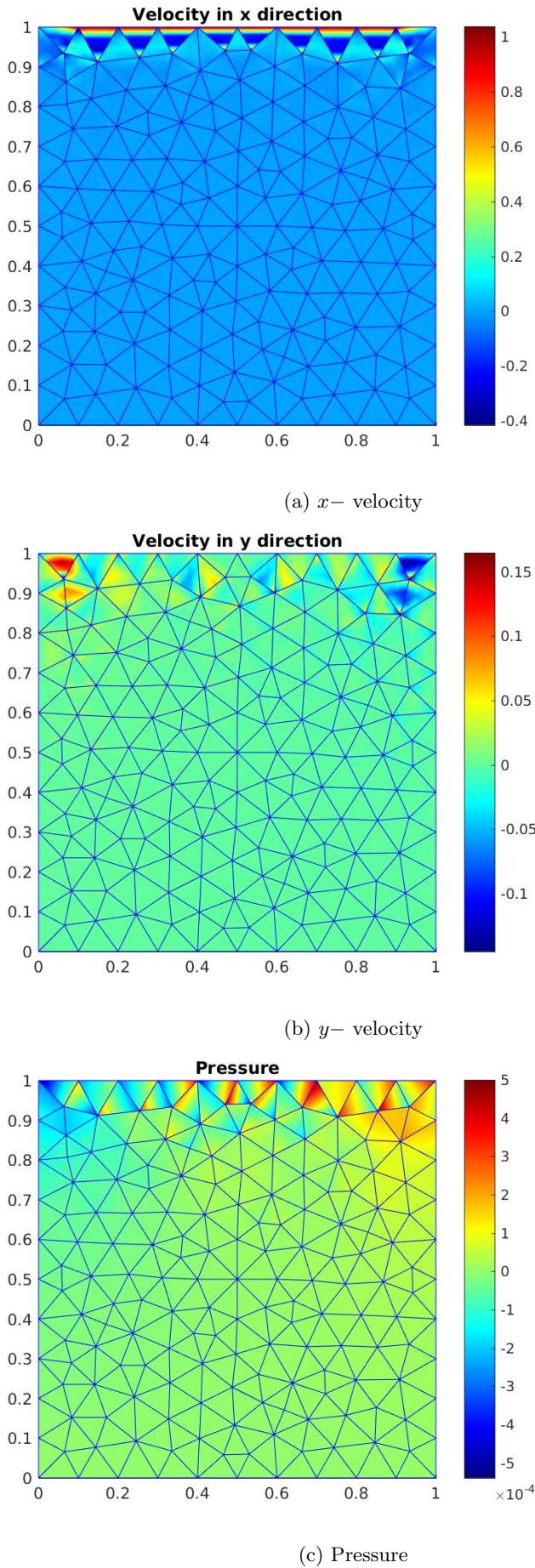
(a) h -convergence test for velocity L^2 error (Logarithmic scale)(b) h -convergence test for pressure in L^2 error (Logarithmic scale)Figure 4.2: h -convergence in L^2 norm for the Stokes flow

(a) h -convergence test for velocity H_0 error(b) h -convergence test for pressure in H_0 error

(a) h -convergence test for velocity H_0 error (Logarithmic scale)(b) h -convergence test for pressure in H_0 error (Logarithmic scale)Figure 4.4: h -convergence in H_0 norm for the Stokes flow

(a) (p -convergence for velocity in L^2 norm for Stokes flow)(b) (p -convergence for velocity in H_0 semi norm for Stokes flow)(c) (p -convergence for pressure in L^2 norm for Stokes flow)Figure 4.5: p -convergence for the Stokes flow

Figure 4.6: Lid driven cavity problem (*bicgstab* solver)

Figure 4.7: Lid driven cavity problem (*minres* solver)

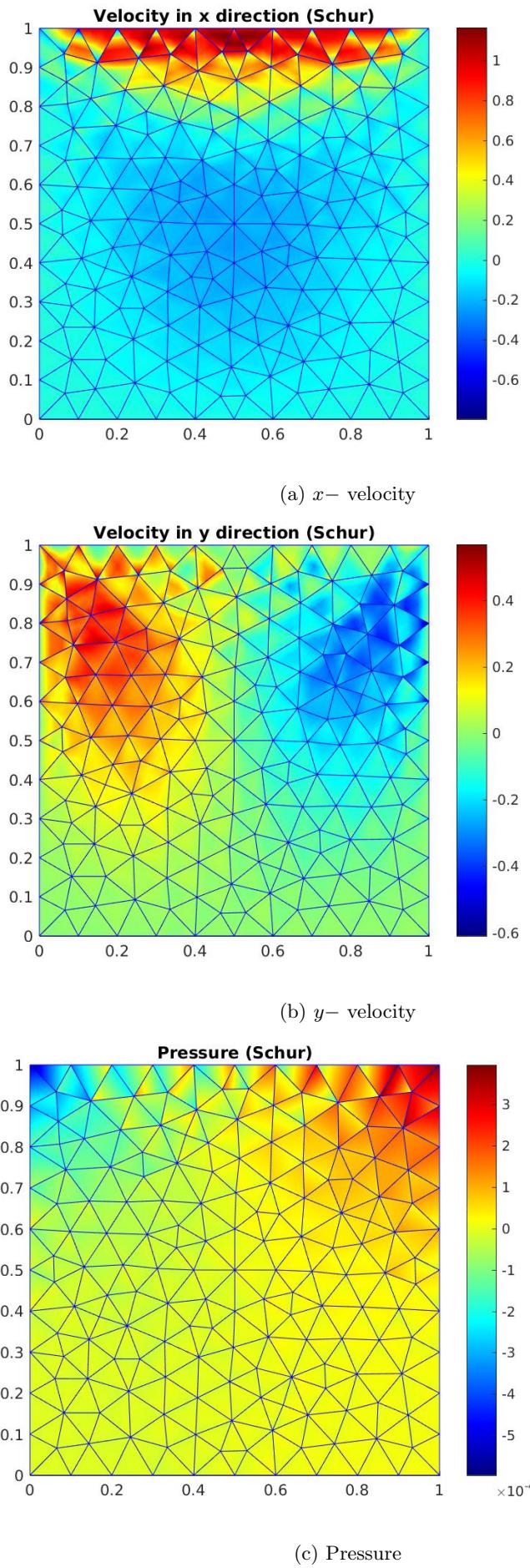
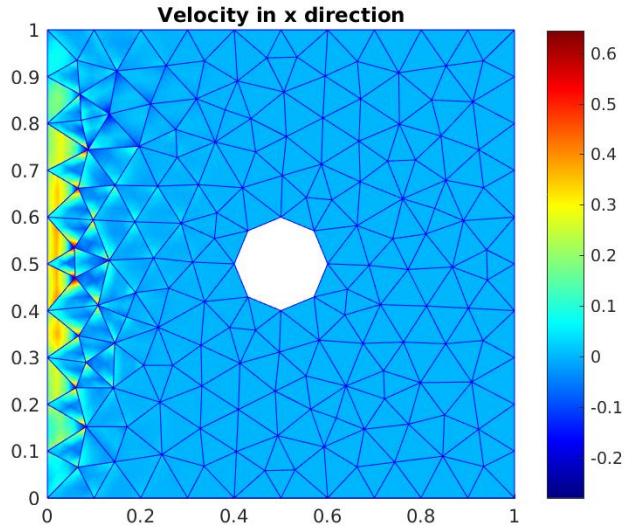
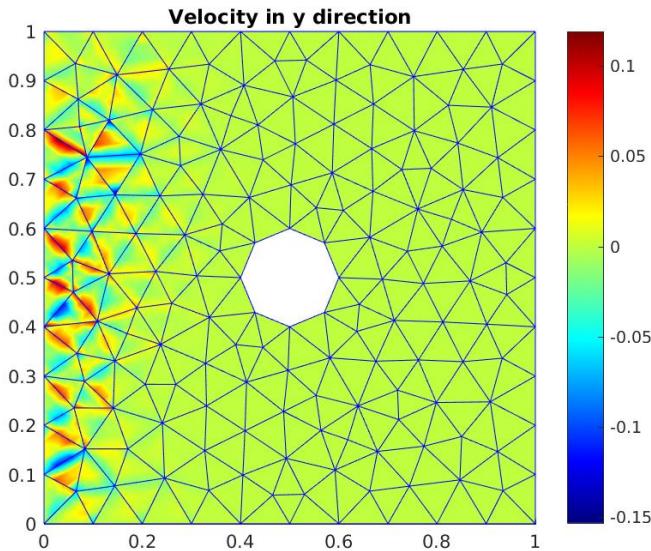
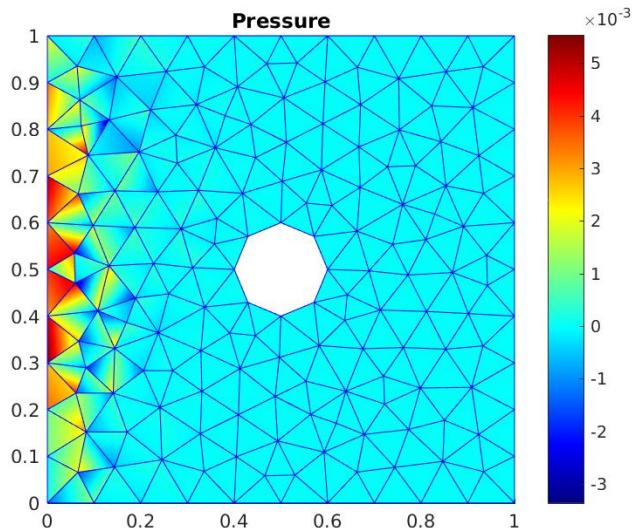
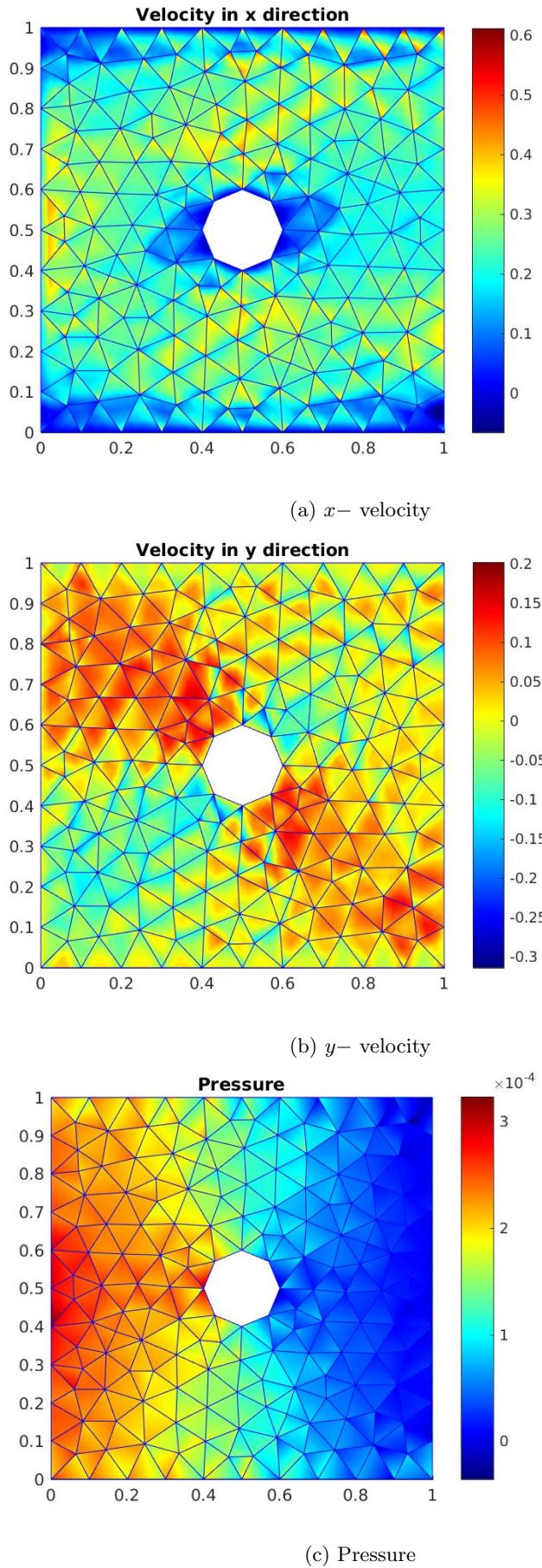


Figure 4.8: Lid driven cavity problem (Schur complement method)

(a) x - velocity(b) y - velocity

(c) Pressure

Figure 4.9: Flow over cylinder (*bicgstab* solver)

Figure 4.10: Flow over cylinder (*minres* solver)

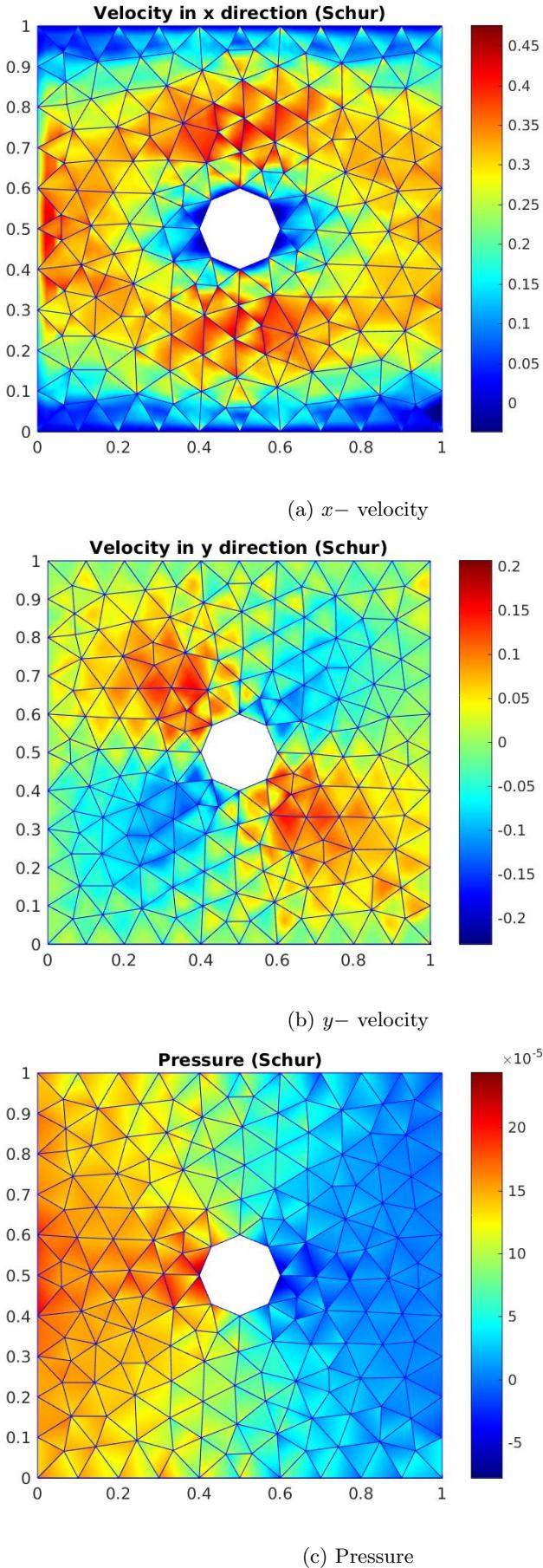
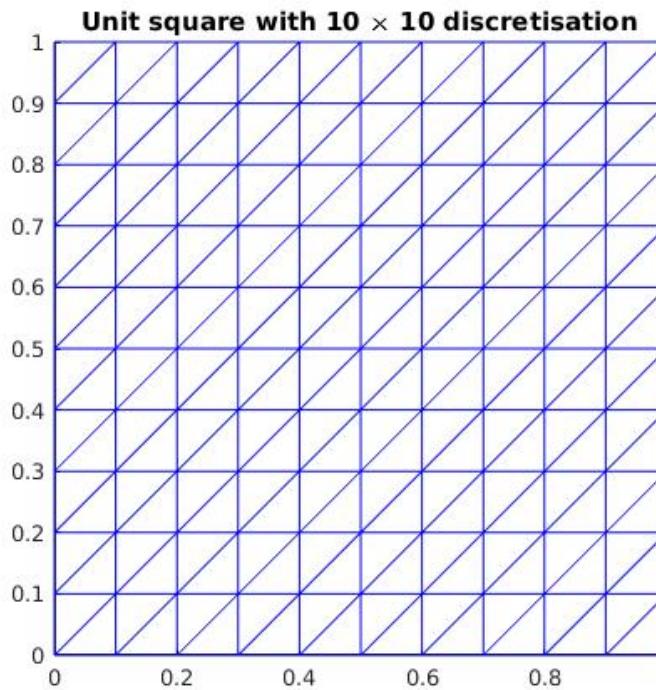
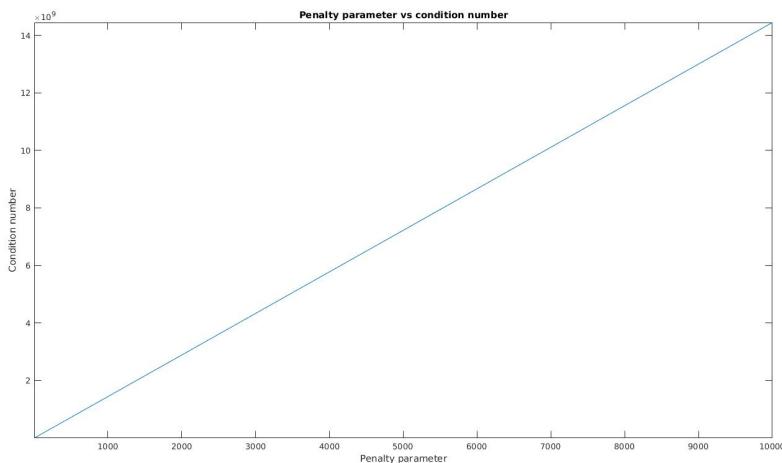
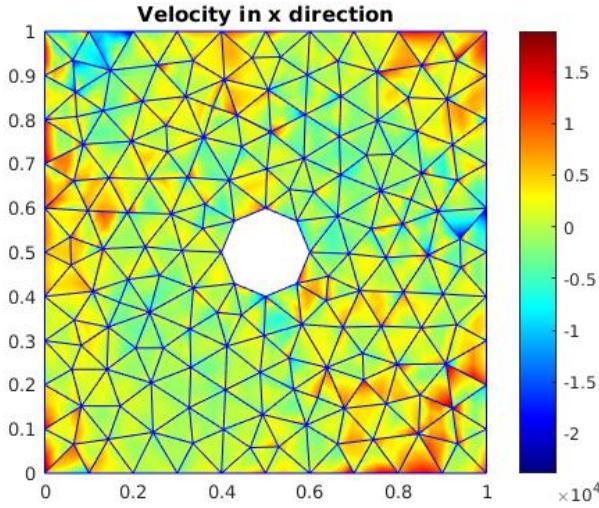


Figure 4.11: Flow over cylinder (Schur complement method)

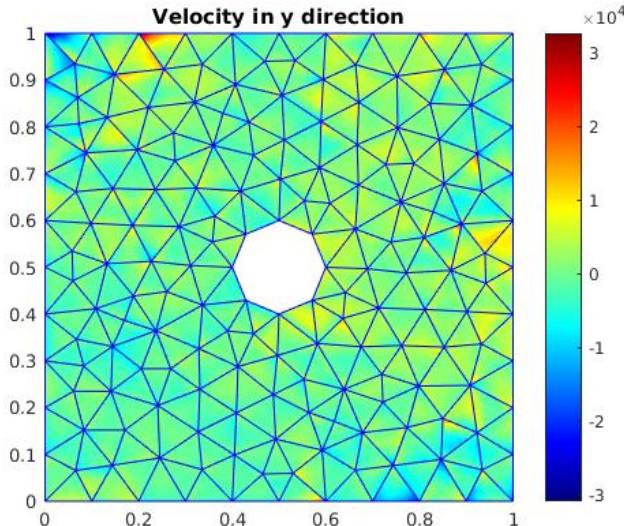
(a) Unit square with 10×10 discretisation

(b) Penalty parameter vs Condition number

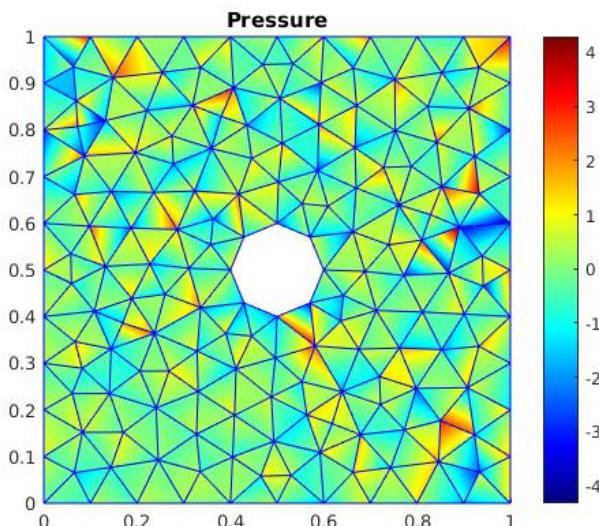
Figure 4.12: Effect of penalty parameter on condition number of the stiffness matrix



(a) x - velocity with penalty parameter smaller than minimum required for coercivity



(b) y - velocity with penalty parameter smaller than minimum required for coercivity



(c) Pressure with penalty parameter smaller than minimum required for coercivity

Figure 4.13: Flow over cylinder for smaller than minimum allowable penalty parameter

4.4.1 Analytical example

We now present an analytical example from [2]. The domain considered for this example is the unit square $[0, 1] \times [0, 1]$ in the $x - y$ plane. The boundary $x = 0$ is Dirichlet boundary with inflow velocity at point $(0, y)$ as $u = (0, 0)$. The boundaries $x = 1$, $y = 0$ and $y = 1$ are Dirichlet boundaries with no slip or zero velocity condition. The boundary $x = 0$ is a Neumann boundary with zero Neumann value i.e. $t = (0, 0)$. The source term is,

$$\begin{aligned} f = & (-4\nu(1+2y)(y^2 - 6xy^2 + 6x^2y^2 - y + 6xy \\ & - 6x^2y + 3x^2 - 6x^3 + 3x^4) + 1 - 2x \\ & + 4x^3y^2(2y^2 - 2y + 1)(y - 1)^2(-1 + 2x)(x - 1)^3, \\ & 4\nu(-1 + 2x)(x^2 - 6x^2y + 6x^2y^2 - x + 6xy \\ & - 6xy^2 + 3y^2 - 6y^3 + 3y^4) + \\ & 4x^2y^3(-1 + 2y)(y - 1)^3(2x^2 - 2x + 1)(x - 1)^2). \end{aligned} \quad (4.7)$$

The analytical solution for pressure and velocity reads as :

$$p = x(1 - x), \quad (4.8)$$

$$\begin{aligned} u = & (x^2(1 - y)^2(2y - 6y^2 + 4y^3), \\ & -y^2(1 - y)^2(2x - 6x^2 + 4x^3)). \end{aligned} \quad (4.9)$$

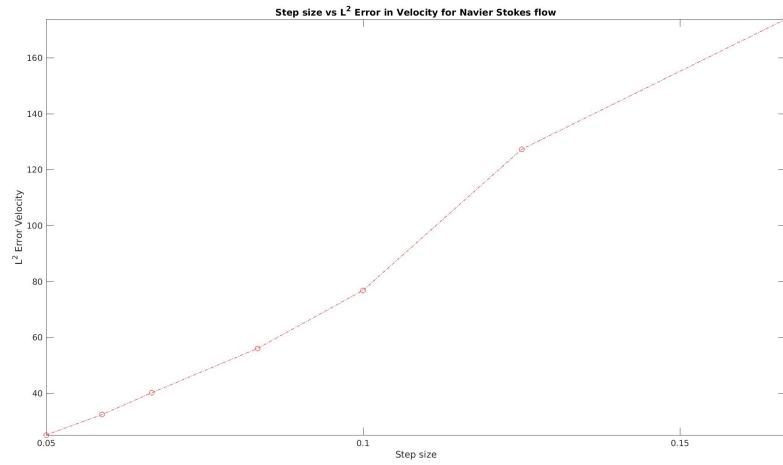
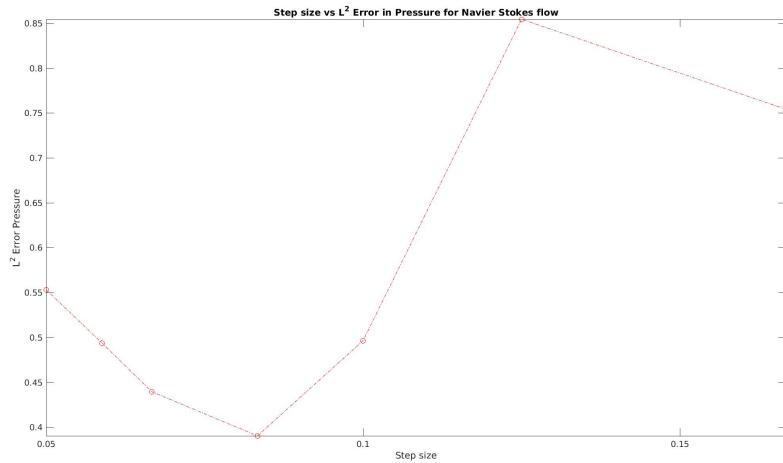
The results of an h -convergence test in the L^2 norm is presented in Figures 4.14, 4.15 and in the H_0 semi norm is presented in Figures 4.16, 4.17. The L^2 error results are found to be in agreement with Montlaur et al. [2] i.e. linear convergence.

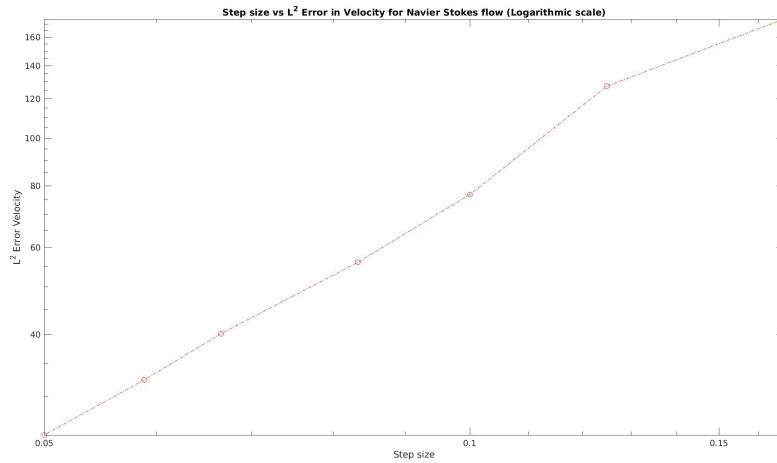
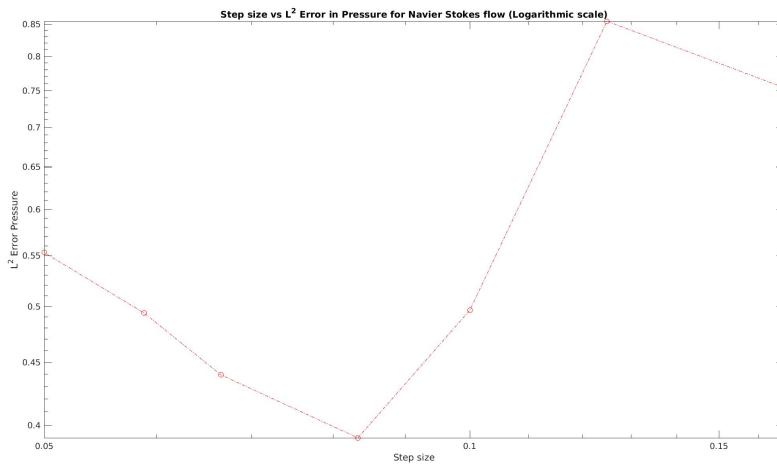
4.4.2 Lid-driven cavity problem

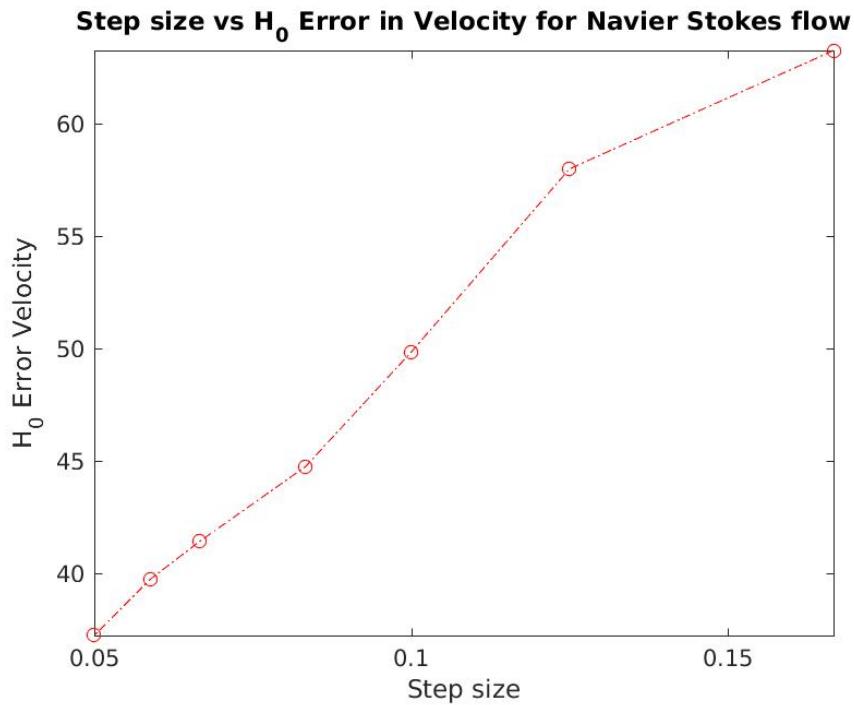
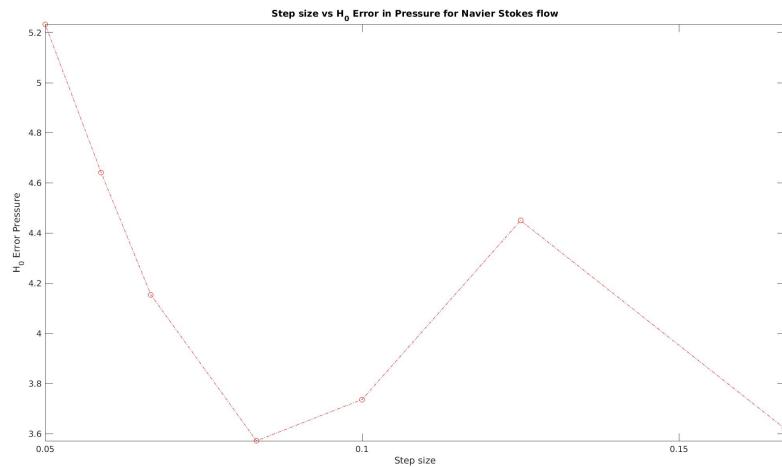
We again consider the example presented in the section 4.2.3 i.e. Benchmark *CFD* problem. The results are presented in Figures 4.18, 4.19 and 4.20. We again observe the boundary layer formation and velocity profile in agreement with literature. We also see the effect of the initial guess, the solution for the Stokes flow, calculated by different solvers.

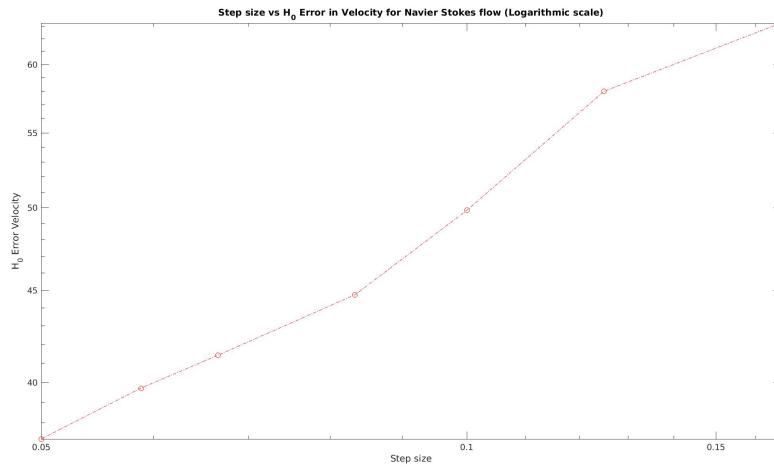
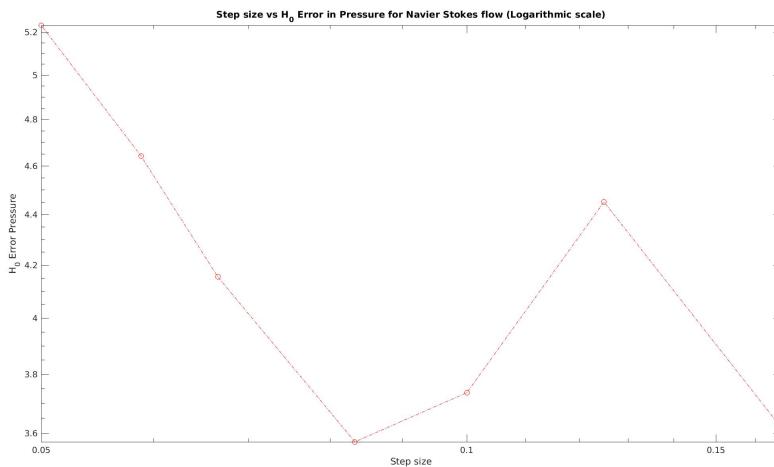
4.4.3 Flow over cylinder

We again consider the example presented in the Section 4.2.4 . However, we multiply the inlet velocity on left boundary with certain factor to regulate the Reynolds number. The results are presented in Figures 4.21, 4.22 and 4.23. We again see high pressure zone before cylinder and low pressure zone after cylinder. Also we see the wake zone in the flow. We also advice to compare the flow profile with that given by the Stokes flow to notice the effect of turbulence or specifically, velocity and pressure fluctuations. We also see the effect of the initial guess, the solution for the Stokes flow, calculated by different solvers.

(a) h -convergence test for velocity L^2 error(b) h -convergence test for pressure in L^2 errorFigure 4.14: h -convergence for the Navier Stokes flow in L^2 error

(a) h -convergence test for velocity L^2 error (Logarithmic scale)(b) h -convergence test for pressure in L^2 error (Logarithmic scale)Figure 4.15: h -convergence for the Navier Stokes flow in L^2 error (Logarithmic scale)

(a) h -convergence test for velocity H_0 error(b) h -convergence test for pressure in H_0 errorFigure 4.16: h -convergence for the Navier Stokes flow in H_0 error

(a) h -convergence test for velocity H_0 error (logarithmic scale)(b) h -convergence test for pressure in H_0 error (Logarithmic scale)Figure 4.17: h -convergence for the Navier Stokes flow in H_0 error (Logarithmic scale)

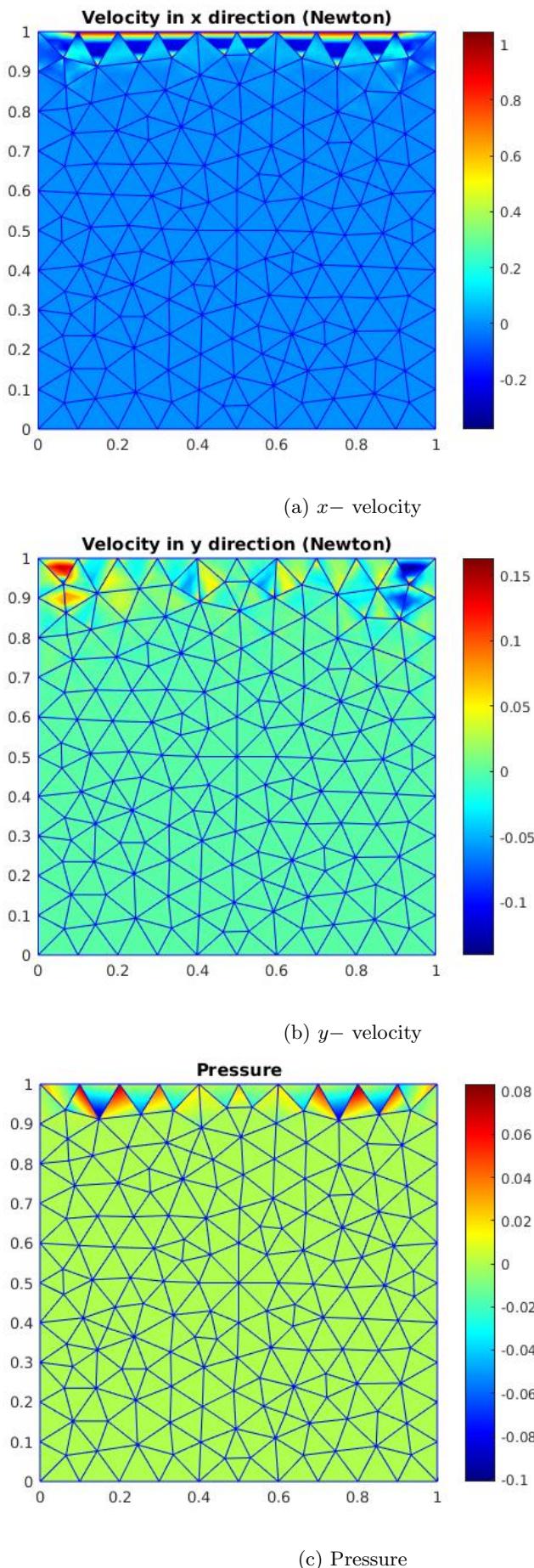


Figure 4.18: Lid driven cavity flow (Initial guess by *bicgstab* solver)

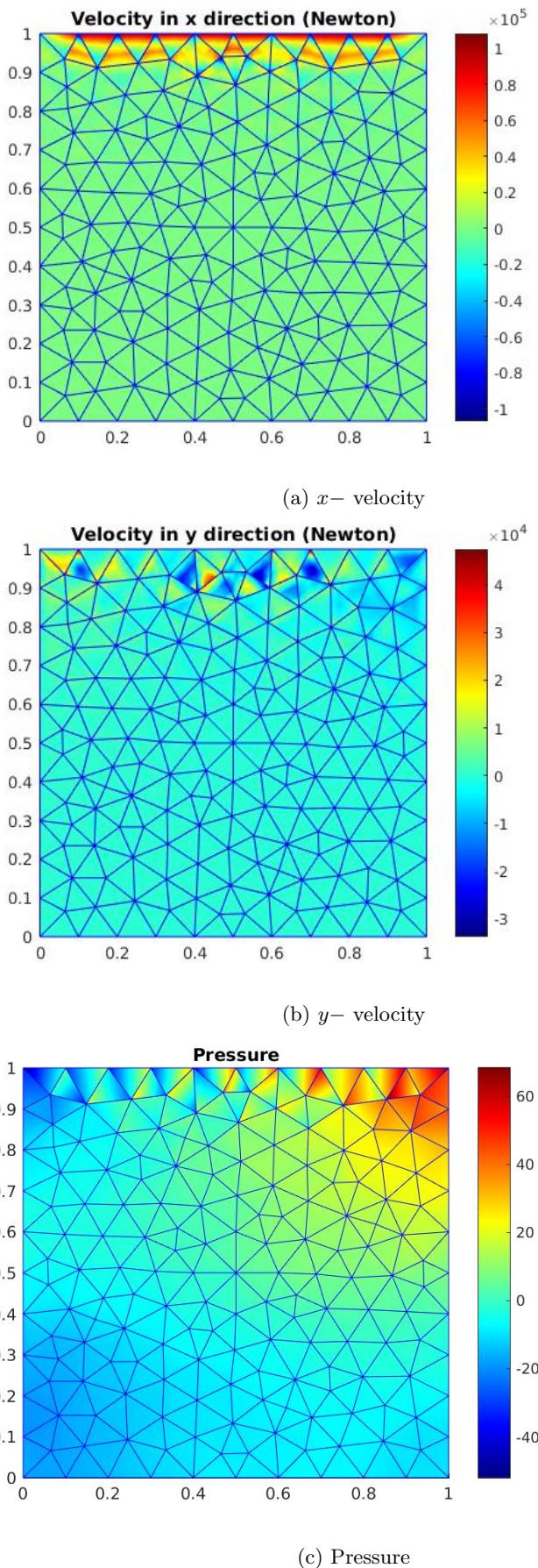


Figure 4.19: Lid driven cavity flow (Initial guess by *minres* solver)

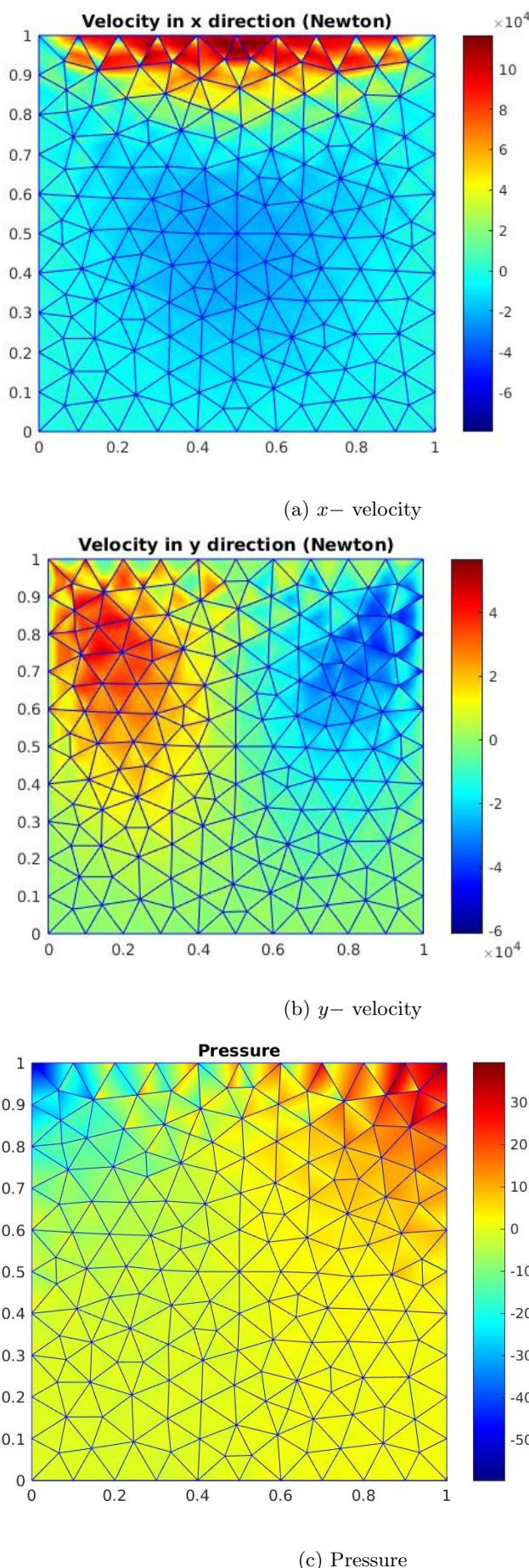
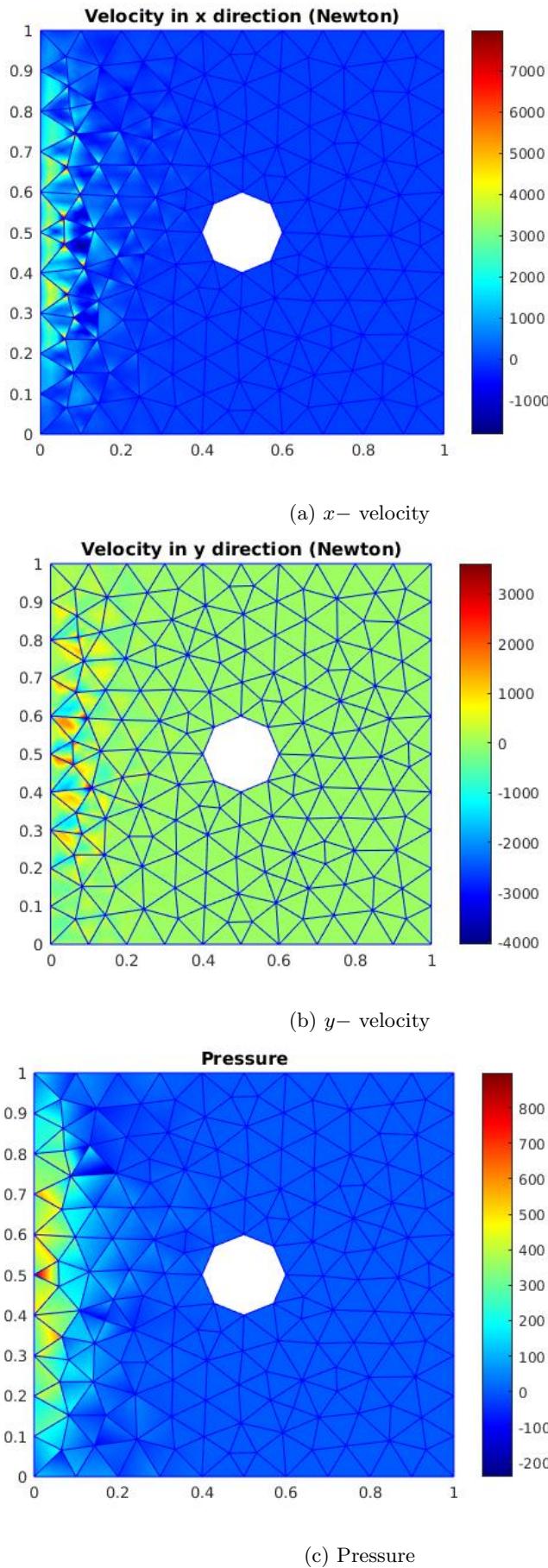


Figure 4.20: Lid driven cavity flow (Initial guess by Schur complement method)

Figure 4.21: Flow over cylinder (Initial guess by *bicgstab* solver)

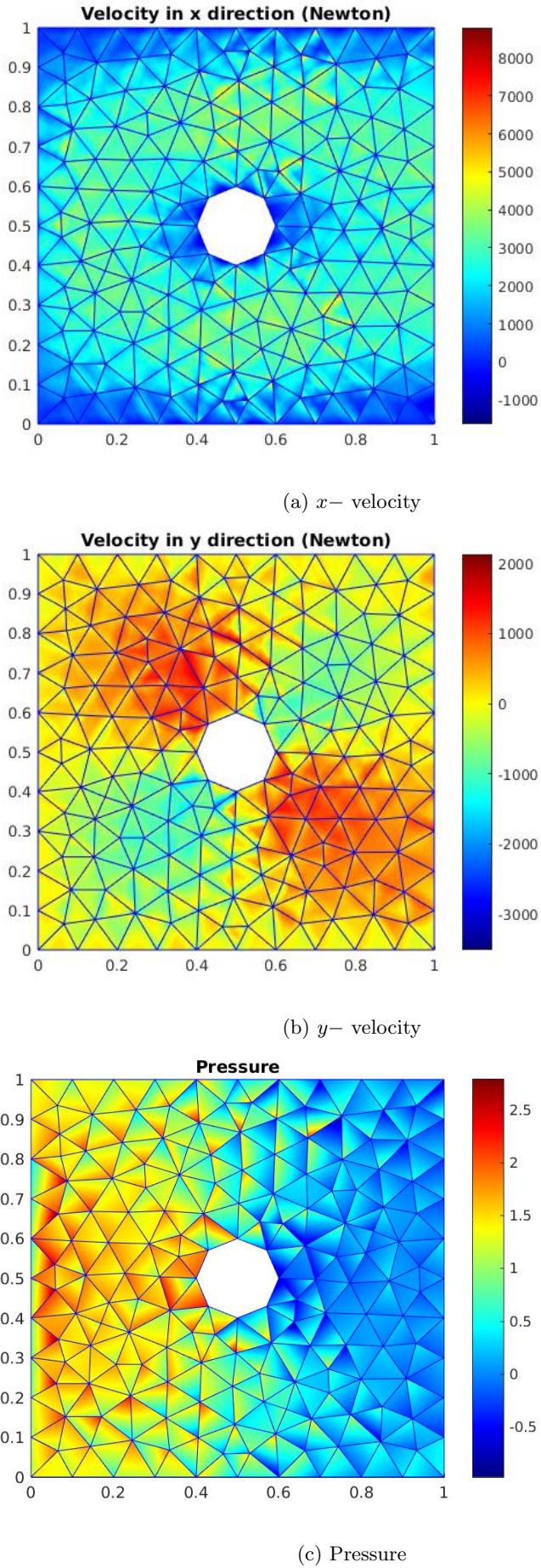


Figure 4.22: Flow over cylinder (Initial guess by minres solver)

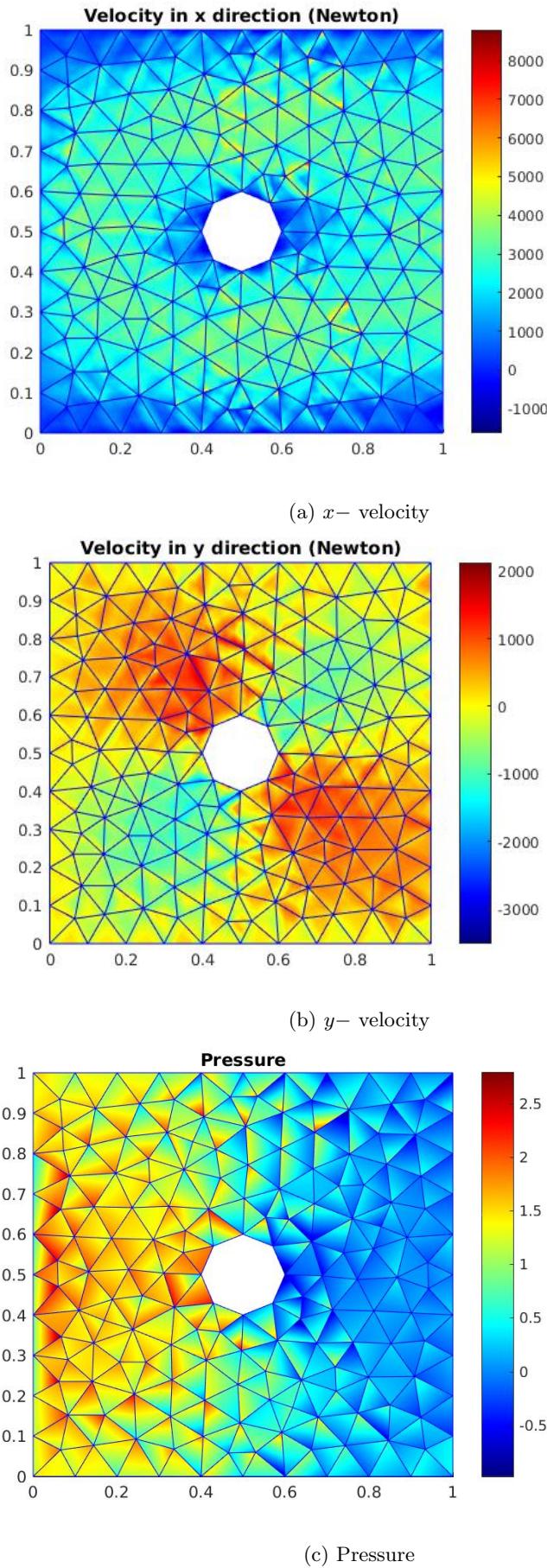


Figure 4.23: Flow over cylinder (Initial guess by Schur complement method)

4.5 Solver selection

As demonstrated earlier in the sections 4.2 and 4.4, the use of different solvers leads to different solutions. We also present now an example measuring the relative residual vs. run time for different solvers for the same problem.

We again consider the problem from the section 4.2.4. The relative residual and the run time for the same is presented in table below. Here, run time refers to time taken for solution of equation of form $AX = B$ and plotting. The matrix A and the vector B are given in assembled form as input. Since the plotting process is exactly same, the run time compares the speed of solver or method. Relative residual is measured as $\frac{\|B - AX\|_2}{\|B\|_2}$.

Solver/Method	Relative residual	Run time
Schur complement method	2.4436e-08	6.6253 Seconds
<i>minres</i>	2.4618e-05	35.7372 Seconds
<i>bicgstab</i>	9.0071e-05	58.3472 Seconds

Through the analysis of the Stokes flow, we notice that that the *bicgstab* solver stops without converging where as the *minres* solver shows convergence but is slower than the Schur complement method. The Schur complement method is very fast and accurate method.

Chapter 5

Summary, conclusion and outlook

5.1 Conclusions

After implementation of the theoretical concepts and achieving experimental outcomes we reach at the following conclusion:

1. The stiffness matrix of the Stokes equation is symmetric and has a positive definite part. Also the number of positive eigenvalues of the stiffness matrix of the Stokes equation equals the total number of velocity degrees of freedom and number of negative eigenvalues of the stiffness matrix of the Stokes equation is same as the total number of pressure degrees of freedom.
2. The Schur complement method is very useful for the Stokes equation due to efficient computation and good accuracy.
3. *minres* solver is able to solve the linear system of equations of the Stokes equation without unexpected termination of solver, i.e. the solver stops either after reaching the maximum number of iterations or specified tolerance level.
4. In contrast, *bicgstab* has shown failure to converge to desired tolerance level.
5. The condition number of the stiffness matrix of the Stokes equation increases with increase in penalty parameter. Therefore, the penalty parameter has to be small enough to limit the condition number. However, the penalty parameter has to be large enough to maintain coercivity of the bilinear form.
6. The Newton method requires solution of large system of equations in each Newton loop adding to heavy computational cost.
7. The solvers/methods which are applicable for the Saddle point problems should be used for solving the weak form of the Stokes equation.

8. The initial guess, in our case the solution obtained from the Stokes equation, is crucial for success of the Newton method.
9. The solution of the Stokes equation and the Navier Stokes equation show close to linear convergence in L^2 norm.
10. The higher polynomial degree does not always guarantee better accuracy. However, the convergence rate increases with increase in polynomial degree.

5.2 Outlook

The current work offers new areas for future development in the fields of discontinuous Galerkin method as well as reduced order modelling with special focus on the Navier Stokes equation.

The present implementation can be tested for computational efforts required for increasing Reynold's number. As the Reynold's number is increased the fluctuations in velocity and pressure become more relevant and hence turbulence plays crucial role in numerical simulations. Computationally this also means need for more accurate solver and efficient computational methods. Additionally, the present implementation can also be extended to transient cases. One of the classical applications could be to create vortex pattern for flow over cylinder.

Also as demonstrated, the solvers play very important role in prediction and reproducibility of the flow physics. Also from computational science viewpoint, solvers for saddle point problems are of special interest. The present analysis only tests few solvers/methods, however, further solvers/methods should be tested and new methods should be investigated to enhance the computational performance.

As future developments, additionally, the Stokes equation and the Navier Stokes equation can be parametrized considering fluid properties, geometry of domain or boundary conditions. While the geometric parametrisation is of great importance in technical areas such as fluid-structure interaction occurring in aerospace engineering and mechanical engineering applications, natural applications, such as porous media flows, are of potential interest due to varying pore sizes. Such parametrisation allows the approximation of the numerical solution with respect to parameter space. Affine transformation for the Stokes equation and the Empirical interpolation method (EIM) for the Navier Stokes equation can be used to evaluate parametrised solutions. Such parametric evaluations are very helpful in optimisation of the size of components or deciding optimum operating condition.

The solution of the parametrized form can be stored for reduced basis evaluations. Model order reduction with a method such as proper orthogonal decomposition or greedy algorithm can be performed. As an example, proper orthogonal decomposition sorts and segregates the stored solution based on the eigenvalues. This sorted and segregated snapshots with parametrised form can be used for the prediction of the full numerical solution. In this process the evaluations are made faster but with increase in approximation error. Hence, time saving vs. induced

error can be compared.

Appendix A

Mathematical preliminaries

We present now mathematical preliminaries, from literature, relevant to the thesis.

A.0.1 Cholesky decomposition

Every symmetric positive definite matrix can be expressed as product of lower triangular matrix and transpose of that lower triangular matrix. That is, if \mathcal{U} is symmetric positive definite matrix then,

$$\mathcal{U} = \mathcal{L}\mathcal{L}^T \quad (\text{A.1})$$

where, \mathcal{L} is lower triangular matrix. It is to be noted that \mathcal{L}^T is an upper triangular matrix.

Cholesky decomposition is useful especially when inverting a matrix in MATLAB. Since the back division operator (\backslash) recognises the lower triangular structure of matrix, the division process is faster.

We now explain the algorithm for Cholesky decomposition.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix}^T \quad . \quad (\text{A.2})$$

We see that,

$$a_{11} = l_{11}^2, \quad a_{22} = l_{21}^2 + l_{22}^2, \quad a_{33} = l_{31}^2 + l_{32}^2 + l_{33}^2 \quad (\text{A.3})$$

and

$$a_{12} = a_{21} = l_{11}l_{21}, \quad a_{13} = a_{31} = l_{11}l_{31}, \quad a_{23} = a_{32} = l_{31}l_{21} + l_{32}l_{22} \quad . \quad (\text{A.4})$$

We now see that for diagonal elements,

$$l_{kk} = \sqrt{a_{kk} - \sum_{k=1}^{j-1} l_{kj}^2} \quad (\text{A.5})$$

and for elements below diagonal,

$$l_{ik} = \frac{1}{l_{kk}} (a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj}) \quad . \quad (\text{A.6})$$

It is to be noted that similar theory is also applicable for Cholesky decomposition with upper triangular matrix instead of lower triangular matrix. Also, this algorithm can be extended to Matrix of any size.

In MATLAB the cholesky decomposition is performed by *chol*. The choice of upper triangular or lower triangular matrix can be adjusted by providing additional input argument '*lower*' or '*upper*'. More information can be found by *help* in MATLAB and MATLAB documentation.

A.0.2 Saddle point formulation

The Saddle point problem has following form,

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ \mathcal{B}_2 & \mathcal{C} \end{pmatrix} \begin{pmatrix} \mathcal{X} \\ \mathcal{Y} \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \quad (\text{A.7})$$

$$\mathcal{A} \in \mathbb{R}^{n \times n}; \mathcal{B}_1, \mathcal{B}_2 \in \mathbb{R}^{m \times n}; \mathcal{C} \in \mathbb{R}^{m \times m} \quad (\text{A.8})$$

with $n \geq m$.

We assume here that $\mathcal{A}, \mathcal{B}_1, \mathcal{B}_2$ are non zeros. Usually constituents $\mathcal{A}, \mathcal{B}_1, \mathcal{B}_2, \mathcal{C}$ satisfy one or more of following properties.

1. $\mathcal{A} = \mathcal{A}^T$ (Symmetric)
2. Symmetric part of \mathcal{A} is positive semidefinite
3. $\mathcal{B}_1 = \mathcal{B}_2 = \mathcal{B}$
4. \mathcal{C} is symmetric and positive semidefinite
5. $\mathcal{C} = 0$ (Zero matrix)

Incompressible Stokes equation is an example of saddle point problem with \mathcal{A} being symmetric positive definite matrix, $\mathcal{B}_2 = \mathcal{B}_1^T$ and $\mathcal{C} = 0$.

We consider following important factorisations :

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ \mathcal{B}_2 & \mathcal{C} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \mathcal{B}_2 \mathcal{A}^{-1} & I \end{pmatrix} \begin{pmatrix} \mathcal{A} & 0 \\ 0 & \mathcal{S} \end{pmatrix} \begin{pmatrix} I & \mathcal{A}^{-1} \mathcal{B}_1 \\ 0 & I \end{pmatrix} \quad , \quad (\text{A.9})$$

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ \mathcal{B}_2 & \mathcal{C} \end{pmatrix} = \begin{pmatrix} \mathcal{A} & 0 \\ \mathcal{B}_2 & \mathcal{S} \end{pmatrix} \begin{pmatrix} I & \mathcal{A}^{-1}\mathcal{B}_1 \\ 0 & I \end{pmatrix} , \quad (\text{A.10})$$

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ \mathcal{B}_2 & \mathcal{C} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \mathcal{B}_2\mathcal{A}^{-1} & I \end{pmatrix} \begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ 0 & \mathcal{S} \end{pmatrix} . \quad (\text{A.11})$$

Here, S is the Schur complement and $\mathcal{S} = \mathcal{C} - \mathcal{B}_2\mathcal{A}^{-1}\mathcal{B}_1$ with size $\mathcal{S} \in \mathbb{R}^{m \times m}$. I is the Identity matrix of size $I \in \mathbb{R}^{n \times n}$

It can be seen if \mathcal{C} is negative semidefinite, $B_1 = B_2^T$ and \mathcal{A} is positive definite, \mathcal{S} is negative definite. For more details on the saddle point problems we refer to literature such as Benzi M. et al[12]. We make some important observations related to Saddle point problems as follow :

1. If \mathcal{A} is symmetric positive definite, Schur complement method is very useful as matrix \mathcal{A} can be inverted efficiently with Cholesky decomposition (Section A.0.1).
2. Saddle point systems obtained in practical problems can be poorly conditioned.
3. Also number of methods such as Krylov subspace methods, Multilevel methods have been developed for saddle point problems.
4. The saddle point problem has positive as well as non positive eigenvalues. If \mathcal{A} is positive definite and \mathcal{C} is negative definite or zero matrix, number of positive eigenvalues is n and number of negative eigenvalues is m .

We now introduce Sobolev spaces and related basic definitions, Linear forms and bilinear forms. Readers are advised to refer to Hesthaven J.S. et al.[11] for further understanding.

A.0.3 Sobolev spaces

Let Ω be an open subset of \mathbb{R}^d and k' a positive integer. Let $L^2(\Omega)$ denote the space of square integrable functions on Ω .

1. The Sobolev space of order k' on Ω is defined by

$$H^{k'}(\Omega) = \{f \in L^2(\Omega) | D^\alpha f \in L^2(\Omega), |\alpha| \leq k'\}, \quad (\text{A.12})$$

where D^α is the partial derivative

$$D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_d^{\alpha_1} \dots \partial x_d^{\alpha_d}} \quad (\text{A.13})$$

in the sense of distributions for the multi-index $\alpha = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}^d$ using the notation $|\alpha| = \alpha_1 + \dots + \alpha_d$.

It holds by construction that $H^{k'+1}(\Omega) \subset H^{k'}(\Omega)$ and that $H^0(\Omega) = L^2(\Omega)$. $H^{k'}(\Omega)$ is a Hilbert space with the inner product

$$(f, g)_{H^{k'}(\Omega)} = \sum_{\alpha \in \mathbb{N}^d, |\alpha| \leq k'} \int_{\Omega} (D^\alpha f)(D^\alpha g) , \quad (\text{A.14})$$

the induced norm

$$\|f\|_{H^{k'}(\Omega)} = \sqrt{(f, f)_{H^{k'}(\Omega)}} = \sqrt{\sum_{\alpha \in \mathbb{N}^d, |\alpha| \leq k'} \int_{\Omega} |D^\alpha f|^2} , \quad (\text{A.15})$$

and the semi norm

$$|f|_{H^{k'}(\Omega)} = \sqrt{\sum_{\alpha \in \mathbb{N}^d, |\alpha| = k'} \int_{\Omega} |D^\alpha f|^2} . \quad (\text{A.16})$$

In case of the discontinuous Galerkin space we use the broken Sobolev norm (for symmetric interior penalty Galerkin), [1]

$$\|f\|_{1,h}^2 = \sum_{\tau_k \in \mathcal{T}} \|\nabla f\|_{L^2(\tau_k)}^2 + \sum_{\tau_k \in \mathcal{T}} \kappa_E \nu \|[[u]]\|_{L^2(\tau_k)}^2 , \quad (\text{A.17})$$

and inner product

$$(f, g) = \sum_{\tau_k \in \mathcal{T}} (f, g)_{L^2(\tau_k)} + \sum_{\tau_k \in \mathcal{T}} \kappa_E \nu ([u], [v])_{L^2(\tau_k)} . \quad (\text{A.18})$$

A.0.4 Basic definitions

We consider here vector space \mathbb{V} over \mathbb{R}

1. For a set $\{w_1, \dots, w_N\} \subset \mathbb{V}$ we denote by

$$\text{span}\{w_1, \dots, w_n\} = \{v \in \mathbb{V} \mid v = \sum_{n=1}^N \alpha_n w_n, \alpha_n \in \mathbb{R}\} \quad (\text{A.19})$$

the linear subspace spanned by the elements w_1, \dots, w_N .

2. The space \mathbb{V} is of finite dimension if there exists a maximal a set of linearly independent elements v_1, \dots, v_N , otherwise \mathbb{V} is of infinite dimension.

3. A norm $\|\cdot\|_{\mathbb{V}}$ on \mathbb{V} is a function $\|\cdot\|_{\mathbb{V}} : \mathbb{V} \rightarrow \mathbb{R}$ such that,

- A. $\|v\|_{\mathbb{V}} \geq 0 \forall v \in \mathbb{V}$ and $\|v\|_{\mathbb{V}} = 0$ iff $v = 0$,
- B. $\|\alpha v\|_{\mathbb{V}} = |\alpha| \|v\|_{\mathbb{V}} \forall \alpha \in \mathbb{R}, v \in \mathbb{V}$,
- C. $\|u + v\| \leq \|u\|_{\mathbb{V}} + \|v\|_{\mathbb{V}} \forall u, v \in \mathbb{V}$.

4. The pair $(\mathbb{V}, \|\cdot\|_{\mathbb{V}})$ is a normed space and we can define a distance function $d(u, v) = \|u - v\|_{\mathbb{V}}$ to measure the distance between two elements $u, v \in \mathbb{V}$.

5. A semi-norm on \mathbb{V} is a function $|\cdot|_{\mathbb{V}} : \mathbb{V} \rightarrow \mathbb{R}$ such that $|v|_{\mathbb{V}} \geq 0$ for all $v \in \mathbb{V}$ and B. and C. above are satisfied. In consequence a semi-norm is a norm if and only if $|v|_{\mathbb{V}} = 0$ implies $v = 0$.

6. Two norms $\|\cdot\|_1$ and $\|\cdot\|_2$ are equivalent if there exists two constants $C_1, C_2 > 0$ such that

$$C_1\|\cdot\|_1 \leq \|\cdot\|_2 \leq C_2\|\cdot\|_1 \quad \forall v \in V \quad (\text{A.20})$$

A.0.5 Linear forms

Let $(\mathbb{V}, \|\cdot\|_{\mathbb{V}})$ be a normed space. Then, we define the following notions.

1. A function $F : \mathbb{V} \rightarrow \mathbb{R}$ is said to be linear if,

$$F(u + v) = F(u) + F(v) \quad \forall u, v \in \mathbb{V} \quad , \quad (\text{A.21})$$

$$F(\alpha u) = \alpha F(u) \quad \forall \alpha \in \mathbb{R}, u \in \mathbb{V} \quad . \quad (\text{A.22})$$

2. F is bounded if there exists a constant $\gamma > 0$ such that,

$$|F(v)| \leq \gamma \|v\|_{\mathbb{V}} \quad \forall v \in \mathbb{V} \quad . \quad (\text{A.23})$$

3. F is continuous if for all $\epsilon > 0$ there exists a $\delta_{\epsilon} > 0$ such that

$$\|u - v\|_{\mathbb{V}} \leq \delta_{\epsilon} \Rightarrow |F(u) - F(v)| < \epsilon \quad . \quad (\text{A.24})$$

The notion of continuity and boundedness is equivalent for linear forms.

A.0.6 Bilinear forms

1. A bilinear form $a(\cdot, \cdot)$ acting on the vector spaces \mathbb{V} and \mathbb{W} is given as

$$a : \mathbb{V} \times \mathbb{W} \Rightarrow \mathbb{R} \quad , \quad (\text{A.25})$$

$$(u, v) \mapsto a(u, v) \quad (\text{A.26})$$

and is linear with respect to each of its arguments.

2. Let \mathbb{V} and \mathbb{W} be endowed with the norms $\|\cdot\|_{\mathbb{V}}$ and $\|\cdot\|_{\mathbb{W}}$. A bilinear form $a(\cdot, \cdot)$ is continuous if there exists a constant $\gamma > 0$ such that,

$$|a(u, v)| \leq \gamma \|u\|_{\mathbb{V}} \|v\|_{\mathbb{W}} \quad \forall u, v \in \mathbb{V} \quad . \quad (\text{A.27})$$

3. If $\mathbb{V} = \mathbb{W}$, a bilinear form $a(\cdot, \cdot)$ is coercive if there exists a constant $\alpha > 0$ such that,

$$a(v, v) \geq \alpha \|v\|_{\mathbb{V}}^2 \quad \forall v \in \mathbb{V} \quad . \quad (\text{A.28})$$

Coercivity of bilinear form

A bilinear form $a(u, v)$ is said to be coercive if there exists a constant $\kappa_e > 0$ such that

$$a(v, v) > \kappa_e \|v\|^2 \quad \forall \quad v \in \mathbb{V} \quad . \quad (\text{A.29})$$

Continuity of bilinear form

A bilinear form $a(u, v)$ is said to be continuous if there exists a constant $\gamma > 0$ such that

$$a(u, v) \leq \gamma \|u\| \|v\| \quad \forall \quad u, v \in \mathbb{V} \quad . \quad (\text{A.30})$$

A.0.7 Condition number

The ratio $\frac{\gamma}{\kappa_e}$ denotes condition number.

A.0.8 Important inequalities

Trace theorem

We now refer to trace inequalities presented by Rivière B.[6]. The trace inequalities are used to define restrictions of Sobolev function along the boundary of domain and used for proper treatment of boundary conditions.

If l is the length of Γ and Ar is the area of τ , $\forall \phi \in P^D(\tau)$, $\forall \Gamma \subset \partial\Omega$:

$$\|\phi\|_{L^2(\Gamma)} \leq \hat{C}_t l^{\frac{1}{2}} Ar^{-\frac{1}{2}} \|\phi\|_{L^2(\tau)} \quad , \quad (\text{A.31})$$

$$\|\phi\|_{L^2(\Gamma)} \leq C_t |h_\tau|^{-\frac{1}{2}} \|\phi\|_{L^2(\tau)} \quad , \quad (\text{A.32})$$

$$\|\nabla \phi \cdot n\|_{L^2(\Gamma)} \leq \hat{C}_t |l|^{\frac{1}{2}} |Ar|^{-\frac{1}{2}} \|\nabla \phi\|_{L^2(\tau)} \quad , \quad (\text{A.33})$$

$$\|\nabla \phi \cdot n\|_{L^2(\Gamma)} \leq C_t |h_\tau|^{-\frac{1}{2}} \|\nabla \phi\|_{L^2(\tau)} \quad . \quad (\text{A.34})$$

Here, \hat{C}_t and C_t are constants independent of h_τ and ϕ but dependent on polynomial degree D . The exact expressions for C_t are given by Warburton T. et al [15].

Cauchy-Schwarz inequality

$$\forall \quad f, g \in L^2(\Omega), \quad |(f, g)_\Omega| \leq \|f\|_{L^2(\Omega)} \|g\|_{L^2(\Omega)} \quad . \quad (\text{A.35})$$

Young's inequality

$$\forall \epsilon > 0 \forall a, b \in \mathbb{R}, \quad ab \leq \frac{\epsilon}{2} a^2 + \frac{1}{2\epsilon} b^2 \quad . \quad (\text{A.36})$$

Appendix B

Online resources

I would also like to mention below online resources which have frequently served as valuable source of information.

1. <http://mathworld.wolfram.com/>
2. <https://de.mathworks.com/products/matlab.html>
3. <http://www.ians.uni-stuttgart.de/MoRePaS/software/rbmatlab/1.13.10/doc/index.html>
4. www.cfd-online.com

Appendix C

Code access

The codes used for the present thesis is uploaded under:
https://github.com/niravshah241/master_thesis.git . It is also recommended to use `help <filename>` command for getting more information about the function definition and input/output data.

Bibliography

- [1] Montlaur A., Fernandez-Mendez S., and Huerta A. Discontinuous Galerkin methods for the Stokes equations using divergence-free approximations. *International Journal for Numerical Methods in Fluids*, 57(9):1071–1092, 2008.
- [2] Montlaur A., Fernandez-Mendez S., Peraire J., and Huerta A. Discontinuous Galerkin methods for the Navier Stokes equations using solenoidal approximations. *International Journal for Numerical Methods in Fluids*, 64(5):549–564, 2010.
- [3] Comrey A. L. The minimum residual method of factor analysis. *Psychological Reports*, 11(1):15–18, 1962.
- [4] Haasdonk B. *Reduced Basis Methods for Parametrized PDEs – A Tutorial Introduction for Stationary and Instationary Problems*. Chapter in P. Benner, A. Cohen, M. Ohlberger and K. Willcox (eds.): "Model Reduction and Approximation: Theory and Algorithms",. SIAM, Philadelphia, 2017. Pages 65-136.
- [5] Haasdonk B. Lecture notes : Reduzierte-Basis-Methoden. *University of Stuttgart*, Summer term 2010.
- [6] Riviere B. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation*. Frontiers in Applied Mathematics. Cambridge University Press, 2008.
- [7] Fritzen F. Lecture notes : Introduction to model order reduction of mechanical systems. *University of Stuttgart*, winter term 2016-2017.
- [8] White F.M. *Fluid mechanics*. Second edition, 97-99, 2002. Academic press.
- [9] van der Vorst H. A. Bi-cgstab: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.
- [10] Peraire J. and Persson P.-O. The Compact Discontinuous Galerkin (CDG) method for Elliptic problems. *SIAM Journal on Scientific Computing*, 30(4):1806–1824, 2008.
- [11] Hesthaven J. S., Rozza G., and Stamm B. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer Briefs in Mathematics. Springer, Switzerland, 1 edition, 2015.

- [12] Benzi M., Golub G. H., and Liesen J. Numerical solution of saddle point problems. *Acta Numerica*, 14:1137, 2005.
- [13] Kundu P. K. and Cohen I. M. *Fluid Mechanics*. Academic Press, 2002.
- [14] Persson P.-O., Bonet J., and Peraire J. Discontinuous Galerkin solution of the Navier Stokes equations on deformable domains. *Computer Methods in Applied Mechanics and Engineering*, 198(17):1585 – 1595, 2009.
- [15] Warburton T. and Hesthaven J. S. On the constants in hp-finite element trace inverse inequalities. *Computer methods in applied mechanics and engineering*, 192(25):2765–2773, 2003.