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 of Stuttgart,
Stuttgart, Deutschland (Germany)

Co-advisor: Prof. Gianluigi Rozza,
Scuola Internazionale Superiore di Studi Avanzati,
Trieste, Italy

Co-advisor: Dr. Martin Hess,
Scuola Internazionale Superiore di Studi Avanzati,
Trieste, Italy

February 26, 2018

0.1 Acknowledgement

The presented in this thesis is a result of the work carried out between Spetember 2017 - March 2018 at the Institute of Applied Analysis and Numerical Simulation, Universität of Stuttgart and 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 Bernard Haasdonk: I experienced the freedom I needed to be creative while at the same time I was always helped to continue research in case of diffi- culties.
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 am really looking forward to continue these valuable working relationships.
To my colleagues: I would also thank colleagues for their positive interest in my work and very positive support for the work.
To my parents: Last but most importantly, I would like to thank my parents for their encouragement, infusing new energy to continue with my work.

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.

Hiermit versichere ich, dass ich die vorliegende Seminararbeit selbstndig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wrtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten bernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Signature/Unterschrift

Place,Date/Ort, Datum

0.3 List of symbols

Following is the list of symbols used throughtout 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 corressponding 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 viscocity
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	Interelement 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 3.1)
$\mathbb{V}$	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 3.10)
$\hat{T}$	Reference triangle
J_k	Jacobian of τ_k (Equation 3.10)
$\hat{X}$	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 3.10)
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 strong form in weak formulation of the Navier Stoks equation (Equation (3.53))
b	Term representing gradient operator in weak formulation of the Navier Stoks equation (Equation (3.53))
c	Term representing non linear terms in weak formulation of the Navier Stoks equation (Equation (3.53))
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 (3.53))
u^{ext}	External trace of Velocity (Equation 3.52)
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 equatio (3.53)
F_2	Discrete form of right hand side of equation (3.29)
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

3.1 Element self (+) and neighbouring element (-) 20
3.2 Continuous domain (left) and discretised domain or grid (right) . . . 21
3.3 Finite Element nodes on triangle for polynomials of different degrees 23

4.1 $((n \otimes \phi)^+, (n \otimes \phi)^+)_{\Gamma \cup \Gamma_D}$ 46
4.2 $((n \otimes \phi)^+, (n \otimes \phi)^-)_{\Gamma \cup \Gamma_D}$ 46
4.3 $((n \otimes \phi)^-, (n \otimes \phi)^+)_{\Gamma \cup \Gamma_D}$ 46
4.4 $((n \otimes \phi)^-, (n \otimes \phi)^-)_{\Gamma \cup \Gamma_D}$ 46
4.5 Sparsity patterns of constituents of $([n \otimes \phi], [n \otimes \phi])_{\Gamma \cup \Gamma_D}$ 46
4.6 46
4.7 $(\nabla \phi^+, (n \otimes \phi)^+)_{\Gamma \cup \Gamma_D}$ 47
4.8 $(\nabla \phi^+, (n \otimes \phi)^-)_{\Gamma \cup \Gamma_D}$ 47
4.9 $(\nabla \phi^-, (n \otimes \phi)^+)_{\Gamma \cup \Gamma_D}$ 47
4.10 $(\nabla \phi^-, (n \otimes \phi)^-)_{\Gamma \cup \Gamma_D}$ 47
4.11 Sparsity patterns of constituents of $(\{\nabla \phi\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D}$ 47
4.12 $(\psi^+, (n \cdot \phi)^+)_{\Gamma \cup \Gamma_D}$ 48
4.13 $(\psi^+, (n \cdot \phi)^-)_{\Gamma \cup \Gamma_D}$ 48
4.14 $(\psi^-, (n \cdot \phi)^+)_{\Gamma \cup \Gamma_D}$ 48
4.15 $(\psi^-, (n \cdot \phi)^-)_{\Gamma \cup \Gamma_D}$ 48
4.16 Sparsity patterns of constituents of $(\{\psi\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}$ 48
4.17 Sparsity pattern of $(\psi, \nabla \cdot \phi)$ 49
4.18 Sparsity pattern of $((u_k \cdot \nabla)\phi, \phi)$ 49
4.19 Sparsity pattern of $((u_k \cdot n)\phi, \phi)_{\Gamma_N}$ 50
4.20 Sparsity pattern of $(\nabla \phi, \nabla \phi)$ 50
4.21 Sparsity patterns of constituents of $((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + ((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$ 51
4.22 Sparsity patterns of constituents of $(abs(u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + (abs(u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$ 52

5.5 $x-$ velocity (bicgstab solver) 60
5.6 $y-$ velocity (bicgstab solver) 60
5.7 Pressure (bicgstab solver) 60
5.8 60
5.9 $x-$ velocity (minres solver) 61
5.10 $y-$ velocity (minres solver) 61
5.11 Pressure (minres solver) 61
5.12 61

5.13 $x-$ velocity (Schur complement method) 62
5.14 $y-$ velocity (Schur complement method) 62
5.15 Pressure (Schur complement method) 62
5.16 62
5.17 $x-$ velocity (bicgstab solver) 63
5.18 $y-$ velocity (bicgstab solver) 63
5.19 Pressure (bicgstab solver) 63
5.20 63
5.21 $x-$ velocity (minres solver) 64
5.22 $y-$ velocity (minres solver) 64
5.23 Pressure (minres solver) 64
5.24 64
5.25 $x-$ velocity (Schur complement method) 65
5.26 $y-$ velocity (Schur complement method) 65
5.27 Pressure (Schur complement method) 65
5.28 65
5.29 67
5.30 69
5.31 70
5.32 71
5.33 72
5.34 $x-$ velocity (Initial guess by bicgstab solver) 73
5.35 $y-$ velocity (Initial guess by bicgstab solver) 73
5.36 Pressure (Initial guess by bicgstab solver) 73
5.37 73
5.38 $x-$ velocity (Initial guess by minres solver) 74
5.39 $y-$ velocity (Initial guess by minres solver) 74
5.40 Pressure (Initial guess by minres solver) 74
5.41 74
5.42 $x-$ velocity (Initial guess by Schur complement method) 75
5.43 $y-$ velocity (Initial guess by Schur complement method) 75
5.44 Pressure (Initial guess by Schur complement method) 75
5.45 75
5.46 $x-$ velocity (Initial guess by bicgstab solver) 76
5.47 $y-$ velocity (Initial guess by bicgstab solver) 76
5.48 Pressure (Initial guess by bicgstab solver) 76
5.49 76
5.50 $x-$ velocity (Initial guess by minres solver) 77
5.51 $y-$ velocity (Initial guess by minres solver) 77
5.52 Pressure (Initial guess by minres solver) 77
5.53 77
5.54 $x-$ velocity (Initial guess by Schur complement method) 78
5.55 $y-$ velocity (Initial guess by Schur complement method) 78
5.56 Pressure (Initial guess by Schur complement method) 78
5.57 78

List of Tables

4.1 Fields of `params` 36
4.2 Fields of `paramsP` 36
4.3 Fields of `grid` 37
4.4 Some other variables 37
4.5 Size and sparsity patterns of different terms 45

Contents

0.1 Acknowledgement . . . 2
0.2 Declaration/Eigenständigkeitserklärung . . . 3
0.3 List of symbols . . . 4

1 Introduction **13**
1.1 Overview . . . 13

2 Perspective and Formulation **15**
2.1 Derivation . . . 15
2.1.1 DNS . . . 17

3 Discretisation and function spaces **19**
3.1 Grid geometry . . . 19
3.2 Grid parameters . . . 20
3.3 Discontinuous Galerkin method . . . 21
3.4 Basis function . . . 22
3.4.1 Nodal Basis Functions . . . 24
3.4.2 Orthonormal Basis Functions . . . 24
3.5 Global and local co-ordinate system . . . 24
3.6 Jump operator . . . 26
3.7 Average operator . . . 26
3.8 L^2 scalar product . . . 26
3.9 Problem statement . . . 27
3.9.1 Stokes strong, weak and discrete form . . . 27
3.9.2 Properties of Stiffness matrix . . . 28
3.9.3 Upwinding . . . 30
3.9.4 Navier Stokes strong, weak and discrete form . . . 31
3.9.5 Newton method . . . 32
3.9.6 Properties of the Stiffness matrix . . . 33
3.9.7 Boundary condition . . . 33
3.10 Selection of solver . . . 33
3.10.1 Biconjugate gradients stabilized method . . . 33
3.10.2 Minimum residual method . . . 33
3.10.3 Schur complement method . . . 34

4 Implementation aspects **35**
4.1 Terminology . . . 35
4.2 Basis Function in RBmatlab . . . 37
4.3 Assembly of average operator . . . 38

4.4 Assembly of jump operator . . . 38
4.5 Matrix assemblies . . . 38
4.6 Setting boundary conditions . . . 42
4.7 Setting source term . . . 42
4.8 Program flow . . . 43
4.8.1 Grid Preparation . . . 43
4.8.2 Function space formulation . . . 43
4.8.3 Matrix assembly . . . 43
4.8.4 Solving assembled form . . . 44
4.8.5 Post processing . . . 44
4.8.6 Newton method . . . 44
4.8.7 Additional remarks . . . 44
4.9 Sparsity pattern . . . 44

5 Numerical experiments **53**
5.1 Error definitions . . . 53
5.2 Stokes flow . . . 54
5.2.1 Properties of Stiffness matrix . . . 54
5.2.2 Analytical example . . . 54
5.2.3 Lid-driven cavity problem . . . 54
5.2.4 Flow over cylinder . . . 55
5.3 Penalty parameter . . . 66
5.4 Navier Stokes flow . . . 66
5.4.1 Analytical example . . . 66
5.4.2 Lid-driven cavity problem . . . 68
5.4.3 Flow over cylinder . . . 68
5.5 Solver selection . . . 68

6 Summary, conclusion and outlook **79**
6.1 Conclusions . . . 79
6.2 Outlook . . . 80

7 Mathematical preliminaries **81**
7.1 Cholesky decomposition . . . 81
7.2 Saddle point formulation . . . 82
7.3 Sobolev spaces . . . 83
7.4 Basic definitions . . . 84
7.5 Linear forms . . . 84
7.6 Bilinear forms . . . 85
7.6.1 Coercivity of bilinear form . . . 85
7.6.2 Continuity of bilinear form . . . 85
7.7 Important inequalities . . . 85
7.7.1 Trace theorem . . . 85
7.7.2 Cauchy-Schwarz inequality . . . 86
7.7.3 Young's inequality . . . 86

8 References **87**
8.1 Online resources . . . 87
8.2 Code access . . . 87

Chapter 1

Introduction

1.1 Overview

The thesis deals with the numerical simulation of the Navier Stokes equation [White F.M. [7]] which is the core of Computational Fluid Dynamics or CFD. The thesis performs numerical simulation of the Navier Stokes equation through the discontinuous Galerkin method.
The goal of the thesis is to derive, discretise and implement the discontinuous Galerkin [Riviere B. [5]] weak form of the Navier Stokes equation and perform numerical simulation of the Navier Stokes equation. In the course of the numerical solution we measure the simulation efforts.
In chapter 2, the Navier Stokes equation is introduced. We first derive the Navier Stokes equation from the conservation equation or the Reynolds transport theoreom. We discuss the flow classification and important issues related to numerical simulation of the Navier Stokes equation.
In Chapter 3, 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 function or the Ansatz function, function spaces for pressure and velocity are introdued. 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 weak form.
In Chapter 4, we discuss the implementation of the discrete form of the Navier Stokes equation in RBmatlab, an opensource software developed at the University of Stuttgart and the University of Münster, for numerical simulation. During the chapter we discuss basis function evaluation in RBmatlab, matrix assembly, dimension of assembled matrices and boundary condition implementation. We further discuss Schur complement method and Newton method for nonlinearity.
In Chapter 5, we present results from numerical experiments, analyse results from benchmark problems, evaluate solver performance and draw related conclusions.
In chapter 6, we present mathematical preliminaries. Familiarity with concepts presented under this chapter is very helpful for understanding of the thesis. Readers familiar with the topics can skip this section.

Chapter 2

Engineering perspective and mathematical formulation

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

2.1 Derivation of the Navier Stokes equation

Before deriving the Navier Stokes equation 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 flow 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. The control surface is the surface enclosing the control volume. All equations can be derived from the Reynold's transport equation as presented by White F.M. [7]:

$$\frac{dB'}{dt'}|_{cs} = \frac{d}{dt'}\int_{cv} b'\rho dV + \int_{cs}(b'\rho)u \cdot dA \tag{2.1}$$

cv = Control volume
cs = Control surface
B' = Extensive property under consideration

b' = Intensive property corresponding to B'
ρ = Density of fluid
u = Velocity of fluid

If in the above equation B' is substituted as momentum M, 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.

$$F = \frac{dM}{dt'} = \frac{d}{dt'}\int_{cv} u\rho dV + \int_{cs} (u\rho)u \cdot dA \tag{2.2}$$

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

$$F = \int_{cs} \sigma \cdot dA + \int_{cv} \rho f dV \tag{2.3}$$

σ = Viscous stress
f = External force per unit volume
Equating external forces with change in momentum i.e. equating (2.2) and (2.3), 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,

$$-2\nabla \cdot (\nu \nabla^s u) + (1/\rho)\nabla p + (u \cdot \nabla)u = f \quad \text{in} \quad \Omega \quad . \tag{2.4}$$

The incompressible mass conservation equation can be written as

$$\nabla \cdot u = 0 \quad \text{in} \quad \Omega \quad . \tag{2.5}$$

The boundary conditions are expressed as,
Dirichlet boundary:

$$u = u_D \quad \text{on} \quad \Gamma_D \quad . \tag{2.6}$$

Neumann boundary:

$$-pn + 2\nu(n \cdot \nabla^s)u = t \quad \text{on} \quad \Gamma_N \quad . \tag{2.7}$$

u = flow velocity and $u : \Omega \to \mathbb{R}^d$
p = pressure and $p : \Omega \to \mathbb{R}$
ν = kinematic viscocity (fluid property) $\nu : \Omega \to \mathbb{R}$
ρ = density (fluid property) $\rho : \Omega \to \mathbb{R}$
f = external force $f : \Omega \to \mathbb{R}^d$
u_D = specified flow velocity at Dirichlet boundary $u_D : \Gamma_D \to \mathbb{R}^d$
n = normal unit vector $n : \partial\Omega \to \mathbb{R}^d$
t = specified Neumann flux $t : \Gamma_N \to \mathbb{R}^d$
$\nabla^s = \frac{1}{2}(\nabla + \nabla^T)$
The equation (2.4) is known as strong form of 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,

$$Re = uL/\nu \tag{2.8}$$

Where, L is the Characteristic geometrical dimension, for example 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.

2.1.1 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 (2.4) is no longer present. This equation is known as Stokes equation [White F.M.[7]]
The strong form of Stokes equation is as follow,

$$-\nu\Delta u + \nabla p = f \quad \text{in} \quad \Omega \tag{2.9}$$

$$u = u_D \quad \text{on} \quad \Gamma_D \tag{2.10}$$

$$-pn + \nu n \cdot \nabla u = t \quad \text{on} \quad \Gamma_N \tag{2.11}$$

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 (2.4) non linear. The fluctuations of velocity and pressure are of the order of the Kolmogrov scale.[Kundu et al. [10]]
The method used for solving Equation (2.4) 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 alternate models are used replacing the original model. However, the alternate model can not explain the flow physics completely. The prediction of accurate flow physics description requires economical numerical solution of (2.4). It is to be noted that simulation of turbulent flow is always a compromise between required flow physics and computational efforts.

Chapter 3

Discretisation and function spaces

3.1 Grid geometry

In numerical analysis a continuous problem is posed over finite number of degrees of freedom. This domain is constructed by dividing the original domain. This 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 betwen 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 3.1)

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, $\tau_k \in \mathcal{T}$.

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 of ver-

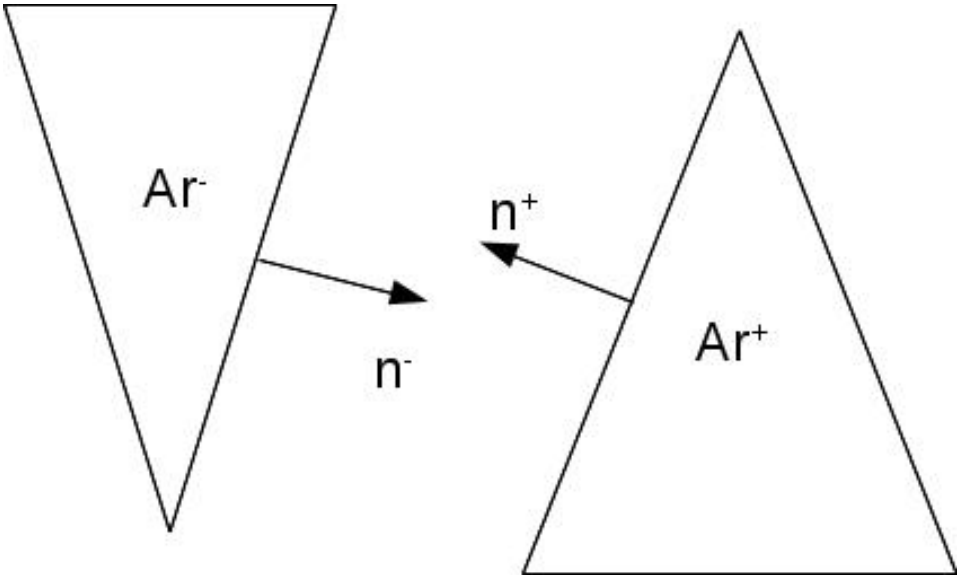


Figure 3.1: Element self (+) and neighbouring element (-)

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

$$r = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3 \quad . \tag{3.1}$$

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

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \quad . \tag{3.2}$$

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 (3.2) we have $\lambda_3 = 1/3$.

3.2 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 Navier Stokes Discontinuous Galerkin 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 domain or precisely, we refer to Figure 3.2.

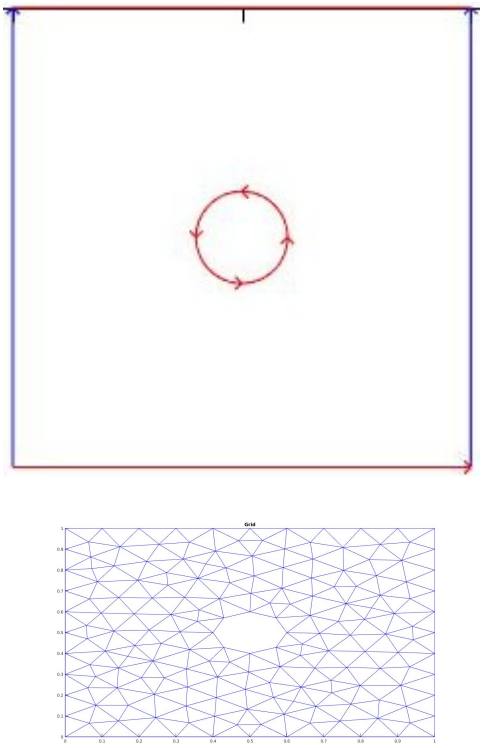


Figure 3.2: Continuous domain (left) and discretised domain or grid (right)

3.3 Discontinuous Galerkin 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. The dimension of truth space is denoted as N.

$$\mathbb{V} = \{\phi \in (L^2(\mathcal{T}))^{d_u} | \quad \phi \in (P^D(\tau_k))^{d_u} \quad \forall \quad \tau_k \in \mathcal{T}\} \tag{3.3}$$

$$\mathbb{Q} = \{\psi \in (L^2(\mathcal{T}))^{d_p} | \quad \psi \in (P^{D-1}(\tau_k))^{d_p} \quad \forall \quad \tau_k \in \mathcal{T}\} \tag{3.4}$$

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. 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. [11] for local discontinuous Galerkin and by Montlaur et al. [2] for the Compact discontinuous Galerkin and the Interior penalty method.
Discontinuous Galerkin methods for the Navier Stokes equation were compared by Motlaur 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, implemenetation of IPM is simpler as compared to implementation of CDG. Both methods, CDG and IPM, have almost similar convergence rates.

3.4 Nodal basis function and Orthonormal basis function

The "Basis functions" are also known as "Ansatz functions". There are two kinds of basis function which are used in the application of continuous Galerkin method or discontinuous Galerkin method.
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.
Similarly for higher order polynomials the nodes are located as shown in Figure 3.3 .

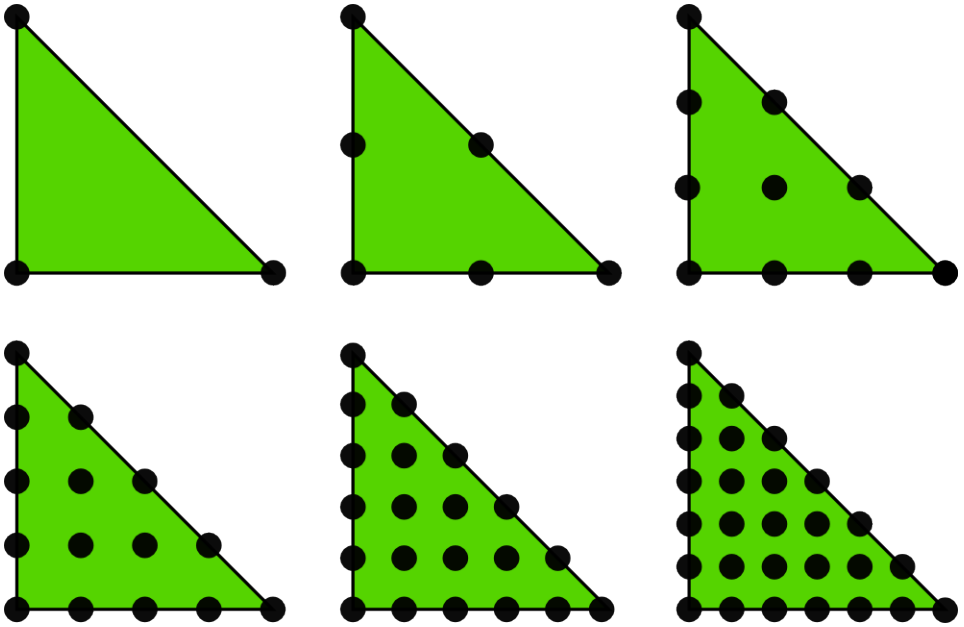


Figure 3.3: Finite Element nodes on triangle for polynomials of different degrees

Source : http://hplgit.github.io/INF5620/doc/pub/sphinx-fem/._main_fem009.html

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

$$u_{npe} = d_u \frac{(D+1)(D+2)}{2} \quad \text{for velocity} \tag{3.5}$$

$$p_{npe} = d_p \frac{(D)(D+1)}{2} \quad \text{for pressure} \tag{3.6}$$

3.4.1 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. For example, Nodal basis 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),

$$\begin{aligned} \hat{\phi}_i &= \frac{x - x_{i-1}}{x_i - x_{i-1}} \quad \text{for} \quad x_{i-1} \leq x \leq x_i \\ \hat{\phi}_i &= \frac{x_{i+1} - x}{x_{i+1} - x_i} \quad \text{for} \quad x_i \leq x \leq x_{i+1} \\ \hat{\phi}_i &= 0 \text{ else} \end{aligned} \tag{3.7}$$

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

3.4.2 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.

$$\begin{aligned} (\hat{\phi}_i, \hat{\phi}_j) &= \int_{\hat{T}} \hat{\phi}_i \hat{\phi}_j = 1 \quad \text{if} \quad i = j \\ (\hat{\phi}_i, \hat{\phi}_j) &= \int_{\hat{T}} \hat{\phi}_i \hat{\phi}_j = 0 \quad \text{if} \quad i \neq j \end{aligned} \tag{3.8}$$

In the present analysis, $(\cdot, \cdot)$ represents L^2 scalar product and $\hat{T}$ is reference triangle (Section 3.5).

3.5 Global and local co-ordinate 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

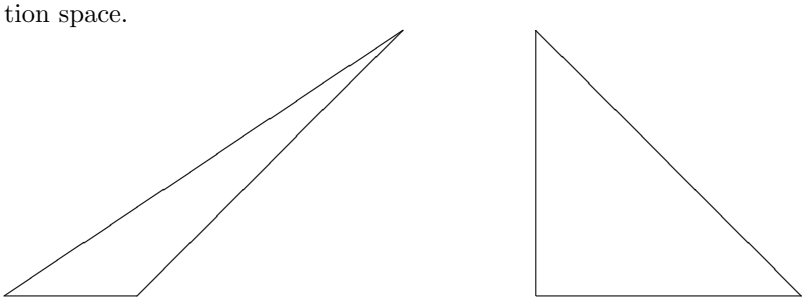
by vertex indices in order forming triangle. The transformation from local coordinate $\hat{X}$ to global coordinate X is defined by the mapping,

$$F_k : \hat{X} \mapsto X \quad \forall \quad \hat{X} \in \hat{T} \quad \text{and} \quad X \in \mathcal{T} \tag{3.9}$$

This mapping function is defined as,

$$F_k(\hat{X}) : X = J_k\hat{X} + C \tag{3.10}$$

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 $\hat{}$. This function space is known as local basis function space.



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

$$\int_\Omega g(x)dx = \sum_{k=1}^{nel}\int_{\tau_k} g(x)dx = \sum_{k=1}^{nel}\int_{\hat{T}} g(\hat{x})|\det(J_k)|d\hat{x} \tag{3.11}$$

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

$$\int_\Gamma g(x)ds = \int_{\hat{\Gamma}} g(\hat{x})ld\hat{s} \tag{3.12}$$

Also, the following holds,

$$\nabla g = JIT_k \quad \hat{\nabla}\hat{g} \tag{3.13}$$

where,
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 in 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,

$$g(x) = \hat{g}(\hat{x}) \quad \text{for} \quad x = F_k(\hat{x}) \quad \text{according to equation (3.10)} \tag{3.14}$$

3.6 Jump operator

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

$$[u] = u^+ \cdot n^+ + u^- \cdot n^- \tag{3.15}$$

where n is the unit normal to an edge of an element pointing outward from the element.
As pointed out by [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 [1] modified jump representation as,
1.

$$\begin{gathered}[pn] = p^+ n^+ + p^- n^- \text{ on } \Gamma \\ [pn] = pn \text{ on } \Gamma_D \\ \text{where } p \text{ is scalar.}\end{gathered}$$

2.

$$\begin{gathered}[n \otimes u] = n^+ \otimes u^+ + n^- \otimes u^- \text{ on } \Gamma \\ [n \otimes u] = n \otimes u \text{ on } \Gamma_D \\ \text{or} \\ [n \cdot u] = n^+ \cdot u^+ + n^- \cdot u^- \text{ on } \Gamma \\ [n \cdot u] = n \cdot u \text{ on } \Gamma_D \\ \text{where } u \text{ is vector and } n \otimes u = u_i n_j\end{gathered}$$

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.

3.7 Average operator

The average operator is defined as,

$$\{u\} = \frac{u^+ + u^-}{2} \tag{3.16}$$

As can be seen definition of average operator does not involve normal and hence, has simpler definition.
Also, u and $\{u\}$ are in same function space i.e. average of a vector is vector and average of scalar is scalar.

3.8 L^2 scalar product

We denote the scalar product of p and q by (p, q) which refers to,
If p and q are scalars,

$$(p, q) = \int_\Omega pq d\Omega \tag{3.17}$$

If p and q are vectors,

$$(p, q) = \int_\Omega p \cdot q d\Omega \tag{3.18}$$

If p and q are tensors,

$$(p,q) = \int_{\Omega} p : q d\Omega \quad \text{where} \quad p : q = Tr(pq^T) \tag{3.19}$$

3.9 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 [2] and [1] we arrive at weak form of Stokes and Navier Stokes interior penalty approximation.

3.9.1 Stokes strong, weak and discrete form

The strong form of the Stokes equation is as follow,

$$-\nu\Delta u + \nabla p = f \quad \text{in} \quad \Omega \tag{3.20}$$

$$u = u_D \quad \text{on} \quad \Gamma_D \tag{3.21}$$

$$-pn + \nu n \cdot \nabla u = t \quad \text{on} \quad \Gamma_N \tag{3.22}$$

The weak form of Stokes equation is as follow,

$$a_{IP}(u,\phi) + b(\phi,p) + (\{p\},[n\cdot\phi])_{\Gamma\cup\Gamma_D} = l_{IP}(\phi) \quad . \tag{3.23}$$

$$\begin{aligned} a_{IP}(u,\phi) &= (\nabla u, \nabla\phi) + C_{11}([n\otimes u],[n\otimes\phi])_{\Gamma\cup\Gamma_D} \\ &-\nu(\{\nabla u\},[n\otimes\phi])_{\Gamma\cup\Gamma_D} - \nu([n\otimes u],\{\nabla\phi\})_{\Gamma\cup\Gamma_D} \quad . \end{aligned} \tag{3.24}$$

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

$$b(\phi,\psi) = -\int_{\mathcal{T}} \psi\nabla\cdot\phi \tag{3.25}$$

$$l_{IP}(\phi) = (f,\phi) + (t,\phi)_{\Gamma_N} + C_{11}(u_D,\phi)_{\Gamma_D} - (n\otimes u_D, \nu\nabla\phi)_{\Gamma_D} \tag{3.26}$$

The discrete form of Stokes equation is written as,

$$AU + BP = F_1 \quad . \tag{3.27}$$

Matrix A and matrix B are calculated as per equation 3.32 and equation 3.34 respectively.
The strong form of continuity equation is as follow,

$$\nabla\cdot u = 0 \quad \text{in} \quad \Omega \quad . \tag{3.28}$$

and the weak for of continuity equation is as follow,

$$b(u,\psi) + (\{\psi\}, [n \cdot u])_{\Gamma \cup \Gamma_D} = (q, n \cdot u_D)_{\Gamma_D} \quad . \tag{3.29}$$

The discrete form of continuity equation is written as,

$$B^T U = F_2 \quad . \tag{3.30}$$

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

$$\underbrace{\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix}}_{\text{Stiffness matrix}} \quad \underbrace{\begin{pmatrix} U \\ P \end{pmatrix}}_{\text{Solution vector}} \quad = \quad \underbrace{\begin{pmatrix} F_1 \\ F_2 \end{pmatrix}}_{\text{Right hand side (Known)}} \tag{3.31}$$

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

3.9.2 Properties of Stiffness matrix

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

$$\begin{aligned} A_{ij} = \sum_{k=1}^{d} (\frac{\partial \phi_i}{\partial x_k}, \frac{\partial \phi_j}{\partial x_k}) + C_{11} \sum_{k=1}^{d} ([\phi_i n_k], [\phi_j n_k])_{\Gamma \cup \Gamma_D} \\ -\nu \sum_{k=1}^{d} ([\phi_i n_k], \{\frac{\partial \phi_j}{\partial x_k}\})_{\Gamma \cup \Gamma_D} - \nu \sum_{k=1}^{d} (\{\frac{\partial \phi_i}{\partial x_k}\}, [\phi_j n_k])_{\Gamma \cup \Gamma_D} \end{aligned} \tag{3.32}$$

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 7.6.1),
$\exists \quad c' > 0$ and for non zero vector z

$$z^T A(\phi, \phi) z \geq c' ||z||^2 \implies z^T A(\phi, \phi) z > 0 \tag{3.33}$$

3. Size of matrix A: $A \in \mathbb{R}^{u_{ndofs} \times u_{ndofs}}$ (u_{ndofs} is total number of degrees of freedom of velocity)
Each element of B can be represented as,

$$B_{ij} = -\int_{\mathcal{T}} \frac{\partial \phi_i}{\partial x_i} \psi_j + (\{\psi_j\}, [n \cdot \phi_i])_{\Gamma \cup \Gamma_D} \tag{3.34}$$

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} on triangular grid and taylor-hood pressure velocity basis function can be calculated as below. In present analysis we have $d_u = 2$ and $d_p = 1$.

$$u_{ndofs} = 2 \quad \left(\frac{(D+1)(D+2)}{2} \right) \quad nel \tag{3.35}$$

$$p_{ndofs} = \left(\frac{D(D+1)}{2} \right) \quad nel \tag{3.36}$$

With above considerations we arrive at following conclusions,
1. Stiffness matrix is symmertic.
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 (7.9) we see that the congruent matrix of stiffness matrix for Stokes equation is,

$$\begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \tag{3.37}$$

We look at the the eigen values of S as the number of positive and negative eigen values 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 eigen values of S are negative.

Coercivity constant for equation for Stokes flow

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

$$\nu_0 \leq \nu \leq \nu_1 \tag{3.38}$$

Using Cauchy Schwarz inequality,

$$\begin{aligned} \sum_{e\in\Gamma\cup\Gamma_D} \int_e \nu[n\otimes\phi] &\leq \sum_{e\in\Gamma\cup\Gamma_D} ||\nu\nabla\phi\cdot n||_{L^2(e)} \quad |||[\phi]|||_{L^2(e)} \\ &\leq \sum_{e\in\Gamma\cup\Gamma_D} ||\nu\nabla\phi\cdot n||_{L^2(e)} (\frac{1}{l})^{(1/2-1/2)} \quad |||[\phi]|||_{L^2(e)} \quad . \end{aligned} \tag{3.39}$$

Based on the Trace inequatlity and the lower and upper bound for viscocity, for neighbouring elements τ_{k_1} and τ_{k_2} sharing the edge e,

$$\begin{aligned} ||\{\nu\nabla\phi\cdot n\}||_{L^2(e)} \leq \frac{C_t\nu_1}{2} h_{\tau_{k_1}}^{-1/2} ||\nabla\phi||_{L^2(\tau_{k_1})} \\ + \frac{C_t\nu_1}{2} h_{\tau_{k_2}}^{-1/2} ||\nabla\phi||_{L^2(\tau_{k_2})} \quad . \end{aligned} \tag{3.40}$$

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

$$\begin{aligned} \int_e \{\nu\nabla\phi\}[n\otimes\phi] \leq \frac{C_t\nu_1}{2} \left(h_{(\tau_{k_1})}^{\frac{-1}{2}} ||\nabla\phi||_{L^2(\tau_{k_1})} + h_{(\tau_{k_2})}^{\frac{-1}{2}} ||\nabla\phi||_{L^2(\tau_{k_2})} \right) \\ l^{\frac{\beta_0}{2}} \left(\frac{1}{l}\right)^{\frac{\beta_0}{2}} |||[\phi]|||_{L^2(e)} \quad . \end{aligned} \tag{3.41}$$

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),

$$\begin{aligned}\int_e \{\nu\nabla\phi\}[n\otimes\phi] \leq C_t\nu_1 \left(\sum_{e\in\Gamma_h\cup\Gamma_D}\frac{1}{l_0^\beta}|||[\phi]|||^2_{L^2(e)}\right)^{1/2} \\ \times\left(\sum_{e\epsilon\Gamma_h}||\nabla\phi||^2_{L^2(\tau_{k_1})}+||\nabla\phi||^2_{L^2(\tau_{k_2})}+\sum_{e\epsilon\Gamma_D}||\nabla\phi||^2_{0,\tau_{k_1}}\right)\end{aligned} \tag{3.42}$$

$$\begin{aligned}\int_e \{\nu\nabla\phi\}[n\otimes\phi] \leq C_t\nu_1\sqrt{n_0}\left(\sum_{e\in\Gamma_h\cup\Gamma_D}\frac{1}{l_0^\beta}|||[\phi]|||^2_{L^2(e)}\right)^{1/2} \\ \left(\sum_{e\in\Gamma_h\cup\Gamma_D}||\nabla\phi||^2_{L^2(\tau)}\right) \quad .\end{aligned} \tag{3.43}$$

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

$$\sum_{e\in\Gamma\cup\Gamma_D}\int_e\{\nu\nabla\phi\}[n\otimes\phi] \leq \frac{\delta}{2}\sum_{\tau\in\mathcal{T}}||\nu^{1/2}\nabla\phi||^2_{L^2(\tau)}+\frac{C_t^2\nu_1^2n_0}{2\delta\nu_0}\sum_{e\in\Gamma\cup\Gamma_D}\frac{1}{l^{\beta_0}}|||[\phi]|||^2_{L^2(e)} \tag{3.44}$$

$$a_\epsilon(\phi,\phi) \geq \left(1-\frac{\delta}{2}|1-\epsilon|\right)\sum_{\tau\epsilon\mathcal{T}}||\nu^{1/2}\nabla v||^2_{L^2(\tau)}+\sum_{e\epsilon\Gamma_h\cup\Gamma_D}\frac{\sigma_e^0-\frac{C_t^2\nu_1^2n_0}{2\delta K_0}|1-\epsilon|}{l^{\beta_0}}|||[v]|||^2_{L^2(e)} \tag{3.45}$$

where, we have Penalty parameter,

$$C_{11} = \frac{\kappa_e}{l} \tag{3.46}$$

where, κ_e is coercivity constant.
We can obtain exact expression for κ_e for triangular mesh and Symmetric Interior Penalty Galerkin formulation $\epsilon = -1$ as,

$$\begin{aligned}\kappa_e = \frac{3({\nu_1}^{\tau_{k_1}})^2}{2{\nu_0}^{\tau_{k_1}}}D(D+1)l^{\beta_0-1}cot\theta^{\tau_{k_1}} \\ +\frac{3({\nu_1}^{\tau_{k_2}})^2}{2{\nu_0}^{\tau_{k_2}}}D(D+1)l^{\beta_0-1}cot\theta^{\tau_{k_2}}\end{aligned} \tag{3.47}$$

3.9.3 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} [5],

$$\begin{aligned}g^{up} = g|_{\tau_1} \quad \text{if} \quad g\cdot n_\tau \geq 0 \\ g^{up} = g|_{\tau_2} \quad \text{if} \quad g\cdot n_\tau < 0\end{aligned} \tag{3.48}$$

In our analysis this is explicitly defined in the term $c(u, u, \phi)$ in section 3.9.4.
The scheme is such that,

$$\begin{aligned} u^{up} &= u \quad \text{if} \quad u \cdot n \geq 0 \\ u^{up} &= u^{ext} \quad \text{if} \quad u \cdot n < 0 \end{aligned} \tag{3.49}$$

3.9.4 Navier Stokes strong, weak and discrete form

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 foces in Stokes flow.
The strong form of Navier Stokes equation can be written as,

$$-\nu \Delta u + \nabla p + (u \cdot \nabla) u = f \quad \text{in} \quad \Omega \tag{3.50}$$

with Dirichlet and Neumann boundary condition as per section 3.9.1. Also the continuity equation as mentioned in section 3.9.1 is valid.
The inertial forces term in weak form with upwinding (section 3.9.3) is as below,

$$\begin{aligned} c(g; u, \phi) = \sum_{i=1}^{nel} \int_{\partial \Omega_i \setminus \Gamma_N} \frac{1}{2} [[(g \cdot n_i)(u^{ext} + u) - |g \cdot n_i|(u^{ext} - u)]] \cdot \phi \\ + \int_{\Gamma_N} (g \cdot n) u \cdot \phi - ((g \cdot \nabla)\phi, u) \end{aligned} \tag{3.51}$$

$$u^{ext} = \lim_{\epsilon \to 0} u(x + \epsilon n_i) \quad \text{for} \quad x \in \partial \mathcal{T}_i \tag{3.52}$$

Hence, the Navier Stokes equation can be written as,

$$a_{IP}(u, \phi) + c(u; u, \phi) + b(\phi, p) + (\{p\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D} = l_{IP}(\phi) \tag{3.53}$$

Here, we can see that the $c(u, u, \phi)$ is non linear term.
In discrete form this equation can be written as,

$$AU + C(U)U + BP = F \tag{3.54}$$

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 continuity equation can be written as,

$$\underbrace{\begin{pmatrix} A + C(U) & B \\ B^T & 0 \end{pmatrix}}_{\text{Stiffness matrix}} \underbrace{\begin{pmatrix} U \\ P \end{pmatrix}}_{\text{Solution vector}} = \underbrace{\begin{pmatrix} F_1 \\ F_2 \end{pmatrix}}_{\text{Right hand side (Known)}} \tag{3.55}$$

In the present analysis, we use Newton method to solve nonlinear system of equations. We present the Newton scheme in section 3.9.5.
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 3.23 for Stokes flow and 3.53 for Navier Stokes flow is satisfied.

3.9.5 Newton method

We derive newton method as per method presented by Haasdonk B. [4] for solving nonlinear system of equation arising out of discrete form of Navier Stokes equation. For $u, \phi, h \in \mathbb{V}$ and $p, \psi, h' \in \mathbb{Q}$

$$S(u) = a(u,\phi) + b(\phi,p) + (\{p\},[n\cdot\phi])_{\Gamma\cup\Gamma_D} - l_{IP}(\phi) \tag{3.56}$$

$$\begin{aligned} S(u+h) - S(u) = (a(u+\delta h,\phi) + c(u+\delta h; u+\delta h,\phi) \\ +b(\phi,p+\delta h') + (\{p+\delta h'\},[n\cdot\phi])_{\Gamma\cup\Gamma_D} - l_{IP}(\phi)) - (a(u,\phi) \\ +c(u,u,\phi) + b(\phi,p) + (\{p\},[n\cdot\phi])_{\Gamma\cup\Gamma_D} - l_{IP}(\phi)) \end{aligned} \tag{3.57}$$

$$\begin{aligned} S(u+h) - S(u) = 2\delta c(u,h,\cdot) + \delta^2 c(h,h,\cdot) + \delta a(h,\cdot) \\ +\delta b(h',\cdot) + \delta(\{h'\},[n\cdot\phi])_{\Gamma\cup\Gamma_D} \end{aligned} \tag{3.58}$$

$$DS(u) = \lim_{\delta\to 0} \frac{S(u+h) - S(u)}{\delta} \tag{3.59}$$

$$DS(u) = 2c(u,h,\cdot) + a(h,\cdot) + b(h',\cdot) + (\{h'\},[n\cdot\phi])_{\Gamma\cup\Gamma_D} \tag{3.60}$$

Following similar procedure we write for continuity equation:

$$S'(u) = b(u,\psi) + (\{\psi\},[n\cdot u])_{\Gamma\cup\Gamma_D} - (\psi, n\cdot u_D)_{\Gamma_D} \tag{3.61}$$

$$S'(u+\delta h) = b(u+\delta h,\psi) + (\{\psi\},[n\cdot u+\delta h])_{\Gamma\cup\Gamma_D} - (\psi, n\cdot u_D)_{\Gamma_D} \tag{3.62}$$

$$DS'(u) = b(\delta h,\psi) + (\{\psi\},[n\cdot\delta h])_{\Gamma\cup\Gamma_D} \tag{3.63}$$

Algotrithm for the Newton method is as follow:
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 Newton method means, solving the equation

$$\underbrace{\begin{pmatrix} A + C(U^{iter}) & B \\ B^T & 0 \end{pmatrix}}_{\text{Stiffness matrix}^{iter}} \underbrace{\begin{pmatrix} U^{iter+1} \\ P^{iter+1} \end{pmatrix}}_{\text{Solution vector}^{iter+1}} = \underbrace{\begin{pmatrix} F_1 \\ F_2 \end{pmatrix}}_{\text{Right hand side (Known function)}} \tag{3.64}$$

to reach convergence i.e. $||U^{iter+1} - U^{iter}|| < tol$
For success of Newton method it is important to have good initial guess. For Navier Stokes method we use solution of Stokes equation as initial guess.

3.9.6 Properties of the Stiffness matrix

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

$$C_{IJ} = \sum_{i=1}^{nel} \int_{\partial \mathcal{T}_i \setminus \Gamma_N} \frac{1}{2}[[(U^{iter} \cdot n_i)({\phi^{ext}}_J + \phi_J) - |U^{iter} \cdot n_i|({\phi^{ext}}_J - \phi_J)]] \cdot \phi_I + \int_{\Gamma_N} (U^{iter} \cdot n)\phi_J \cdot \phi_I - ((U^{iter} \cdot \nabla)\phi_J, \phi_I) \tag{3.65}$$

Size of matrix : $C \in \mathbb{R}^{u_{ndofs} \times u_{ndofs}}$.
We now can see that whereas Stiffness matrix of Stokes equation is symmetric and has Symmetric positive definite part A, the Stiffness matrix of Navier Stokes equation is non symmetric.

3.9.7 Boundary condition

We impose in our analysis boundary condition weakly. This is done by the linear terms on 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 3.53 and equation 3.23.

3.10 Selection of solver

In order to solve variation form of 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 3.10.3) is implemented separately based on Cholesky decomposition (Section 7.1).

3.10.1 Biconjugate gradients stabilized method

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

$$KX = B \tag{3.66}$$

with K = Coefficient matrix, X = Vector of unknowns and B = vector of value of known function. The coefficient matrix A need not be symmetric.
In the present analysis it was found that the *bicgstab* stops before converging to specified tolerance level or reaching maximum number of iterations.

3.10.2 Minimum residual method

The *minres* method is special kind of conjugate gradient method but does not require LU decomposition. It also solves the equations 3.66. However, the coefficient matrix K must be symmetric.

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

3.10.3 Schur complement method

We subdivide the matrix form of Stokes equation (7.2) into smaller dimension system by Schur complement. We also note that matrix A is symmetric positive definite and matrix K is symmetric. Here, $A \in \mathbb{R}^{u_{ndofs} \times p_{ndofs}}$; $B \in \mathbb{R}^{u_{ndofs} \times p_{ndofs}}$; $U, F_1 \in \mathbb{R}^{u_{ndofs}}$; $P, F_2 \in \mathbb{R}^{p_{ndofs}}$.
We solve equation (7.2) in following steps,
STEP 1:

$$U = A^{-1}(F_1 - BP) \tag{3.67}$$

The matrix A is inverted by Cholesky decomposition (section 7.1).
STEP 2 :
We substitute now equation (3.67) into equation (3.30) and (3.27)

$$-B^T A^{-1} BP = F_2 - B^T A^{-1} F_1 \tag{3.68}$$

STEP 3 :
We now back substitute P in equation (3.67) and compute U in order to eventually obtain the solution vector.
As pointed by Fritzen F. [6], 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 Schur complement is much faster and in fact, for low flow velocities accurate method. Also Cholesky decoposition provides error message in case A is not symmetric positive definite, indicating improper choice of penalty parameter.

Chapter 4

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 of Haasdonk B.[3] 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.

4.1 Terminology

A. `params` and `paramsP` : These are structures corresponding to velocity and pressure respectively and primarily containing following fields. (Table 4.1 and Table 4.2)
B. `grid` : It is the structure containing fields relevant to information stored in grid. Below are the fields of grid. (Table 4.3)
C. We also some other variables. (Table 4.4)

Table 4.1: Fields of `params`

Fieldname	**Representation**	**Size**	**Description**
`dimrange`	d_u	$\mathbb{N}$	User defined field
`pdeg`	D	$\mathbb{N}$	User defined field
`ndofs_per_element`	u_{npe}	$\mathbb{N}$	Calculated as per equation 3.5
`ndofs`	u_{ndofs}	$\mathbb{N}$	Calculated as per equation 3.35
`dofs`	U	$\mathbb{R}^{u_{ndofs}}$	Calculated after solving 3.53 and 3.23
`show_sparsity`	-	Bool variable	User defined switch to plot sparsity pattern

Table 4.2: Fields of `paramsP`

Fieldname	**Representation**	**Size**	**Description**
`dimrange`	d_p	$\mathbb{N}$	User defined field
`pdeg`	$D-1$	$\mathbb{N}$	Calculated as taylor hood element
`ndofs_per_element`	p_{npe}	$\mathbb{N}$	Calculated as per equation 3.6
`ndofs`	p_{ndofs}	$\mathbb{N}$	Calculated as per equation 3.36
`dofs`	P	$\mathbb{R}^{p_{ndofs}}$	Calculated after solving 3.53 and 3.23
`show_sparsity`	-	Bool variable	User defined switch to plot sparsity pattern

Table 4.3: 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 4.4: 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

4.2 Basis Function in RBmatlab

The Basis functions in RBmatlab can be point evaluated by a routine called `ldg_evaluate_basis`.
We approximatete the solution at any point from solution of degrees of freedom in orthonormal basis. In matrix formulation this means basis function is matrix of the size,

$$\phi \in \mathbb{R}^{u_{npe} \times d_u} \tag{4.1}$$

$$\psi \in \mathbb{R}^{p_{npe} \times d_p} \tag{4.2}$$

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.
The derivative of the basis functions is provided by a routine `ldg_evaluate_basis_derivative`. The derivative of basis function $(\phi)_i$, where

$1 \leq i \leq u_{npe}$ is a cell containing matrix $\nabla(\phi)_i$ of size,

$$\nabla(\phi)_i \in \mathbb{R}^{d_u \times d} \tag{4.3}$$

where 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,

$$\nabla(\psi)_i \in \mathbb{R}^{d_p \times d} \tag{4.4}$$

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

4.3 Assembly of average operator

Average of quantity A_h in discrete form is assembled as,

$$\{A_h\} = (A_h^+ + A_h^-)/2 \tag{4.5}$$

The assembly of the average operator is relatively simple as compared to the jump operator which is explained in Section 4.5.

4.4 Assembly of jump operator

The jump of the quantity A_h in discrete form is assembled as,

$$[A_h] = A_h^+ n^+ + A_h^- n^- \tag{4.6}$$

In case of terms such as $[A_h], [B_h]$ we assemble matrices as,
For internal edges Γ,

$$[A_h], [B_h] = A_h^+ n^+ B_h^+ n^+ + A_h^+ n^+ B_h^- n^- + A_h^- n^- B_h^+ n^+ + A_h^- n^- B_h^- n^- \tag{4.7}$$

and for Dirichlet edges Γ_D

$$[A_h], [B_h] = A_h^+ n^+ B_h^+ n^+ \tag{4.8}$$

4.5 Matrix assemblies

It is to be noted that $\hat{\phi}$ is evaluated at local coordinate corresponding to global coordinate in accordance with equation 3.14.
The matrices from weak form of Navier Stokes equation have been assembled in 3 steps,
1. Evaluating function at vertex of local element and transform to global geometry.
2. Transform function evaluation from step 1 to global geometry (if not done in step 1) and performing integral of function over local element.
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 index of element degree of freedom in global degree of freedom vector.
We also perform numerical integration over domain Ω as,

$$\int_{\Omega} f(x) = \sum_{i=1}^{nop} f(x_i) w_i \tag{4.9}$$

Where,
x_i = Location of function evaluation
w_i = Weight at corressponding 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.

$$\det J(k) = 2Ar(k) \tag{4.10}$$

With this preliminary informations we discuss now the assembly of matrices.
1. $(\nabla\phi, \nabla\phi)$:
Step 1: Evaluation of $(\nabla\phi, \nabla\phi)$,
We first evaluate the derivative of $\hat{\phi}$ and $JIT(k,:,:)$ at point x_i through `ldg_evaluate_basis_deriva` and `grid` structure. 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 3.13.
We than assemble matrix $res_1[i,j] = \phi_i \phi_j^T$ for $1 \leq i, j \leq u_{npe}$
Step 2: Performing integration in global co-ordinate system,
We perform the numerical integration as per 4.9
$res_2 = \int_{\hat{T}} (res_1)(2Ar(k))$
Step 3: Looping over each element and performing following operation in each loop $res_3[ids\_velocity, ids\_velocity] = res_2$
2. $([n \otimes \phi], [n \otimes \phi])_{\Gamma \cup \Gamma_D}$:
Step 1: On local element following matrices are evaluated,
$res_1^{++} = (n \otimes \hat{\phi})^+ (n \otimes \hat{\phi})^+$
$res_1^{+-} = (n \otimes \hat{\phi})^+ (n \otimes \hat{\phi})^-$
$res_1^{-+} = (n \otimes \hat{\phi})^- (n \otimes \hat{\phi})^+$
$res_1^{--} = (n \otimes \hat{\phi})^- (n \otimes \hat{\phi})^-$
Please note that $\hat{\phi_h}$ is evaluated at local coordinate corressponding to global co-ordinate.
Step 2: In step 2 we perform following integration, We perform the numerical integration as per equation 3.12
$res_2^{++} = \int_{\Gamma} res_1^{++} EL(i,j)$
$res_2^{+-} = \int_{\Gamma} res_1^{+-} EL(i,j)$
$res_2^{-+} = \int_{\Gamma} res_1^{-+} EL(i,j)$
$res_2^{--} = \int_{\Gamma} res_1^{--} EL(i,j)$
Step 3: Loop over all elements, define the global assembly matrix as zero matrix and perform following operation in each loop,
$res_3^{++}[ids\_velocity\_self, ids\_velocity\_self] = res_2^{++}$
$res_3^{+-}[ids\_velocity\_self, ids\_velocity\_neighbour] = res_2^{+-}$
$res_3^{-+}[ids\_velocity\_neighbour, ids\_velocity\_self] = res_2^{-+}$
$res_3^{--}[ids\_velocity\_neighbour, ids\_velocity\_neighbour] = res_2^{--}$
Finally,
$res_3 = res_3^{++} + res_3^{+-} + res_3^{-+} + res_3^{--}$

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. $(\{\nabla\phi\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D}$:
Step 1: On local element following matrices are evaluated,
$res_1^{++} = (\nabla\phi)^+(n \otimes \hat{\phi})^+$
$res_1^{+-} = (\nabla\phi)^+(n \otimes \hat{\phi})^-$
$res_1^{-+} = (\nabla\phi)^-(n \otimes \hat{\phi})^+$
$res_1^{--} = (\nabla\phi)^-(n \otimes \hat{\phi})^-$
Step 2: In step 2 we perform following integration, We perform the numerical integration as per equation 3.12
$res_2^{++} = \int_\Gamma res_1^{++} EL(i,j)$
$res_2^{+-} = \int_\Gamma res_1^{+-} EL(i,j)$
$res_2^{-+} = \int_\Gamma res_1^{-+} EL(i,j)$
$res_2^{--} = \int_\Gamma res_1^{--} EL(i,j)$
Step 3: Loop over all elements, define the global assembly matrix as zero matrix and perform following operation in each loop,
$res_3^{++}[ids\_velocity\_self, ids\_velocity\_self] = res_2^{++}$
$res_3^{+-}[ids\_velocity\_self, ids\_velocity\_neighbour] = res_2^{+-}$
$res_3^{-+}[ids\_velocity\_neighbour, ids\_velocity\_self] = res_2^{-+}$
$res_3^{--}[ids\_velocity\_neighbour, ids\_velocity\_neighbour] = res_2^{--}$
Finally,
$res_3 = res_3^{++} + res_3^{+-} + res_3^{-+} + res_3^{--}$
It is to be noted that on Dirichlet boundary only $res_1^{++}, res_2^{++}, res_3^{++}$ is evaluated as all other terms are zero.
The same routine is also used in case of $([n \otimes \phi], \{\nabla u\})_{\Gamma \cup \Gamma_D}$
4. $(\{\psi\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}$:
Step 1: On local element following matrices are evaluated,
$res_1^{++} = (\psi)^+(n \cdot \hat{\phi})^+$
$res_1^{+-} = (\psi)^+(n \cdot \hat{\phi})^-$
$res_1^{-+} = (\psi)^-(n \cdot \hat{\phi})^+$
$res_1^{--} = (\psi)^-(n \cdot \hat{\phi})^-$
Please note that $\hat{\phi}$ is evaluated at local coordinate corresponding to global coordinate in accordance with 3.14.
Step 2: In step 2 we perform following integration, We perform the numerical integration as per equation 3.12
$res_2^{++} = \int_\Gamma res_1^{++} EL(i,j)$
$res_2^{+-} = \int_\Gamma res_1^{+-} EL(i,j)$
$res_2^{-+} = \int_\Gamma res_1^{-+} EL(i,j)$
$res_2^{--} = \int_\Gamma res_1^{--} EL(i,j)$
Step 3: Loop over all elements, define the global assembly matrix as zero matrix and perform following operation in each loop,
$res_3^{++}[ids\_velocity\_self, ids\_velocity\_self] = res_2^{++}$
$res_3^{+-}[ids\_velocity\_self, ids\_velocity\_neighbour] = res_2^{+-}$
$res_3^{-+}[ids\_velocity\_neighbour, ids\_velocity\_self] = res_2^{-+}$
$res_3^{--}[ids\_velocity\_neighbour, ids\_velocity\_neighbour] = res_2^{--}$
Finally,
$res_3 = res_3^{++} + res_3^{+-} + res_3^{-+} + res_3^{--}$
It is to be noted that on Dirichlet boundary only $res_1^{++}, res_2^{++}, res_3^{++}$ is evalu-

ated as all other terms are zero.
5. $-\int_{\hat{T}} \psi \nabla \cdot \phi$ We note that, In accordance with equation 3.13

$$\nabla\phi = JIT\hat{\nabla}\hat{\phi} \tag{4.11}$$

and in accordance with equation 3.14

$$\psi(x) = \hat{\psi}(\hat{x}) \tag{4.12}$$

Step 1 : We first evaluate JIT, $\hat{\nabla}\hat{\phi}$ and ψ and assemble following local matrix
$res_1 = \hat{\psi}_i \hat{\nabla} \cdot \hat{\phi}_j$
Step 2 : We integrate the function over domain
$res_2 = -\int_{\hat{T}} (res_1)(2Ar(k))$
Step 3: Assemble the global matrix
$res_3[ids\_presure, ids\_velocity] = res_2$
We use the same routine for $-\int_{\hat{T}} \nabla \cdot \phi_i \psi_j$
6. $(t, \phi)_{\Gamma_N}$:
Step 1: On local element following function is evaluated $res_1 = \hat{\phi}_i \cdot t$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \int_{\Gamma_N} res_1 EL(i,j)$
Step 3: Loop over element having Neumann boundary and perform following operation in each loop $res_3[ids] = res_2$
7. $(u_D, \phi)_{\Gamma_D}$:
Step 1: On local element following function is evaluated $res_1 = \hat{\phi}_i u_D$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \int_{\Gamma_D} res_1 grid.EL(i,j)$
Step 3: Loop over element having Dirichlet boundary and perform following operation in each loop $res_3[ids] = res_2$
8. $(\psi, n \cdot u_D)_{\Gamma_D}$:
Step 1: On local element following function is evaluated $res_1 = \hat{\psi} n \cdot u_D$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \int_{\Gamma_D} res_1 grid.EL(i,j)$
Step 3: Loop over element having Dirichlet boundary and perform following operation in each loop $res_3[ids] = res_2$
9. (f, ϕ):
Step 1: On local element following function is evaluated $res_1 = \hat{\phi} f$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \int_{\hat{\Omega}} res_1 2Ar(k)$
Step 3: Loop over element having Dirichlet boundary and perform following operation in each loop $res_3[ids] = res_2$
10. $(n \otimes u_D, \nabla\phi)_{\Gamma_D}$:
Step 1: On local element following function is evaluated $res_1 = n \otimes u_D \nabla\phi$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \int_{\Gamma_D} res_1 EL(i,j)$
Step 3: Loop over element having Dirichlet boundary and perform following operation in each loop $res_3[ids] = res_2$
We discuss now assembly of non linear terms. We now introduce initial guess u_k which will be iterated further.
11. $-((u_k \cdot \nabla)\phi, \phi)$:

Step 1: On local element following function is evaluated $res_1 = (u_k \cdot \nabla\hat{\phi}_i\hat{\phi}_j)$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = -\int_{\hat{T}}(res_1)(2Ar(k))$
Step 3: Loop over all elements and perform following operation in each loop $res_3[ids, ids] = res_2$
12. $((u_k \cdot n)\phi, \phi)_{\Gamma_N}$:
Step 1: On local element following function is evaluated $res_1 = (u_k \cdot n)\hat{\phi}_i\hat{\phi}_j$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \frac{1}{2}\int_{\Gamma_N} res_1 EL(i,j)$
Step 3: Loop over all Neumann edge and perform following operation in each loop $res_3[ids, ids] = res_2$
13. $((u_k \cdot n)\phi, \phi^{ext})_{\partial T\setminus\Gamma_N}$:
Step 1: On local element following function is evaluated $res_1 = (u_k \cdot n)\hat{\phi}_i\hat{\phi}_j^{ext}$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \frac{1}{2}\int_{\Gamma} res_1 EL(i,j)$
Step 3: Loop over all internal edge and Dirichlet edge and perform following operation in each loop $res_3[ids, ids^{ext}] = res_2$
14. $(|u_k \cdot n|\phi, \phi^{ext})_{\partial T\setminus\Gamma_N}$:
Step 1: On local element following function is evaluated $res_1 = (|u_k \cdot n|)\hat{\phi}_i\hat{\phi}_j^{ext}$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \frac{1}{2}\int_{\Gamma} res_1 EL(i,j)$
Step 3: Loop over all internal edge and Dirichlet edge and perform following operation in each loop $res_3[ids, ids^{ext}] = res_2$
15. $((u_k \cdot n)\phi, \phi)_{\partial T\setminus\Gamma_N}$:
Step 1: On local element following function is evaluated $res_1 = (u_k \cdot n)\hat{\phi}_i\hat{\phi}_j$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \frac{1}{2}\int_{\Gamma} res_1 EL(i,j)$
Step 3: Loop over all internal edge and Dirichlet edge and perform following operation in each loop $res_3[ids, ids] = res_2$
16. $((|u_k \cdot n|\phi, \phi)_{\partial T\setminus\Gamma_N}$:
Step 1: On local element following function is evaluated $res_1 = (|u_k \cdot n|)\hat{\phi}_i\hat{\phi}_j$ in accordance with equation 3.14
Step 2: An integral is performed over an element $res_2 = \frac{1}{2}\int_{\Gamma} res_1 EL(i,j)$
Step 3: Loop over all internal edge and Dirichlet edge and perform following operation in each loop $res_3[ids, ids] = res_2$

4.6 Setting boundary conditions

We extract the Dirichlet boundary edges and Neumann boundary 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.

4.7 Setting source term

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

equation 3.53 and equation 3.23.

4.8 Program flow

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

4.8.1 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`.

4.8.2 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 (4.1 and 4.2). These structures contain fields (as relevant to function space formulation) `pdeg` and `dimrange`. `pdeg` represents polynomial degree of Ansatz function and `dimrange` represents dimension of qunatity 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 viscocity in
`params.kinematic_viscosity`. Variable `mu` also holds the value of kinematic viscocity. We define penalty parameter in variable C_{11}. Bool variable `show_sparsity` plots sparsity pattern of each matrix if set to true.

4.8.3 Matrix assembly

We now assemble all the matrices as explained during matrix assembly (routine `assemble_stifness_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]`.

4.8.4 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_residua` 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.

4.8.5 Post processing

We now enter into `stifness_matrix_test`, to check properties of assembled stiffness matrix as explained in section 3.9.2. 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_derivati` contain anlytical expression for derivative to be used for H_0 norm.

4.8.6 Newton method

We now enter into newton method (Section 3.9.5). This is implemented in script `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.

4.8.7 Additional remarks

We plot the solution using `ldg_plot` written for plotting discontinuous functions. The Dirichlet values are defined in `dirichlet_values`, the Neumann values are defined in `neumann_values` and source function is defined in `func_rhs`.

4.9 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 4.5.

Table 4.5: Size and sparsity patterns of different terms

Matrix term	**Size**	**Sparsity pattern**
$(\nabla\phi, \nabla\phi)$	$\mathbb{R}^{u_{ndofs} \times u_{ndofs}}$	Figure 4.20
$([n \otimes \phi], [n \otimes \phi])_{\Gamma \cup \Gamma_D}$	$\mathbb{R}^{u_{ndofs} \times u_{ndofs}}$	Figure 4.6
$(\{\nabla\phi\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D}$	$\mathbb{R}^{u_{ndofs} \times u_{ndofs}}$	Figure 4.11
$(\{\psi\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}$	$\mathbb{R}^{p_{ndofs} \times u_{ndofs}}$	Figure 4.16
$(-\int_{\hat{T}} \psi \nabla \cdot \phi)$	$\mathbb{R}^{p_{ndofs} \times u_{ndofs}}$	Figure 4.17
$-((u_k \cdot \nabla)\phi, \phi)$	$\mathbb{R}^{u_{ndofs} \times u_{ndofs}}$	Figures 4.18
$((u_k \cdot n)\phi, \phi)_{\Gamma_N}$	$\mathbb{R}^{u_{ndofs} \times u_{ndofs}}$	Figures 4.19
$((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N}$	$\mathbb{R}^{u_{ndofs} \times u_{ndofs}}$	Figures 4.21a
$(\lvert u_k \cdot n \rvert \phi, \phi^{ext})_{\partial T \setminus \Gamma_N}$	$\mathbb{R}^{u_{ndofs} \times u_{ndofs}}$	Figures 4.22a
$((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$	$\mathbb{R}^{u_{ndofs} \times u_{ndofs}}$	Figures 4.21b
$(\lvert u_k \cdot n \rvert \phi, \phi)_{\partial T \setminus \Gamma_N}$	$\mathbb{R}^{u_{ndofs} \times u_{ndofs}}$	Figures 4.22b

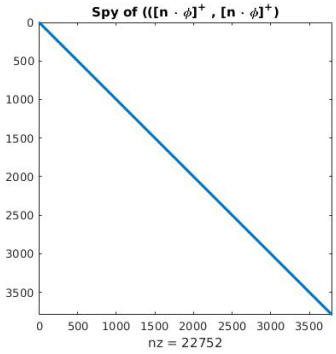


Figure 4.1: $((n \otimes \phi)^+, (n \otimes \phi)^+)_{\Gamma \cup \Gamma_D}$

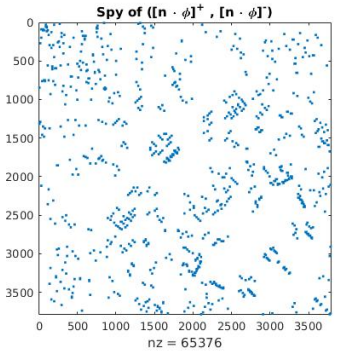


Figure 4.2: $((n \otimes \phi)^+, (n \otimes \phi)^-)_{\Gamma \cup \Gamma_D}$

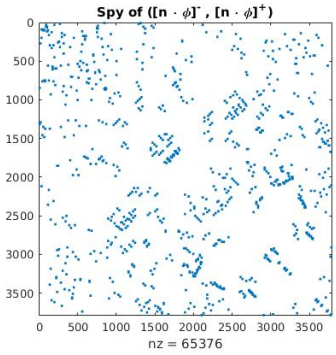


Figure 4.3: $((n \otimes \phi)^-, (n \otimes \phi)^+)_{\Gamma \cup \Gamma_D}$

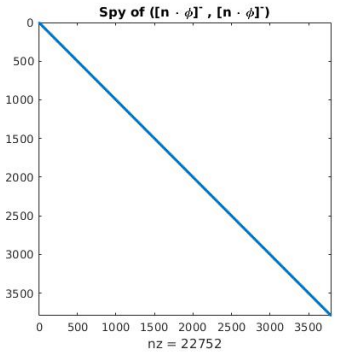
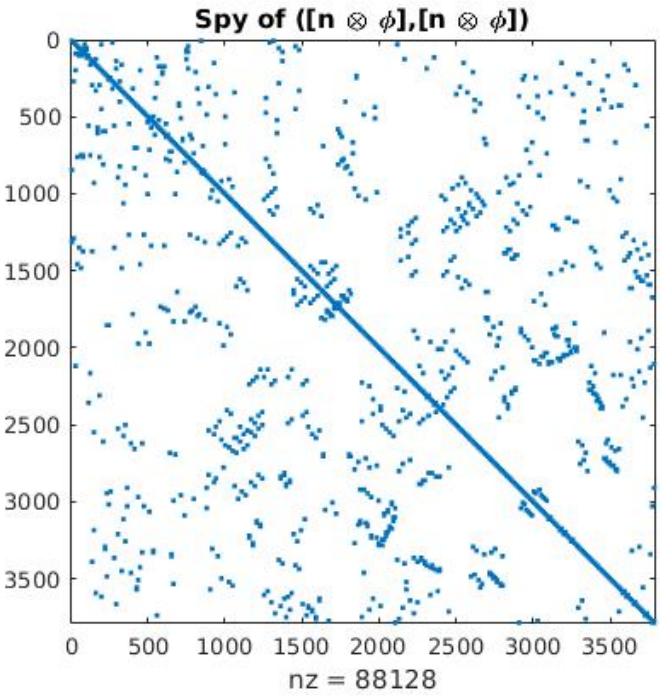


Figure 4.4: $((n \otimes \phi)^-, (n \otimes \phi)^-)_{\Gamma \cup \Gamma_D}$



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

Figure 4.5: Sparsity patterns of constituents of $([n \otimes \phi], [n \otimes \phi])_{\Gamma \cup \Gamma_D}$

Figure 4.6

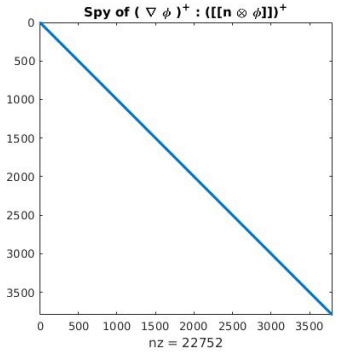


Figure 4.7: $(\nabla\phi^+, (n \otimes \phi)^+)_{\Gamma \cup \Gamma_D}$

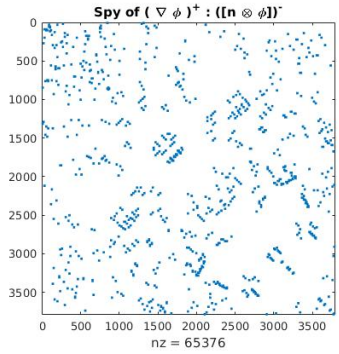


Figure 4.8: $(\nabla\phi^+, (n \otimes \phi)^-)_{\Gamma \cup \Gamma_D}$

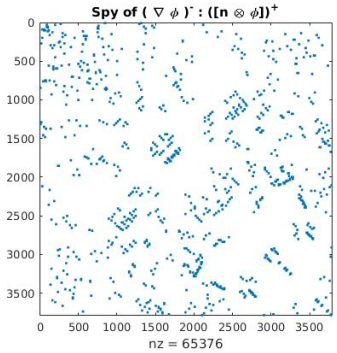


Figure 4.9: $(\nabla\phi^-, (n \otimes \phi)^+)_{\Gamma \cup \Gamma_D}$

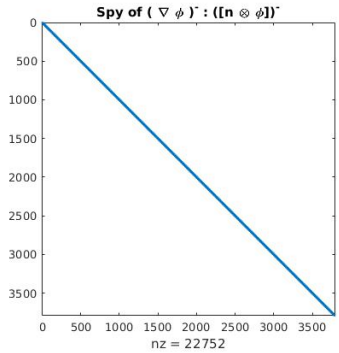
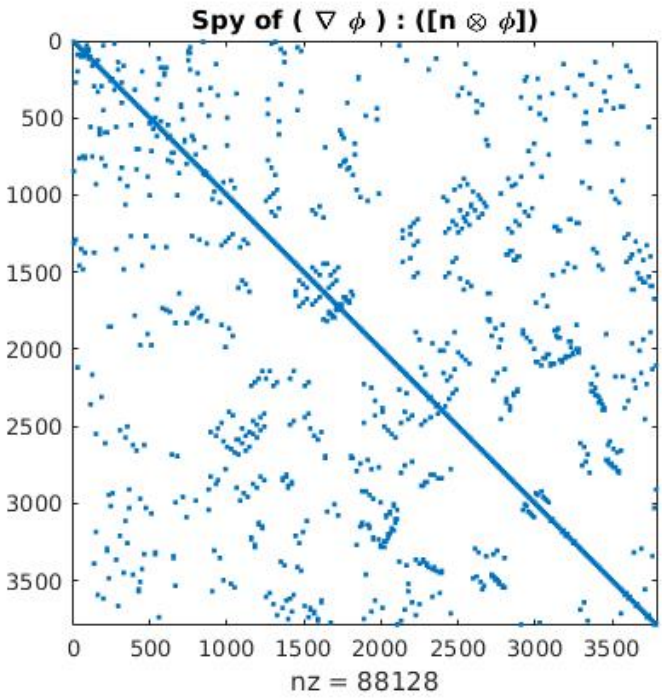


Figure 4.10: $(\nabla\phi^-, (n \otimes \phi)^-)_{\Gamma \cup \Gamma_D}$



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

Figure 4.11: Sparsity patterns of constituents of $(\{\nabla\phi\}, [n \otimes \phi])_{\Gamma \cup \Gamma_D}$

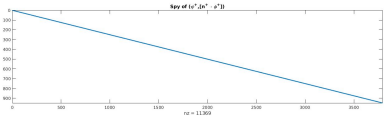


Figure 4.12: $(\psi^+, (n \cdot \phi)^+)_{\Gamma \cup \Gamma_D}$

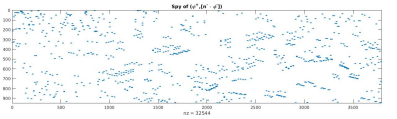


Figure 4.13: $(\psi^+, (n \cdot \phi)^-)_{\Gamma \cup \Gamma_D}$

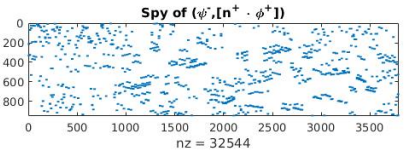


Figure 4.14: $(\psi^-, (n \cdot \phi)^+)_{\Gamma \cup \Gamma_D}$

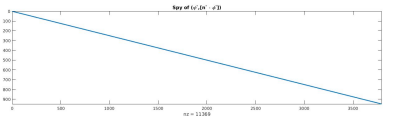
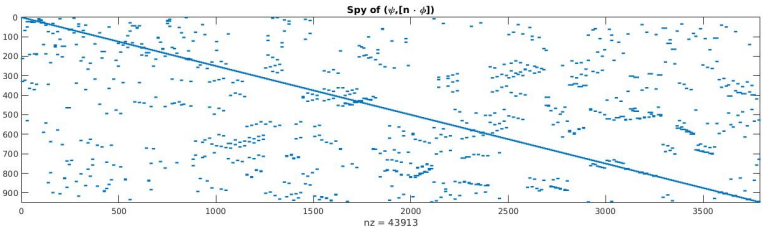


Figure 4.15: $(\psi^-, (n \cdot \phi)^-)_{\Gamma \cup \Gamma_D}$



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

Figure 4.16: Sparsity patterns of constituents of $(\{\psi\}, [n \cdot \phi])_{\Gamma \cup \Gamma_D}$

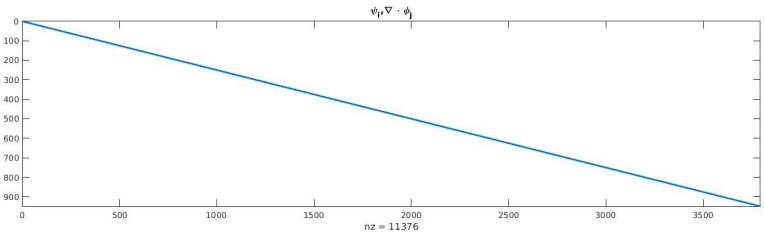


Figure 4.17: Sparsity pattern of $(\psi, \nabla \cdot \phi)$

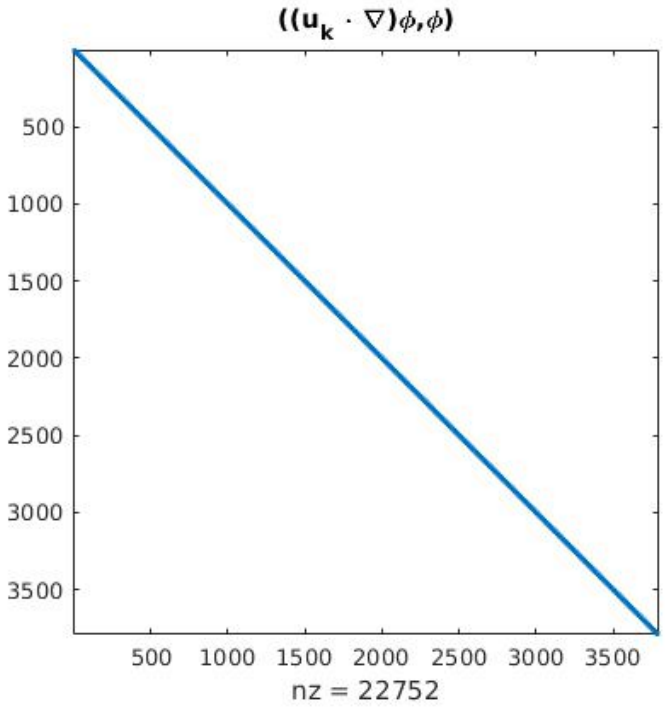


Figure 4.18: Sparsity pattern of $((u_k \cdot \nabla)\phi, \phi)$

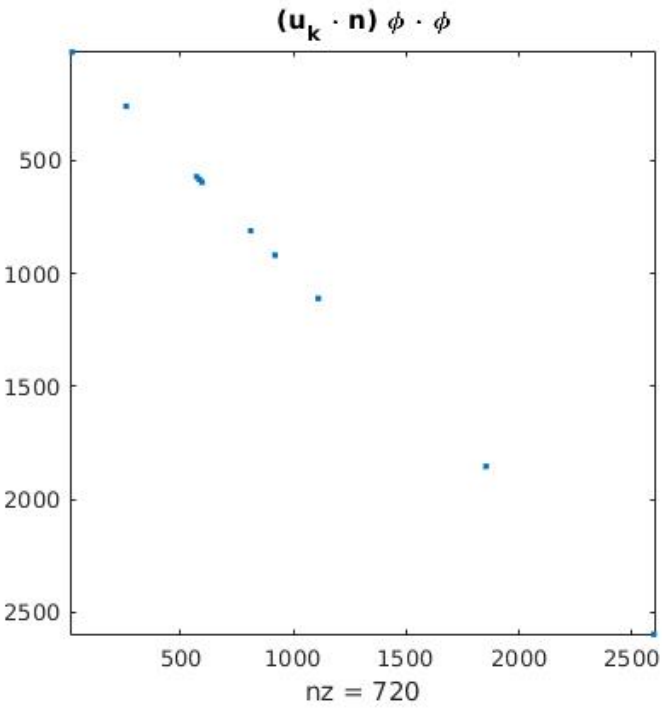


Figure 4.19: Sparsity pattern of $((u_k \cdot n)\phi, \phi)_{\Gamma_N}$

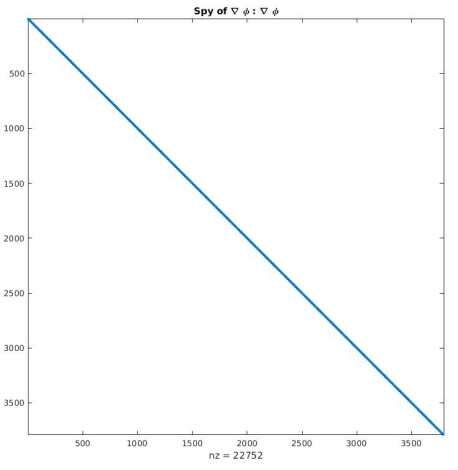
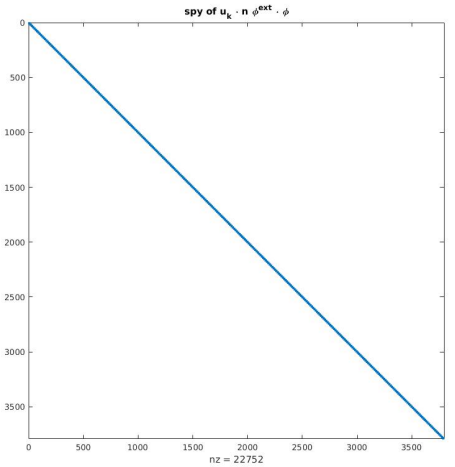
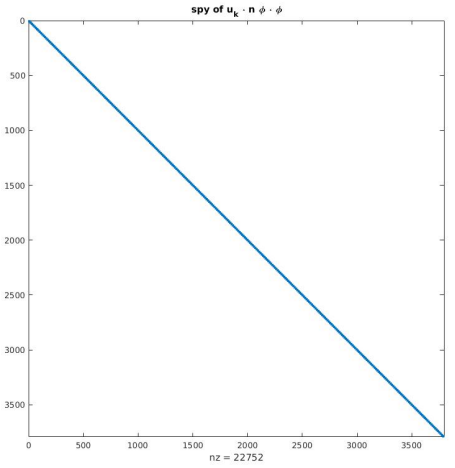


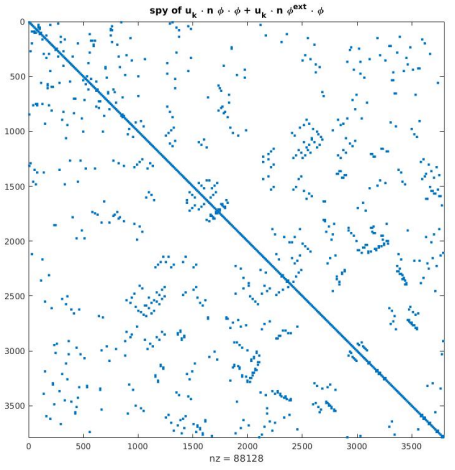
Figure 4.20: Sparsity pattern of $(\nabla\phi, \nabla\phi)$



(a) $(((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N})$

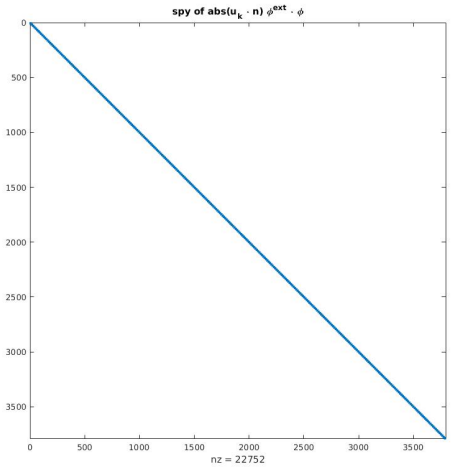


(b) $(((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N})$

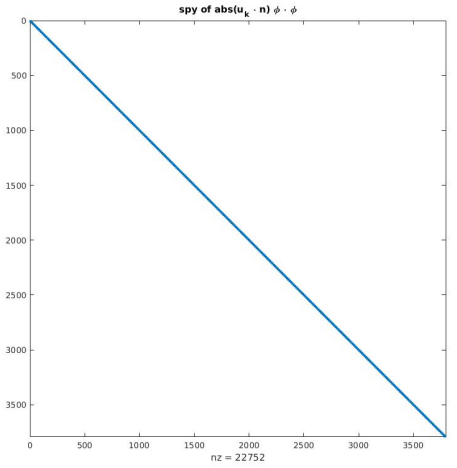


(c) $((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + ((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$

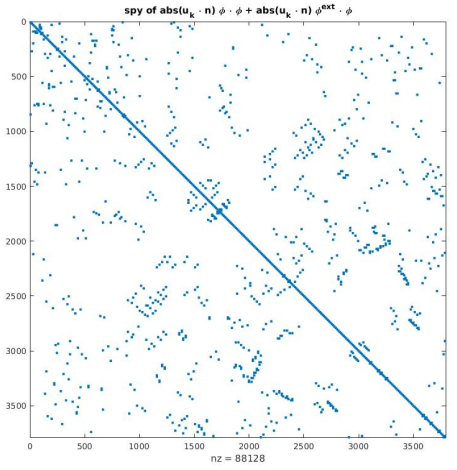
Figure 4.21: Sparsity patterns of constituents of $((u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + ((u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$



(a) $((abs(u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N})$



(b) $((abs(u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N})$



(c) $(abs(u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + (abs(u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$

Figure 4.22: Sparsity patterns of constituents of $(abs(u_k \cdot n)\phi, \phi^{ext})_{\partial T \setminus \Gamma_N} + (abs(u_k \cdot n)\phi, \phi)_{\partial T \setminus \Gamma_N}$

Chapter 5

Numerical experiments

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

5.1 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,

$$P_{error,W} = ||P - P_h||_W \quad . \tag{5.1}$$

In the present analysis, we measure the error in the L^2 norm and the H_0 semi norm and present results of an $h-$convergence test.
The L^2 norm of the error is measured as,

$$P_{error,L^2} = \int_\Omega |P - P_h|^2 \quad . \tag{5.2}$$

The H_0 semi norm of the error is defined as,

$$P_{error,H_0} = \sum_{k=1}^{nel} \int_{\tau_k} |\nabla P - \nabla P_h|^2 \quad . \tag{5.3}$$

We use the notations from the Section 3.9.4 and the Section 3.9.1.

5.2 Stokes flow

5.2.1 Properties of Stiffness matrix

We recall now some conclusions from the Section 3.9.2 and the Section 3.9.6. 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. It 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, $tol > 0$.

5.2.2 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,

$$p = (1 - x) \tag{5.4}$$

$$u = (y(1 - y), 0) \quad . \tag{5.5}$$

The grid is equally divided in both directions with number of divisions in each direction as $5, 10, 15, 20$ giving step size of each element in each of the $x-y$ direction as $0.2, 0.1, 0.067, 0.05$.
The results of an $h-$convergence test in the L^2 norm is presented in Figures 5.1a, 5.2a, 5.1b and 5.2b and in the H_0 semi norm is presented in Figures 5.3a, 5.4a, 5.3b and 5.4b.
We now present additional examples and check whether the implementation of the Stokes flow is capable of reproducing the physics of the problem.

5.2.3 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,

$$u = (10x, 0)^T \quad \text{for} \quad 0 \leq x \leq 0.1 \tag{5.6}$$

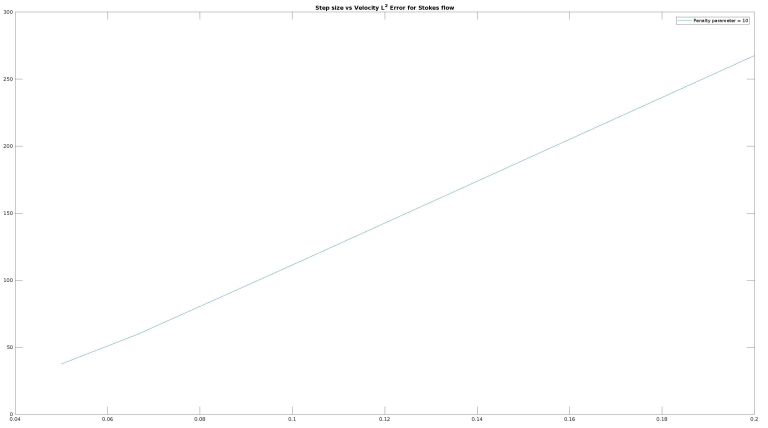
$$u = (1, 0)^T \quad \text{for} \quad 0.1 \leq x \leq 0.9 \tag{5.7}$$

$$u = (10 - 10x, 0)^T \quad \text{for} \quad 0.9 \leq x \leq 1 \quad . \tag{5.8}$$

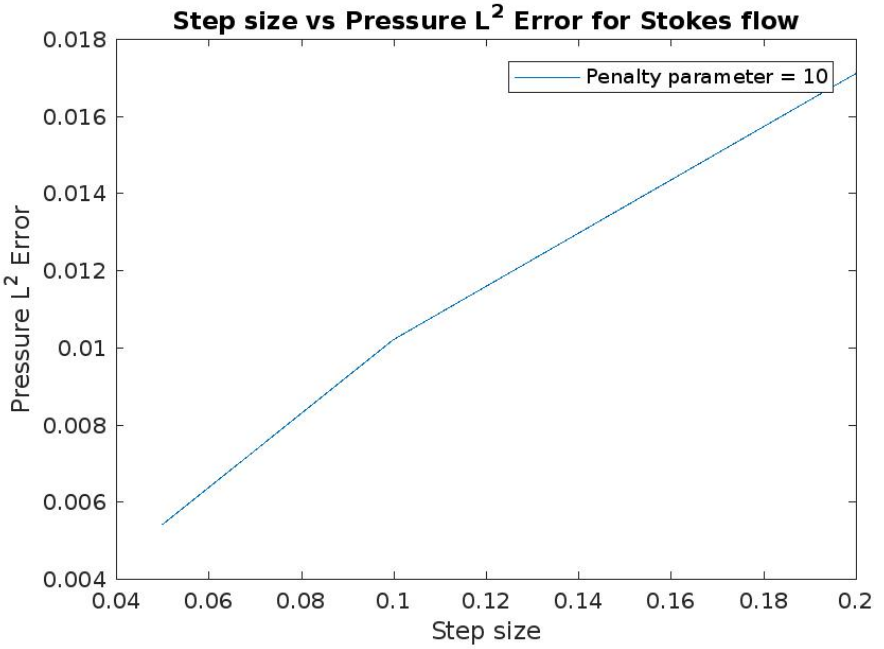
The results are shown in Figures 5.8, 5.12 and 5.16.

5.2.4 Flow over cylinder

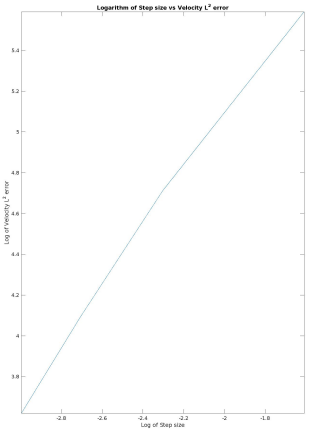
The domain considered for this example is the unit square [0,1] $\times$ [0,1] with a cut out cylinder of diameter 0.2 centered at $(0.5, 0.5)$ i.e. the center of cylinder coincides with the center of the square in the $x - y$ plane. The boundary $x = 0$ is Dirichlet boundary with inflow velocity at 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 5.20, 5.24 and 5.28 give physically relevant result for example, low pressure zone after cylinder, high pressure zone before cylinder and wake zone after cylinder for velocity.



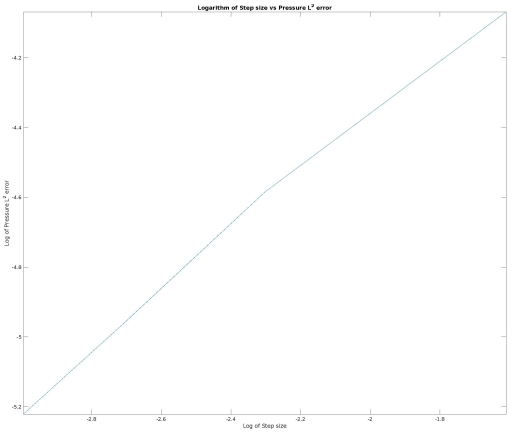
(a) $h-$ convergence test for velocity L^2 error



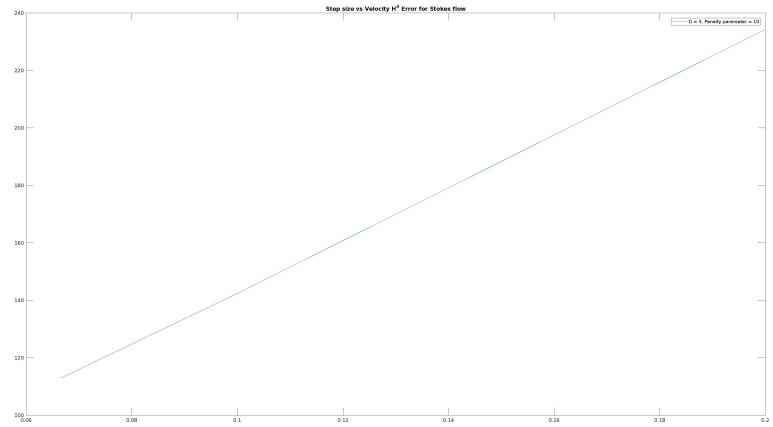
(b) $h-$ convergence test for pressure in L^2 error



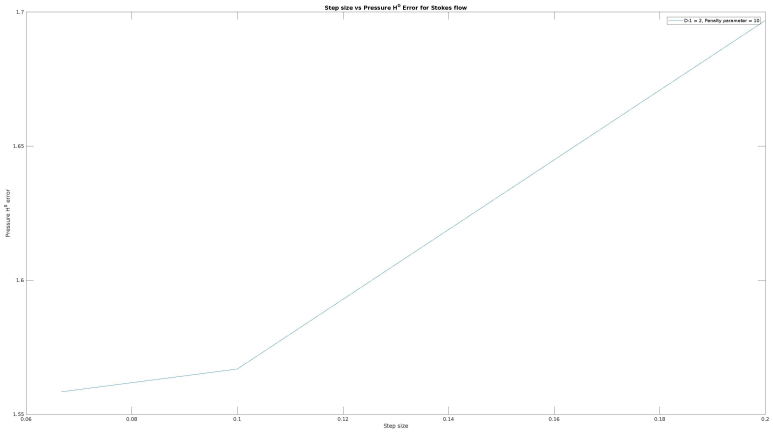
(a) $h-$ convergence test for velocity L^2 error (logarithmic scale)



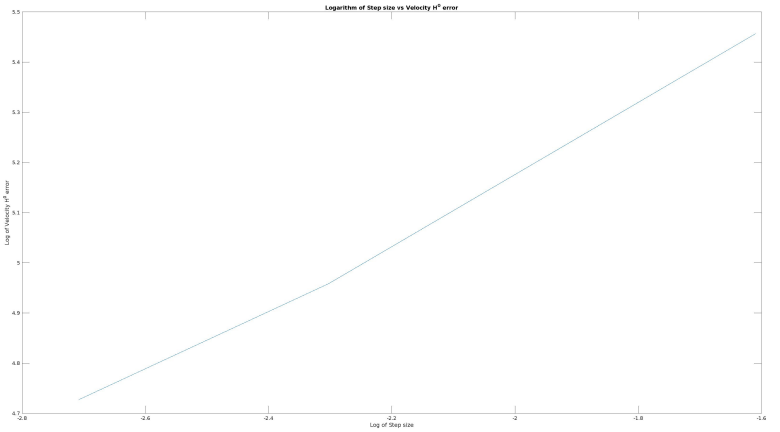
(b) $h-$ convergence test for pressure in L^2 error (logarithmic scale)



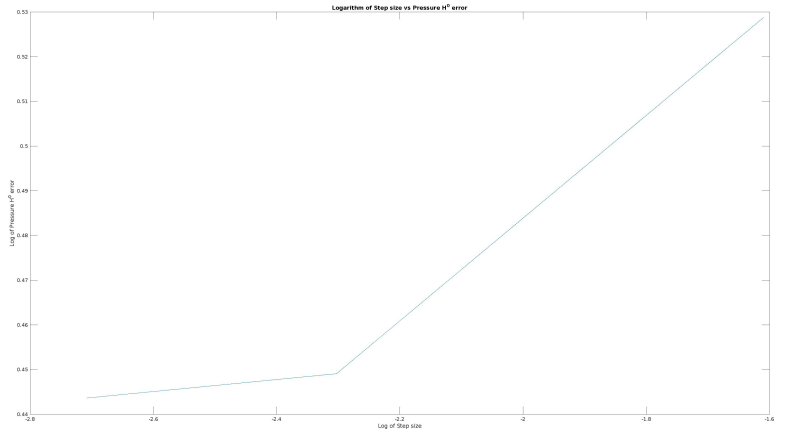
(a) $h-$ convergence test for velocity h^0 error



(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)

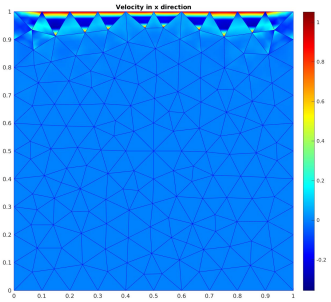


Figure 5.5: $x-$ velocity (bicgstab solver)

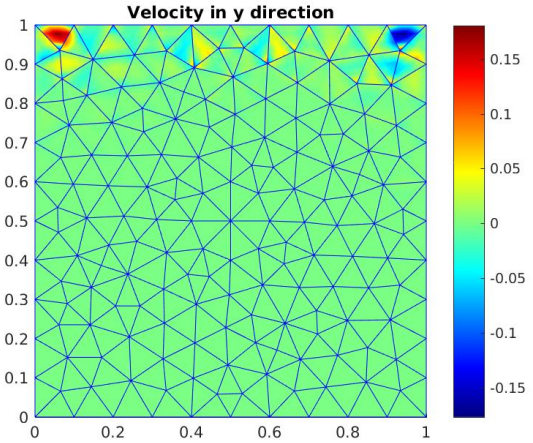


Figure 5.6: $y-$ velocity (bicgstab solver)

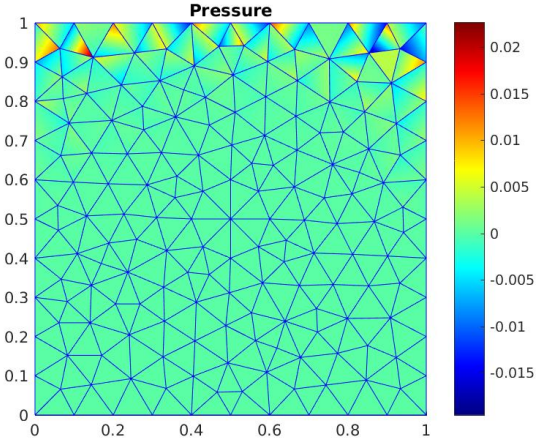


Figure 5.7: Pressure (bicgstab solver)

Figure 5.8

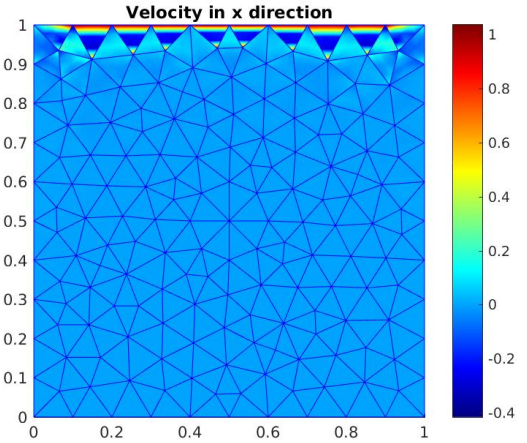


Figure 5.9: $x-$ velocity (minres solver)

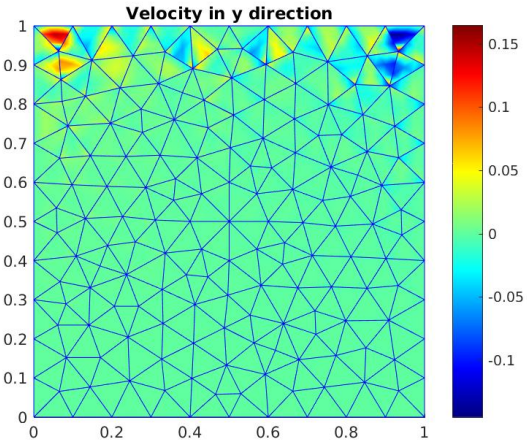


Figure 5.10: $y-$ velocity (minres solver)

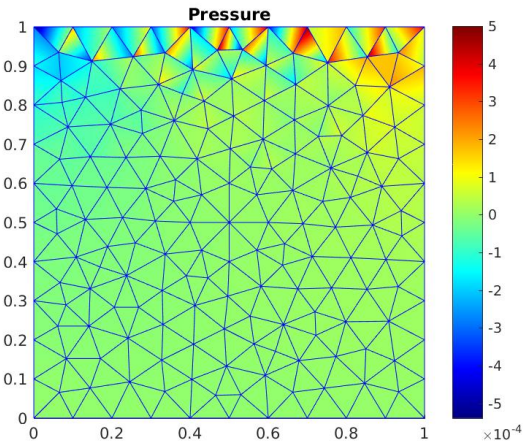


Figure 5.11: Pressure (minres solver)

Figure 5.12

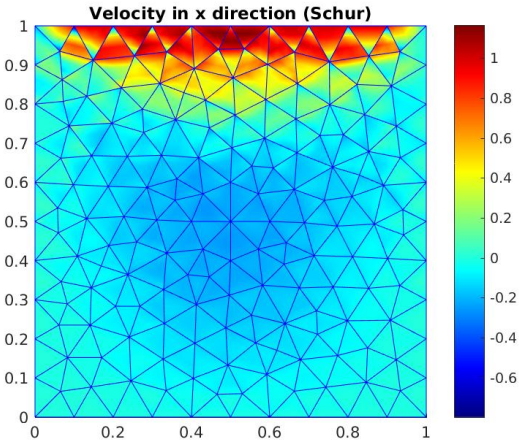


Figure 5.13: $x-$ velocity (Schur complement method)

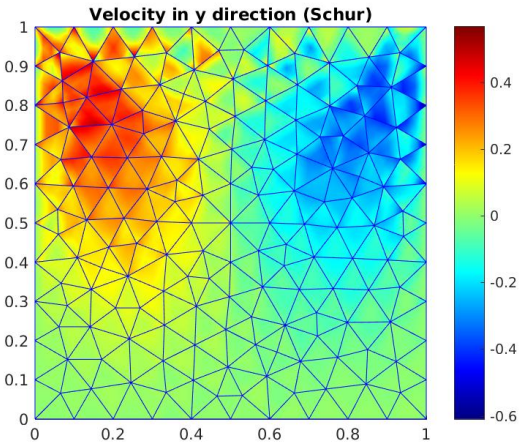


Figure 5.14: $y-$ velocity (Schur complement method)

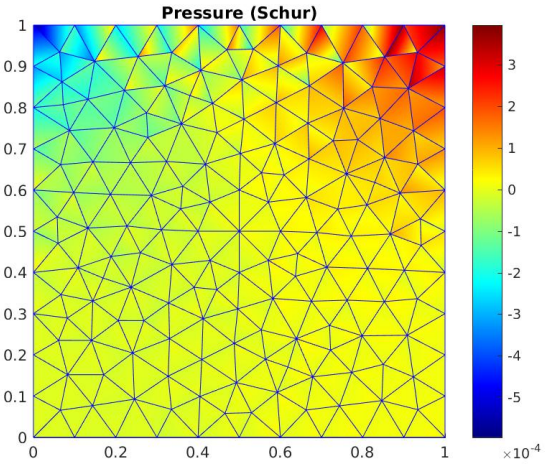


Figure 5.15: Pressure (Schur complement method)

Figure 5.16

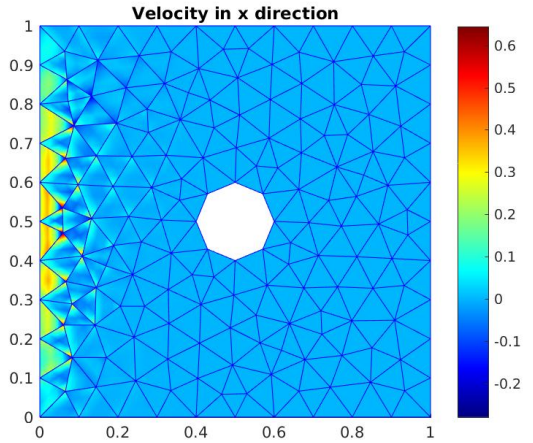


Figure 5.17: $x-$ velocity (bicgstab solver)

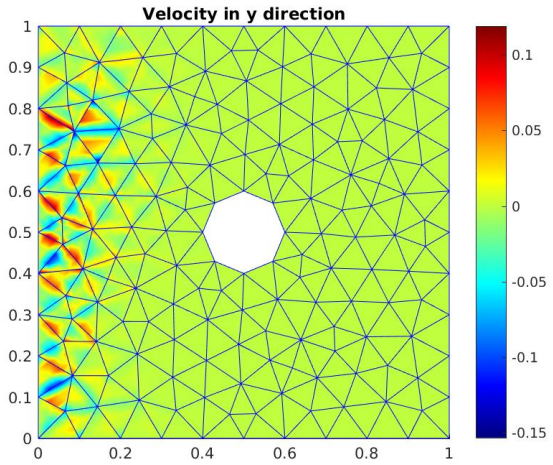


Figure 5.18: $y-$ velocity (bicgstab solver)

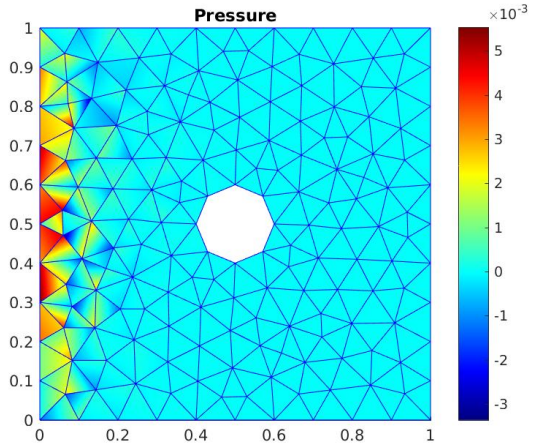


Figure 5.19: Pressure (bicgstab solver)

Figure 5.20

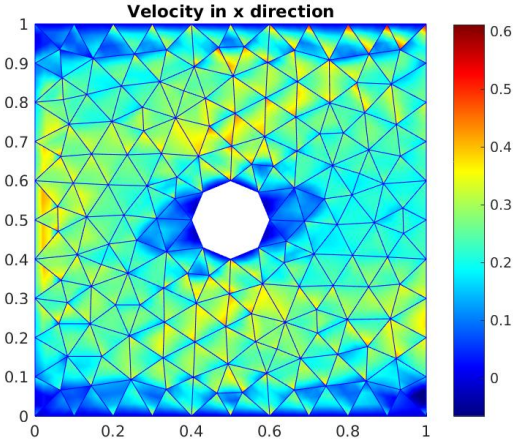


Figure 5.21: $x-$ velocity (minres solver)

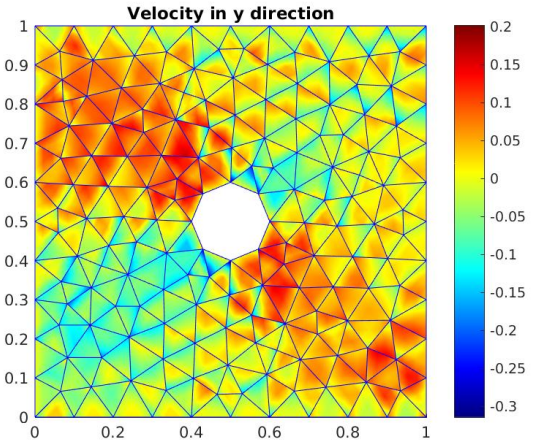


Figure 5.22: $y-$ velocity (minres solver)

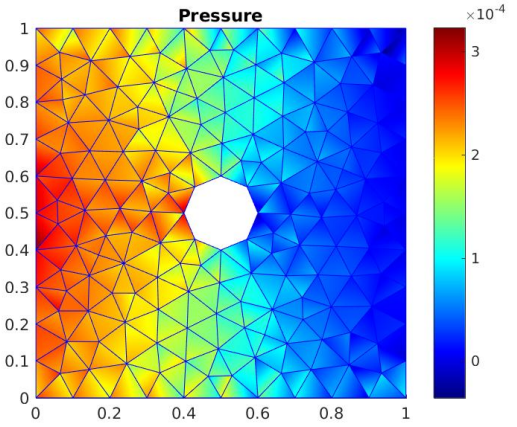


Figure 5.23: Pressure (minres solver)

Figure 5.24

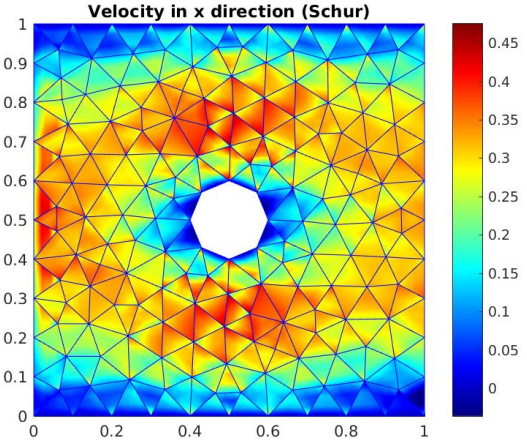


Figure 5.25: $x-$ velocity (Schur complement method)

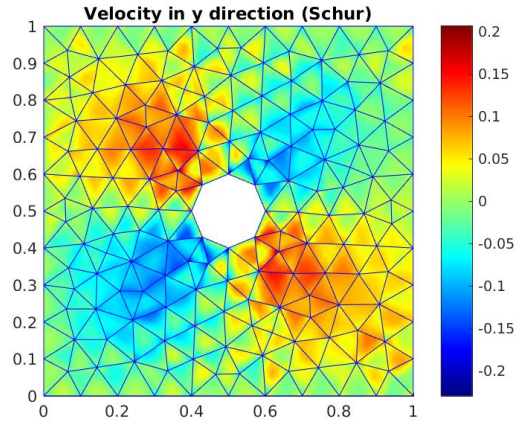


Figure 5.26: $y-$ velocity (Schur complement method)

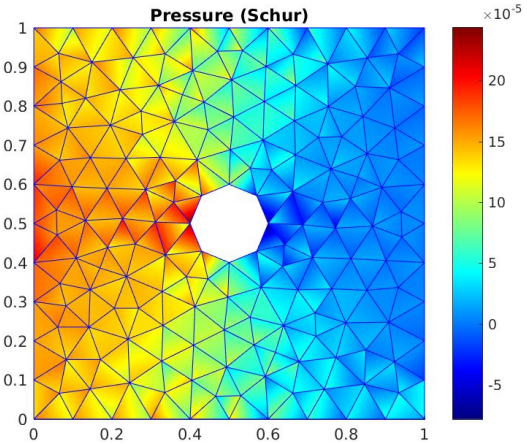


Figure 5.27: Pressure (Schur complement method)

Figure 5.28

5.3 Penalty parameter

We now measure the effect of the penalty parameter on the condition number of the matrix. 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 5.29b, 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. The penalty parameter ranged from $1e-2$ to $1e4$. The corresponding condition number ranged from $3.750e5$ to $1.445e10$. We see a linear increase in the condition number of the stiffness matrix with respect to the penalty parameter. (Figure 5.29)

5.4 Navier Stokes flow

We recall that we use the initial guess from the Stokes equation and the Newton method presented in the Section 3.9.5. We also note that the stiffness matrix in case of the Navier Stokes equation is non symmetric and therefore, solvers applicable only for symmetric matrices can not be used.

5.4.1 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,

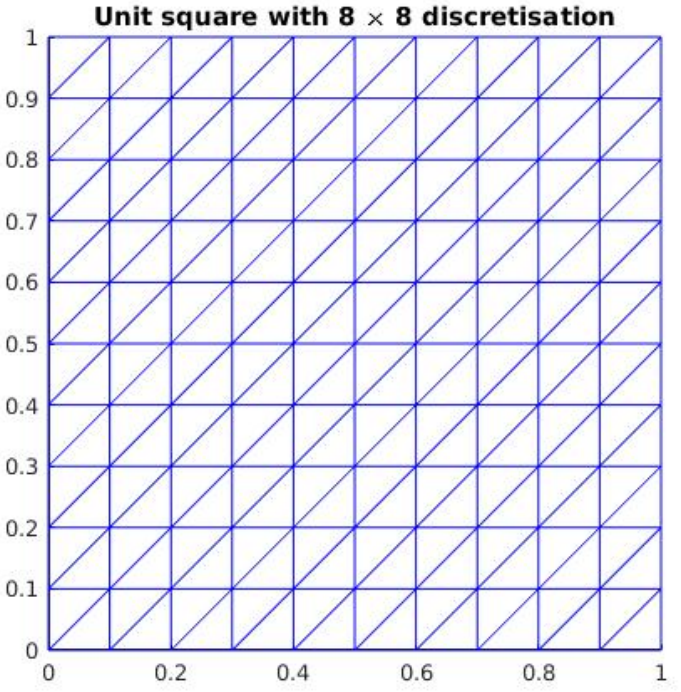
$$\begin{aligned} f = (&-4\nu(1+2y)(y^2 - 6xy^2 + 6x^2y^2 - y + 6xy \\ &-6x^2y + 3x^2 - 6x^3 + 3x^4) + 1 - 2x \\ &+4x^3y^2(2y^2 - 2y + 1)(y-1)^2(-1+2x)(x-1)^3, \\ &4\nu(-1+2x)(x^2 - 6x^2y + 6x^2y^2 - x + 6xy \\ &-6xy^2 + 3y^2 - 6y^3 + 3y^4) + \\ &4x^2y^3(-1+2y)(y-1)^3(2x^2 - 2x + 1)(x-1)^2) \quad . \end{aligned} \tag{5.9}$$

The analytical solution for pressure and velocity reads as,

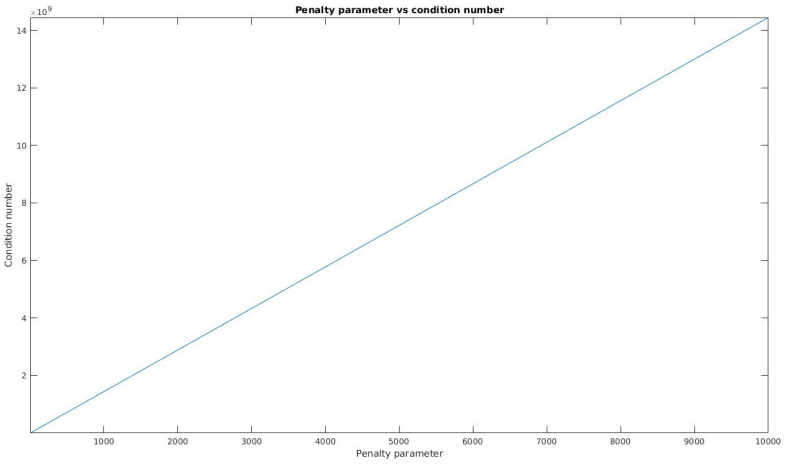
$$p = x(1-x) \tag{5.10}$$

$$\begin{aligned} u = (&x^2(1-y)^2(2y - 6y^2 + 4y^3), \\ &-y^2(1-y)^2(2x - 6x^2 + 4x^3)) \quad . \end{aligned} \tag{5.11}$$

The grid is equally divided in both directions with number of divisions in each direction as $10, 15, 17, 20$ giving step size of each element in each of $x-y$ direction as $0.1, 0.0667, 0.0558, 0.05$.



(a) Unit square with 8×8 discretisation



(b) Penalty parameter vs Condition number

Figure 5.29

The results of an $h-$ convergence test in the L^2 norm is presented in Figures 5.30, 5.31 and in the H_0 semi norm is presented in Figures 5.32, 5.33.

5.4.2 Lid-driven cavity problem

We again consider the example presented in the Section 5.2.3 i.e. Benchmark CFD problem. The results are presented in Figures 5.37, 5.41 and 5.45. We also see the effect of the initial guess, the solution for the Stokes flow, calculated by different solvers.

5.4.3 Flow over cylinder

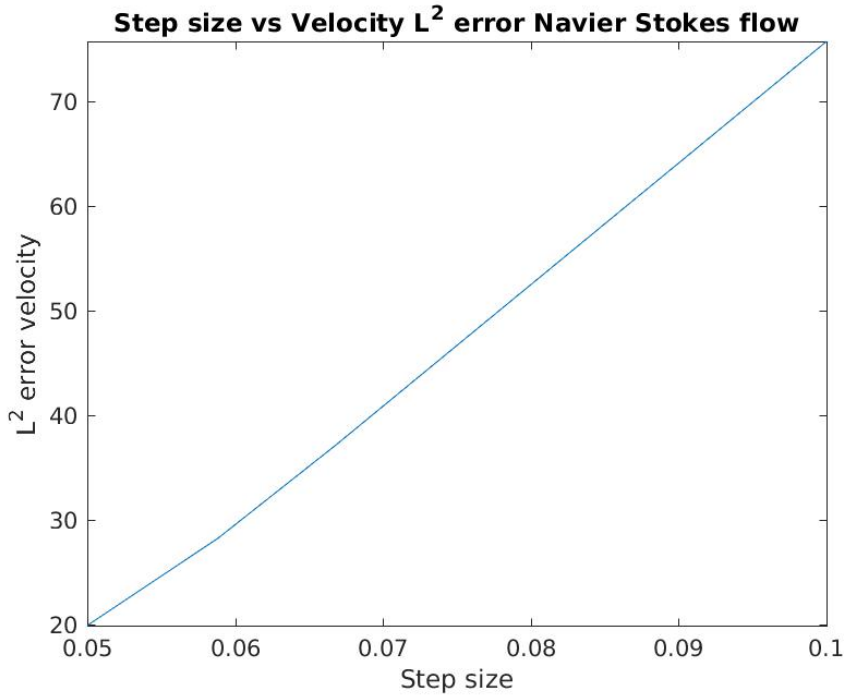
We again consider the example presented in the Section 5.2.4 . However, we multiply the inlet velocity on left boundary with certain factor to regulate the Reynolds number. The results are presented in Figures 5.49, 5.53 and 5.57. We also see the effect of the initial guess, the solution for the Stokes flow, calculated by different solvers.

5.5 Solver selection

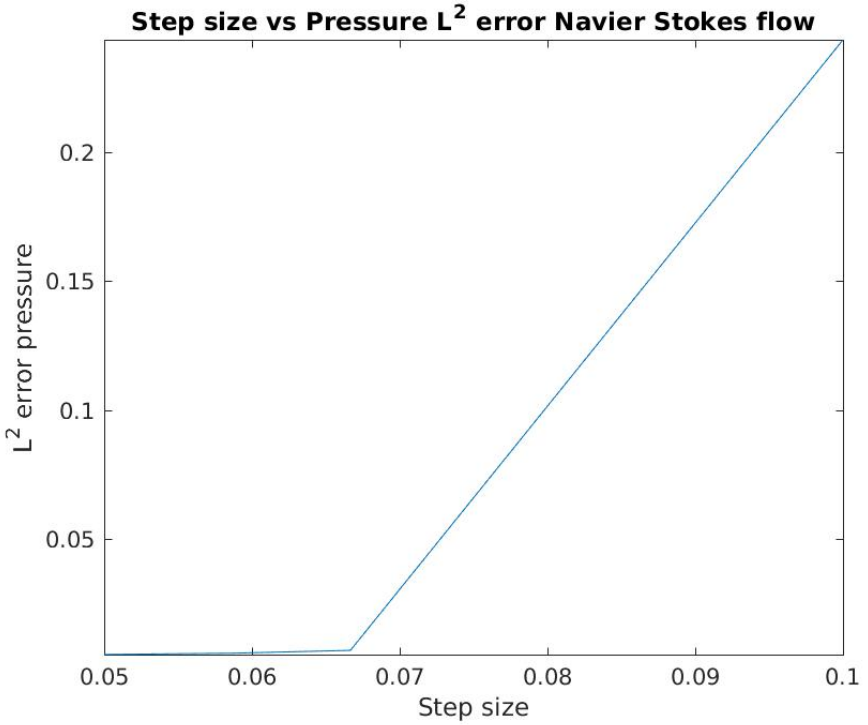
As demonstrated earlier in the Sections 5.2 and 5.4, the use of different solvers leads to different solutions. We also present now an example measuring the residual vs. run time for different solvers for the same problem.
We again consider the problem from the Section 5.2.4. The residual and the run 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. Residual is measured as $\frac{norm(B-AX,2)}{norm(B,2)}$ where $norm(\cdot, 2)$ refers to $2-$norm of vector.

Solver/Method	**Residual**	**Run time**
Schur complement method	2.4436e-08	6.6253 Seconds
minres	2.4618e-05	35.7372 Seconds
bicgstab	9.0071e-05	58.3472 Seconds

It is to be noted that *bicgstab* stops before convergence and without reaching to maximum number of iterations.

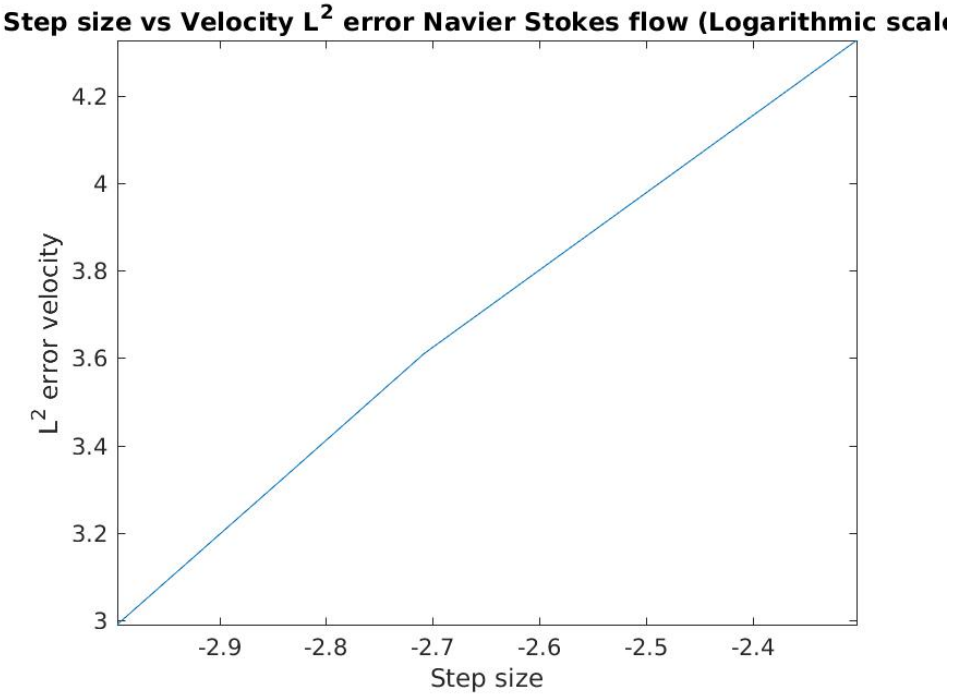


(a) $h-$ convergence test for velocity L^2 error

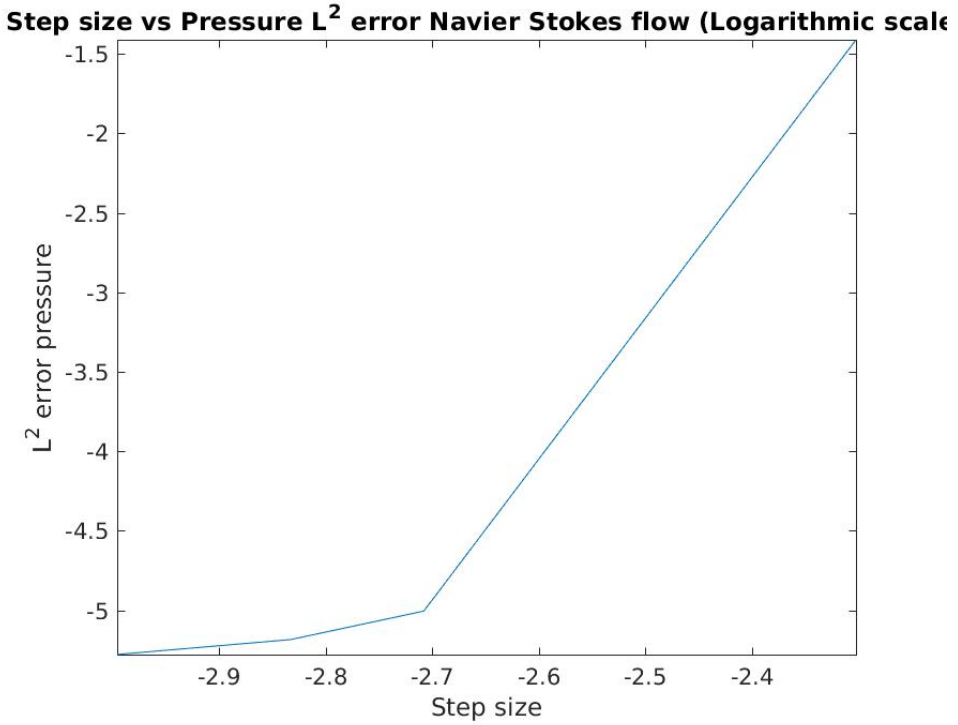


(b) $h-$ convergence test for pressure in L^2 error

Figure 5.30

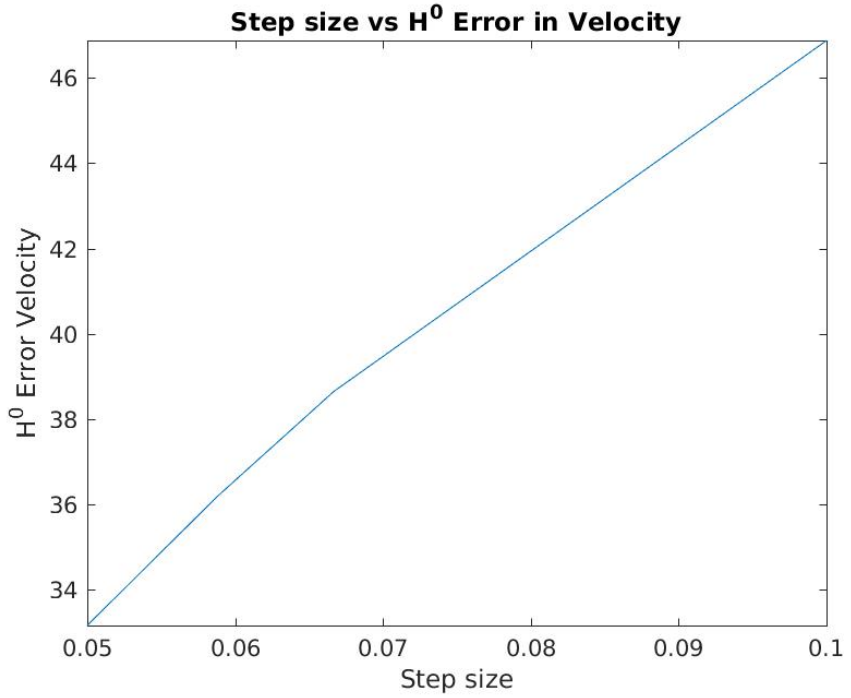


(a) $h-$ convergence test for velocity L^2 error (logarithmic scale)

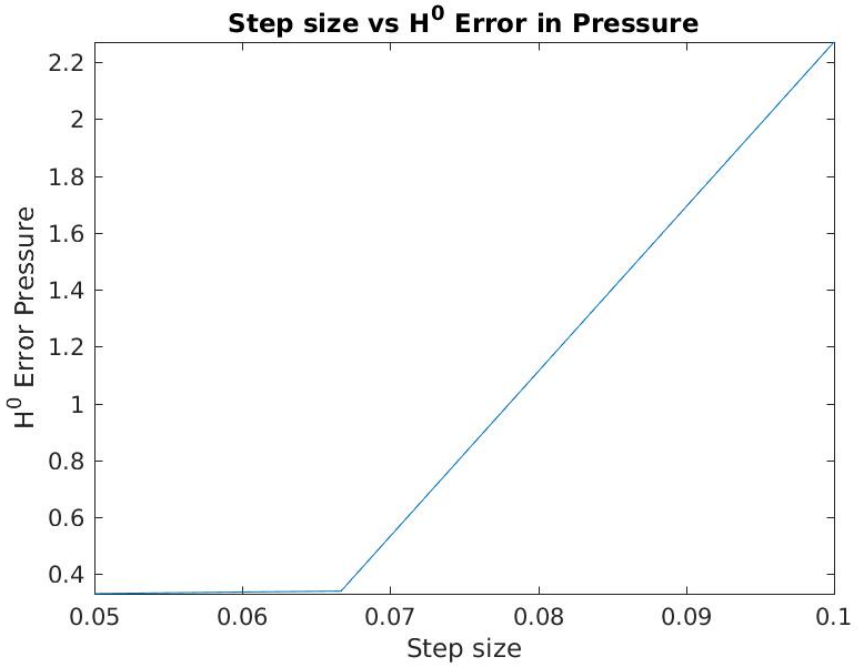


(b) $h-$ convergence test for pressure in L^2 error (logarithmic scale)

Figure 5.31

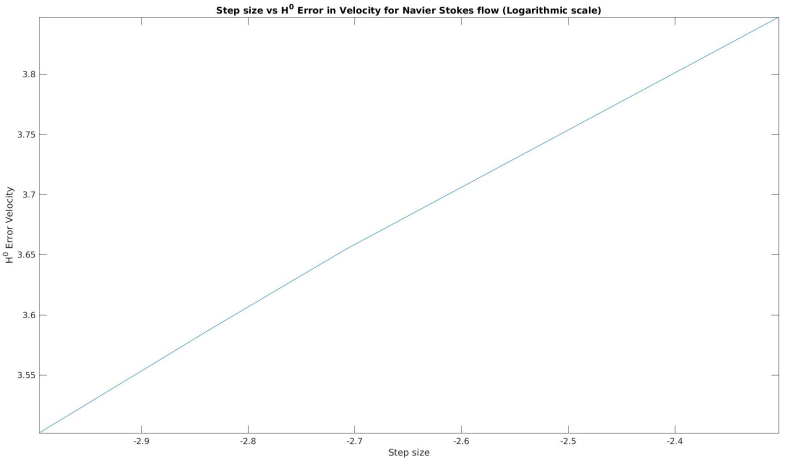


(a) $h-$ convergence test for velocity H^0 error

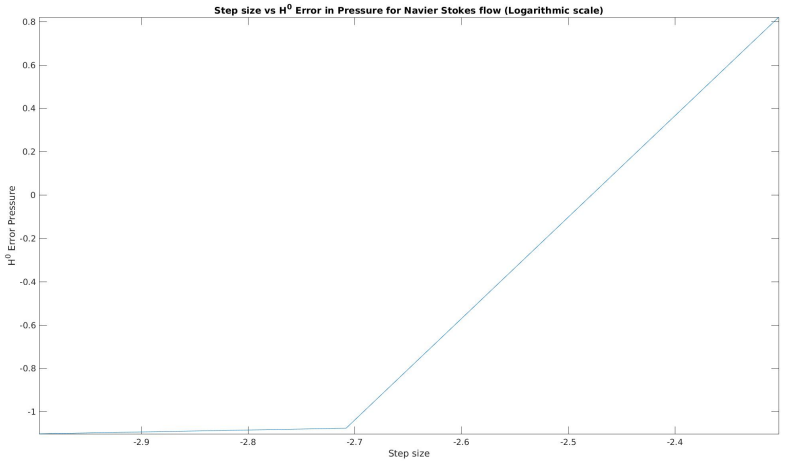


(b) $h-$ convergence test for pressure in H^0 error

Figure 5.32



(a) $h-$ convergence test for velocity H^0 error (logarithmic scale)



(b) $h-$ convergence test for pressure in H^0 error (logarithmic scale)

Figure 5.33

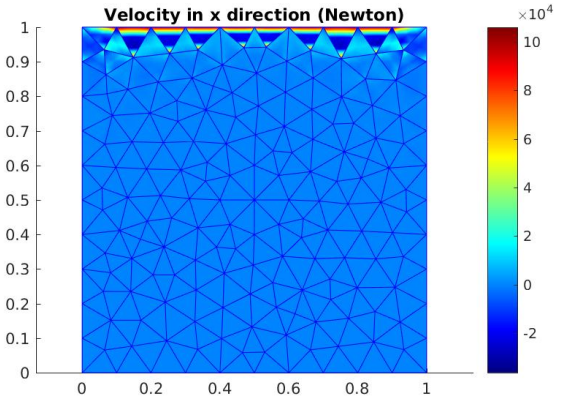


Figure 5.34: $x-$ velocity (Initial guess by bicgstab solver)

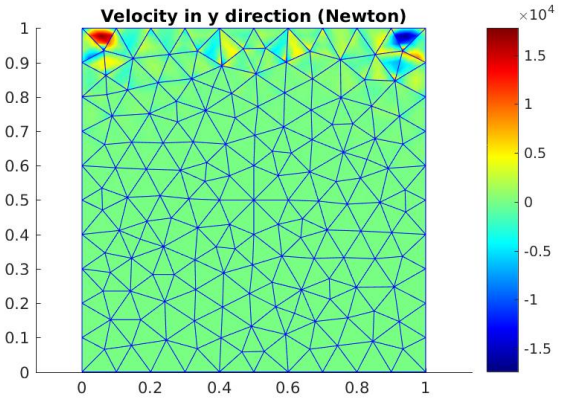


Figure 5.35: $y-$ velocity (Initial guess by bicgstab solver)

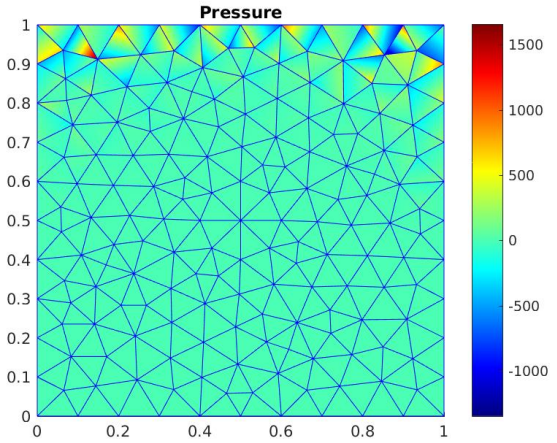


Figure 5.36: Pressure (Initial guess by bicgstab solver)

Figure 5.37

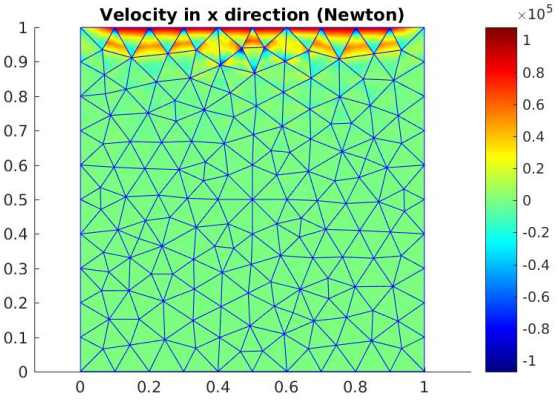


Figure 5.38: $x-$ velocity (Initial guess by minres solver)

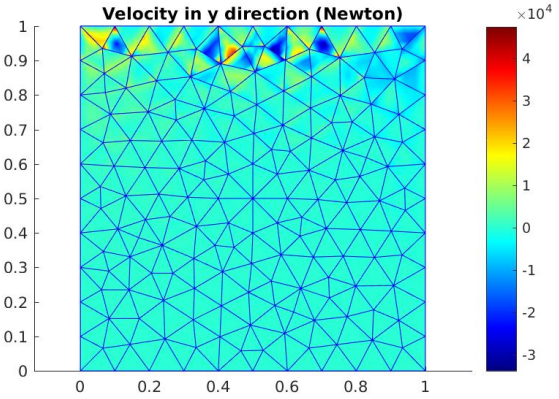


Figure 5.39: $y-$ velocity (Initial guess by minres solver)

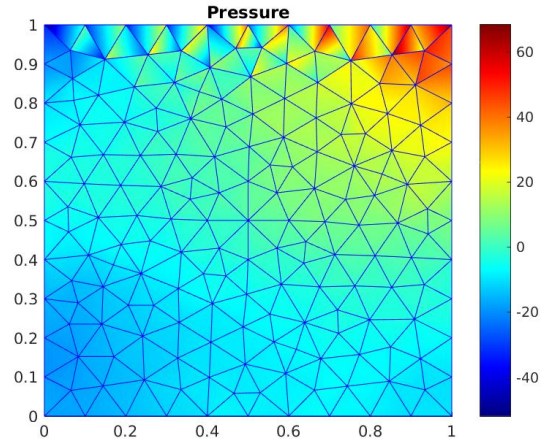


Figure 5.40: Pressure (Initial guess by minres solver)

Figure 5.41

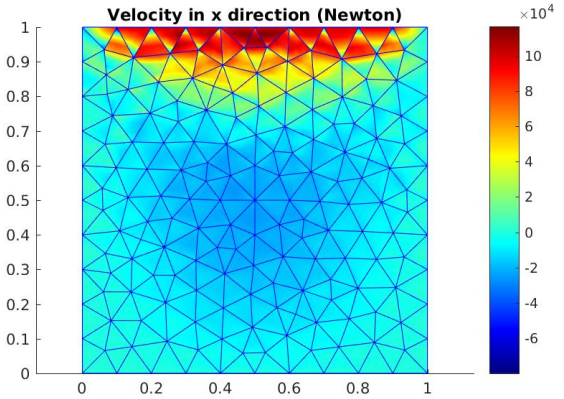


Figure 5.42: $x-$ velocity (Initial guess by Schur complement method)

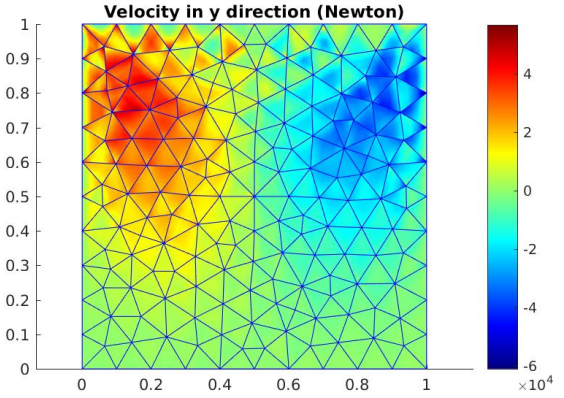


Figure 5.43: $y-$ velocity (Initial guess by Schur complement method)

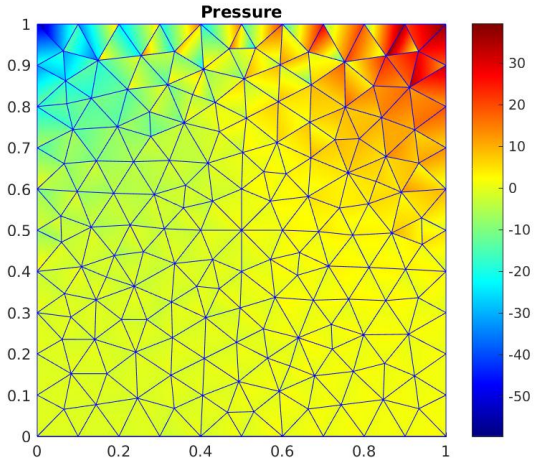


Figure 5.44: Pressure (Initial guess by Schur complement method)

Figure 5.45

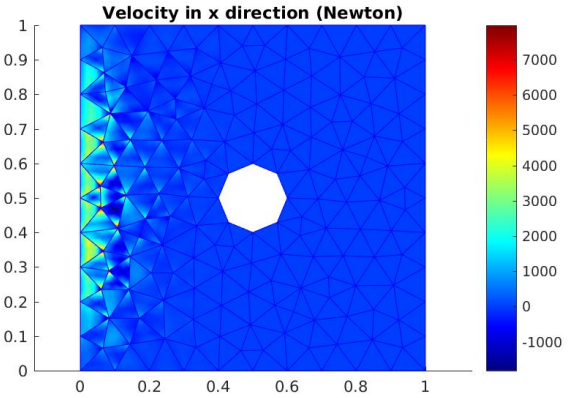


Figure 5.46: $x-$ velocity (Initial guess by bicgstab solver)

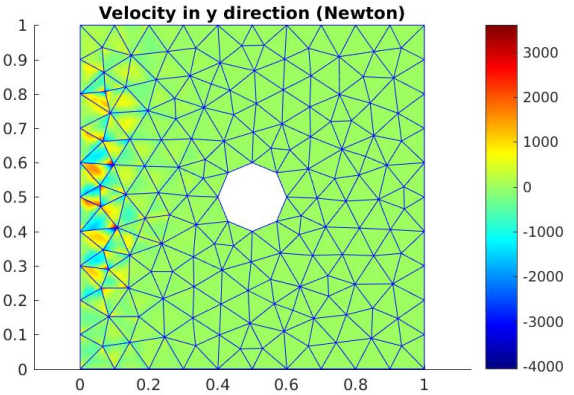


Figure 5.47: $y-$ velocity (Initial guess by bicgstab solver)

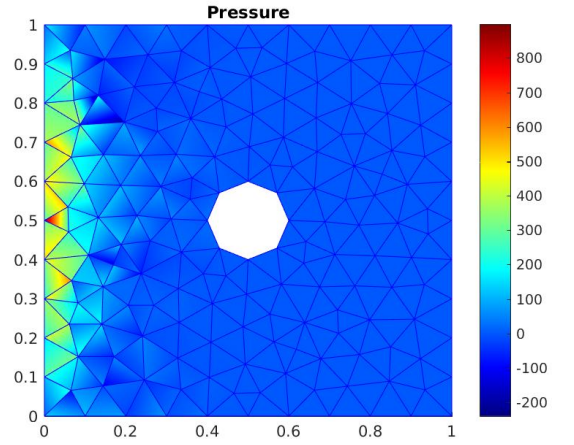


Figure 5.48: Pressure (Initial guess by bicgstab solver)

Figure 5.49

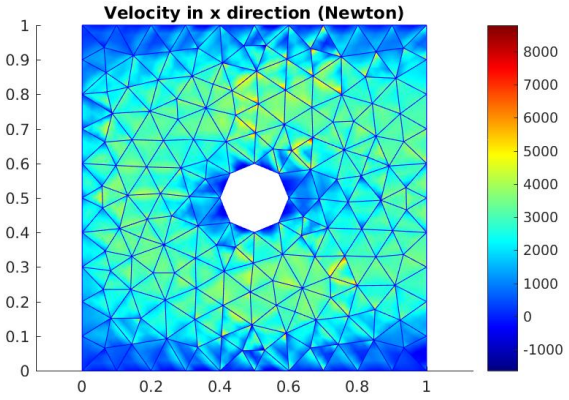


Figure 5.50: $x-$ velocity (Initial guess by minres solver)

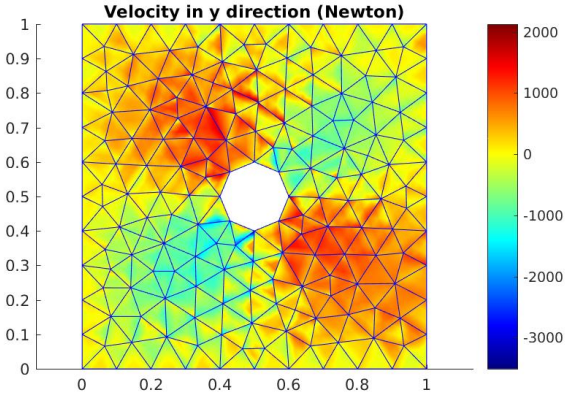


Figure 5.51: $y-$ velocity (Initial guess by minres solver)

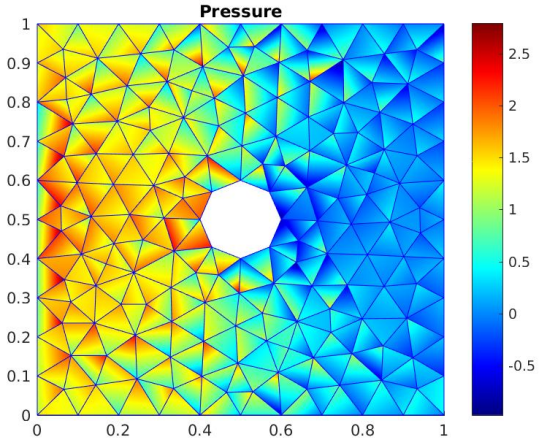


Figure 5.52: Pressure (Initial guess by minres solver)

Figure 5.53

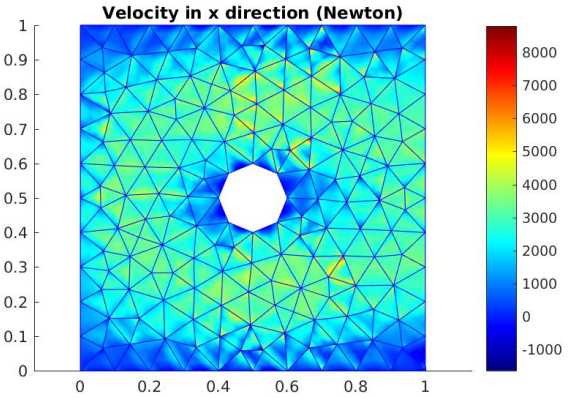


Figure 5.54: $x-$ velocity (Initial guess by Schur complement method)

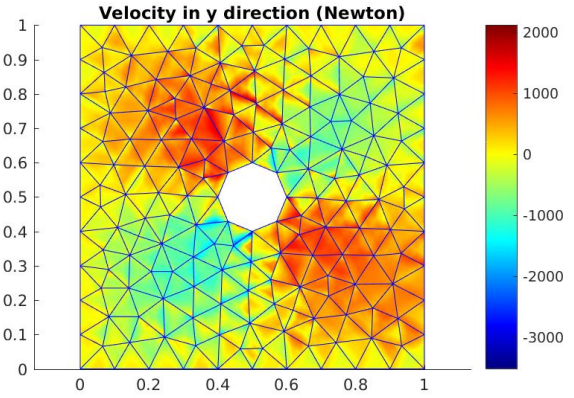


Figure 5.55: $y-$ velocity (Initial guess by Schur complement method)

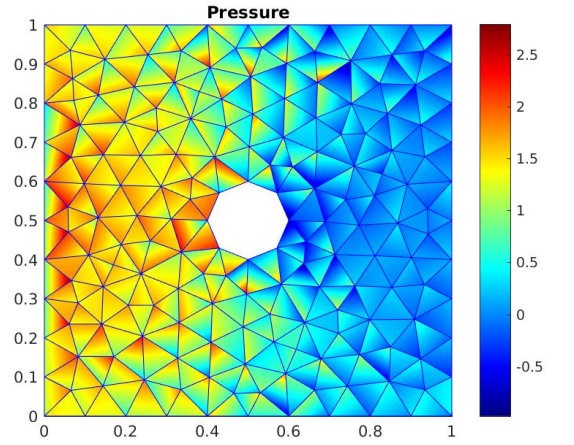


Figure 5.56: Pressure (Initial guess by Schur complement method)

Figure 5.57

Chapter 6

Summary, conclusion and outlook

6.1 Conclusions

After implementation of theoretical concepts and achieving experimental outcomes we arrive at following conclusion:
1. The Stiffness matrix of Stokes equation is symmetric and has positive definite part. Also the number of positive eigenvalues of stiffness matrix of Stokes equation is total number of velocity degrees of freedom and number of positive eigenvalues of stiffness matrix of Stokes equation is total number of pressure degrees of freedom.
2. The Schur complement method is very useful for Stokes flow due to efficient computation and good accuracy.
3. *minres* solver able to solve linear system of equation of stokes equation without unexpected termination of solver, i.e. the solver stops either after reaching maximum number iteration or specified tolerance level.
4. In contrast, *bicgstab* can stop without reaching specified tolerance level or maximum nu,ber of iterations.
5. The condition number of Stiffness matrix of Stokes equation increases with increase in penalty parameter. Therefore, the penalty parameter has to be small enough to limit the cndition number. However, the penalty parameter has to be large enough to maintain coercivity of bilinear form.
6. The stiffness matrix of Navier stokes equation is non symmetric. Therefore, the solvers such as *minres*, which are aplicable only for symmetric stiffness matrices are not useful. Therefore, solvers such as *bicgstab*, which are applicable for nonsymmetric stiffness matrices, are useful.
7. The newton methods requires solution of large system fo equation in each newton loop adding to heavy computational cost.
8. The solution of Stokes equation is dependent also on solver used.

9. The initial guess, in our case solution obtained from Stokes equation, is crucial for success of newton method.

6.2 Outlook

As future development, the Stokes equation and Navier Stokes equation can be parametrized for parameters such as fluid properties, geometry of domain or boundary conditions. This allows the approximation of the numerical solution with respect to parameter space. Affine transformation for Stokes equation and Empirical interpolation method for Navier Stokes equation can be used to evaluate parametrised solutions.
The solution of the parametrized form can be stored for reduced basis evaluations. Model order reduction with a method such as proper orthogonal decompositon 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.

Chapter 7

Mathematical preliminaries

We present now mathematical preliminaries from references relevant to the thesis.

7.1 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,

$$\mathcal{U} = \mathcal{L}\mathcal{L}^T \tag{7.1}$$

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 an Matrix in MATLAB. Since the back division operator (\) recognises the lower triangular structure of matrix, the division process is faster.
We now explain the algorithm for Cholesky decomposition.

$$\underset{\mathcal{U}}{\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}} = \underset{\mathcal{L}}{\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix}} \underset{\mathcal{L}^T}{\begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix}} \tag{7.2}$$

We see that,

$$a_{11} = l_{11}^2, \quad a_{22} = l_{21}^2 + l_{22}^2, \quad a_{33} = l_{31}^2 + l_{32}^2 + l_{33}^2 \tag{7.3}$$

and

$$a_{12} = a_{21} = l_{11}l_{21}, \quad a_{13} = a_{31} = l_{11}l_{31}, \quad a_{23} = a_{32} = l_{31}l_{21} + l_{32}l_{22} \tag{7.4}$$

We now see that for diagonal elements,

$$l_{kk} = \sqrt{a_{kk} - \sum_{k=1}^{j-1} l_{kj}^2} \tag{7.5}$$

and for elements below diagonal,

$$l_{ik} = \frac{1}{l_{kk}}(a_{ik} - \sum_{j=1}^{k-1} l_{ij}l_{kj}) \tag{7.6}$$

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.

7.2 Saddle point formulation

The saddle point problem has following form,

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ \mathcal{B}_2 & \mathcal{C} \end{pmatrix} \begin{pmatrix} \mathcal{X} \\ \mathcal{Y} \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{7.7}$$

$$\mathcal{A} \in \mathbb{R}^{n\times n}; \mathcal{B}_1, \mathcal{B}_2 \in \mathbb{R}^{m\times n}; \mathcal{C} \in \mathbb{R}^{m\times m} \tag{7.8}$$

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 semi definite
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 :

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ \mathcal{B}_2 & \mathcal{C} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \mathcal{B}_2\mathcal{A}^{-1} & I \end{pmatrix} \begin{pmatrix} \mathcal{A} & 0 \\ 0 & \mathcal{S} \end{pmatrix} \begin{pmatrix} I & \mathcal{A}^{-1}\mathcal{B}_1 \\ 0 & I \end{pmatrix} \tag{7.9}$$

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ \mathcal{B}_2 & \mathcal{C} \end{pmatrix} = \begin{pmatrix} \mathcal{A} & 0 \\ \mathcal{B}_2 & \mathcal{S} \end{pmatrix} \begin{pmatrix} I & \mathcal{A}^{-1}\mathcal{B}_1 \\ 0 & I \end{pmatrix} \tag{7.10}$$

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ \mathcal{B}_2 & \mathcal{C} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \mathcal{B}_2\mathcal{A}^{-1} & I \end{pmatrix} \begin{pmatrix} \mathcal{A} & \mathcal{B}_1 \\ 0 & \mathcal{S} \end{pmatrix} \tag{7.11}$$

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 negativesemi definite, $B_1 = B_2^T$ 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 [9]. We make some important observations related to Saddle point problems as follow :
1. In case $\mathcal{A}$ is symmetric positive definite the Schur complement is very useful as matrix $\mathcal{A}$ can be inverted efficiently with Cholesky decomposition (Section 7.1).
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 [8] for further understanding.

7.3 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

$$H^{k'}(\Omega) = \{f \in L^2(\Omega) | D^\alpha f \in L^2(\Omega), |\alpha| \leq k'\}, \tag{7.12}$$

where D^α is the partial derivative

$$D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_d^{\alpha_1} .....\partial x_d^{\alpha_d}} \tag{7.13}$$

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

$$(f, g)_{H^{k'}(\Omega)} = \sum_{\alpha \in \mathbb{N}^d, |\alpha| \leq k'} \int_\Omega (D^\alpha f)(D^\alpha g) \tag{7.14}$$

and the induced norm

$$||f||_{H^{k'}(\Omega)} = \sqrt{(f, f)_{H^{k'}(\Omega)}} = \sqrt{\sum_{\alpha \in \mathbb{N}^d, |\alpha| \leq k'} \int_\Omega |D^\alpha f|^2} \tag{7.15}$$

and the semi norm

$$|f|_{H^{k'}(\Omega)} = \sqrt{\sum_{\alpha \in \mathbb{N}^d, |\alpha| = k'} \int_\Omega |D^\alpha f|^2} \quad . \tag{7.16}$$

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

$$||f||^2_{1,h} = \sum_{\tau_k \in \mathcal{T}} ||\nabla f||^2_{L^2(\tau_k)} + \sum_{\tau_k \in \mathcal{T}} \kappa_E \nu |||[[u]]||^2_{L^2(\tau_k)} \tag{7.17}$$

and inner product

$$(f,g) = \sum_{\tau_k \in \mathcal{T}} (f,g)_{L^2(\tau_k)} + \sum_{\tau_k \in \mathcal{T}} \kappa_E \nu([u],[v])_{L^2(\tau_k)} \quad . \tag{7.18}$$

7.4 Basic definitions

We consider here vector space $\mathbb{V}$ over $\mathbb{R}$
1. For a set $\{w_1, ..., w_N\} \subset \mathbb{V}$ we denote by

$$span\{w_1, ..., w_n\} = \{v \in \mathbb{V} | v = \sum_{n=1}^{N} \alpha_n w_n, \alpha_n \in \mathbb{R}\} \tag{7.19}$$

the linear subspace spanned by the elements $w_1, ..., w_N$.
2. The space $\mathbb{V}$ is of finite dimension if there exists a maximal a set of linearly independent elements $v_1, ..., v_N$, otherwise $\mathbb{V}$ is of infinite dimension.
3. A norm $||.||_{\mathbb{V}}$ on $\mathbb{V}$ is a function $||.||_{\mathbb{V}} : \mathbb{V} \to \mathbb{R}$ 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. $||u + v|| \leq ||u||_{\mathbb{V}} + ||v||_{\mathbb{V}} \ u, v \in \mathbb{V}$.
4. The pair $(\mathbb{V}, ||.||_{\mathbb{V}})$ 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 $|.|_{\mathbb{V}} : \mathbb{V} \to \mathbb{R}$ 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 $||.||_1$ and $||.||_2$ are equivalent if there exists two constants $C_1, C_2 > 0$ such that

$$C_1 ||.||_1 \leq ||.||_2 \leq C_2 ||.||_1 \forall v \in V \tag{7.20}$$

7.5 Linear forms

Let $(\mathbb{V}, ||.||_{\mathbb{V}})$ be a normed space. Then, we define the following notions.
1. A function $F : \mathbb{V} \to \mathbb{R}$ is said to be linear

$$F(u + v) = F(u) + F(v) \forall u, v \in \mathbb{V} \tag{7.21}$$

$$F(\alpha u) = \alpha F(u) \forall \alpha \in \mathbb{R}, u \in \mathbb{V} \tag{7.22}$$

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

$$|F(v)| \leq \gamma ||v||_{\mathbb{V}} \forall v \in \mathbb{V} \tag{7.23}$$

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

$$||u - v||_{\mathbb{V}} \leq \delta_\epsilon \Rightarrow |F(u) - F(v)| < \epsilon \tag{7.24}$$

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

7.6 Bilinear forms

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

$$a : \mathbb{V} \times \mathbb{W} \Rightarrow \mathbb{R} \tag{7.25}$$

$$(u, v) \mapsto a(u, v) \tag{7.26}$$

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,

$$|a(u, v)| \leq \gamma ||u||_{\mathbb{V}} ||v||_{\mathbb{W}} \quad \forall \quad u, v \in \mathbb{V} \tag{7.27}$$

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

$$a(v, v) \geq \alpha ||v||^2_{\mathbb{V}} \quad \forall \quad v \in \mathbb{V} \tag{7.28}$$

7.6.1 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

$$a(v, v) > \kappa_e ||v||^2 \quad \forall \quad v \in \mathbb{V} \tag{7.29}$$

7.6.2 Continuity of bilinear form

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

$$a(u, v) \leq \gamma ||u|| ||v|| \quad \forall \quad u, v \in \mathbb{V} \tag{7.30}$$

7.7 Important inequalities

7.7.1 Trace theorem

We now refer to trace inequalities presented by Rivière B.[5]. 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, $\forall \phi \in P^D(\tau)$, $\forall \Gamma \subset \partial\Omega$

$$||\phi||_{L^2(\Gamma)} \leq \hat{C}_t l^{\frac{1}{2}} Ar^{\frac{-1}{2}} ||\phi||_{L^2(\tau)} \tag{7.31}$$

$$||\phi||_{L^2(\Gamma)} \leq C_t |h_\tau|^{\frac{-1}{2}} ||\phi||_{L^2(\tau)} \tag{7.32}$$

$$||\nabla\phi \cdot n||_{L^2(\Gamma)} \leq \hat{C}_t |l|^{\frac{1}{2}} |Ar|^{\frac{-1}{2}} ||\nabla\phi||_{L^2(\tau)} \tag{7.33}$$

$$||\nabla\phi \cdot n||_{L^2(\Gamma)} \leq C_t |h_\tau|^{\frac{-1}{2}} ||\nabla\phi||_{L^2(\tau)} \tag{7.34}$$

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 [12].

7.7.2 Cauchy-Schwarz inequality

$$\forall f, g \in L^2(\Omega),\ |(f,g)_\Omega| \leq ||f||_{L^2(\Omega)} ||g||_{L^2(\Omega)}$$

7.7.3 Young's inequality

$$\forall \epsilon > 0,\ \forall a, b \in \mathbb{R},\ ab \leq \frac{\epsilon}{2} a^2 + \frac{1}{2\epsilon} b^2$$

Chapter 8

References

8.1 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

8.2 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] Haasdonk B. *Reduced Basis Methods for Parametrized PDEs – A Tutorial Introduction for Stationary and Instationary Problems. Chapter to appear in in P. Benner, A. Cohen, M. Ohlberger and K. Willcox (eds.): "Model Reduction and Approximation: Theory and Algorithms",*. SIAM, Philadelphia, 2016.

[4] Haasdonk B. Lecture notes : Reduzierte-Basis-Methoden. *University of Stuttgart*, Summer term 2010.

[5] Riviere B. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation.* Frontiers in Applied Mathematics. Cambridge University Press, 2008.

[6] Fritzen F. Lecture notes : Introduction to model order reduction of mechanical systems. *University of Stuttgart*, winter term 2016-2017.

[7] White F.M. *Fluid mechanics.* McGraw-Hill international editions. Mechanical engineering series. McGraw-Hill Ryerson, Limited, 1986.

[8] 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.

[9] Benzi M., Golub G. H., and Liesen J. Numerical solution of saddle point problems. *Acta Numerica*, 14:1137, 2005.

[10] Kundu P. K. and Cohen I. M. *Fluid Mechanics.* Academic Press, 2002.

[11] 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.

[12] 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.