USER MANUAL

PDE FINITE ELEMENT SOLVER

April 26, 2018

Hasan Shetabivash

Contents

Chapter 1

Possoin Solver

1.1 Introduction

The main purpose of this project is to develop a code to solve poisson equation with dirichlet boundary conditions. To this end a software is developed with C# language using object-oriented concepts to simulate and solve poisson equation in a square domain. A view of this software is shown in Fig. 3.1.

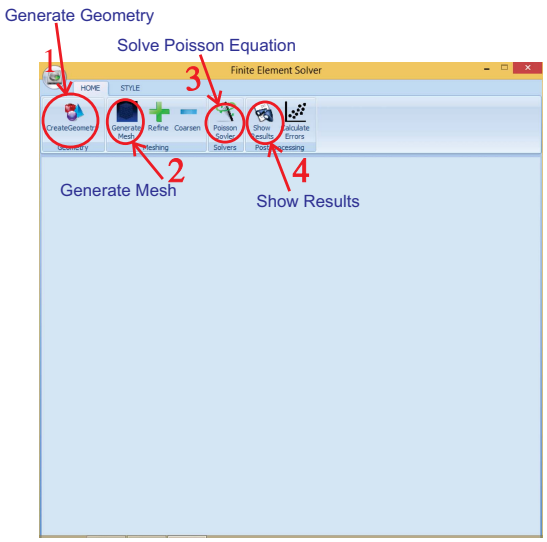


Figure 1.1: General view of software developed for solving poisson equation

As it is shown in Fig. 3.1, the software is capable of generation domain and mesh. Also it is possible to refine the mesh in order to obtain better results. After mesh gener-

ation poisson solver will solve the problem and results will be shown for each node. Finally, errors of the calculation can be plotted to get an insight about convergence of the solver.

1.2 Problem

In this project we're solving Poisson equation

$$-\nabla^2 u = f \quad \text{in} \quad \Omega \tag{1.1}$$

$$u = 0 \quad \text{on} \quad \partial\Omega \tag{1.2}$$

Using Reighly-Ritz, weak formulation of the problem is to find:

$$u = \underset{w \in X}{\operatorname{argmin}} \quad \underbrace{\frac{1}{2} a(w,w) - l(w)}_{J(w)} \tag{1.3}$$

or find $u \in X$ such that:

$$a(u,v) = l(v) \quad , \forall v \in X \tag{1.4}$$

where

$$X = \{v \in H^1(\Omega) | v|_\Gamma = 0\} \equiv H_0^1(\Omega) \tag{1.5}$$

$$a(w,v) = \int_\Omega \nabla w \cdot \nabla v dA \tag{1.6}$$

1.3 Geometry

The geometry of this project is a unit square which is specified with four nodes located on the corners of the geometry. In order to generate the geometry there is a command button (Fig. 3.1) by which the domain of the solution would be created.

1.4 Mesh generation

In order to obtain accurate numerical results is of the utmost importance to choose an appropriate mesh type. Finite element method has several advantages over other methods and mesh generation is one of the most important of them. Finite element is able to solve equations using triangular elements which is a crucial choice when dealing with complex geometries.

Element used in this project is a subparametric quadratic triangle with six points (Fig. 3.2). Three of these points are considered main points located at vertices and other three points are located in the middle of the each edge.

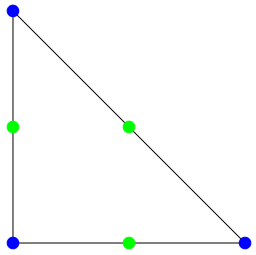


Figure 1.2: Subparametric quadratic triangle with six points

Clicking on the "Generate Mesh" button the software will regularly discretize the domain into P2 elements. To obtain better and more accurate results the mesh should be fine enough, so, after mesh generation by clicking on the "Refine" button the mesh will be refined. In Fig. 3.3 a sample of generated and refined meshes is shown. After mesh generation there is a possibility to export the mesh to different formats which Fig. 1.4 illustrated an exported mesh from the software. As it is shown in Table. 2.1, each element encompasses six points and each point has two different numbering; local and global indicators.

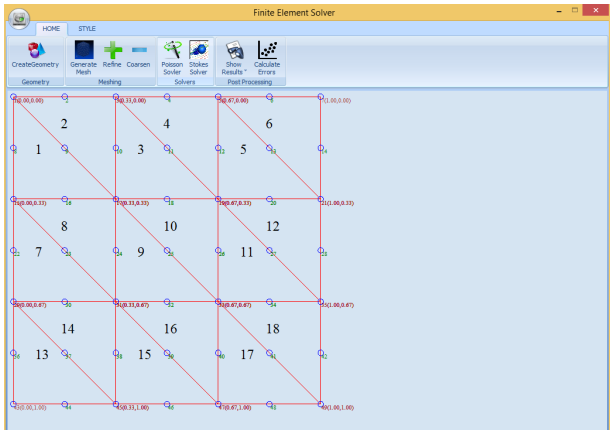


Figure 1.3: A sample generated mesh using the developed software

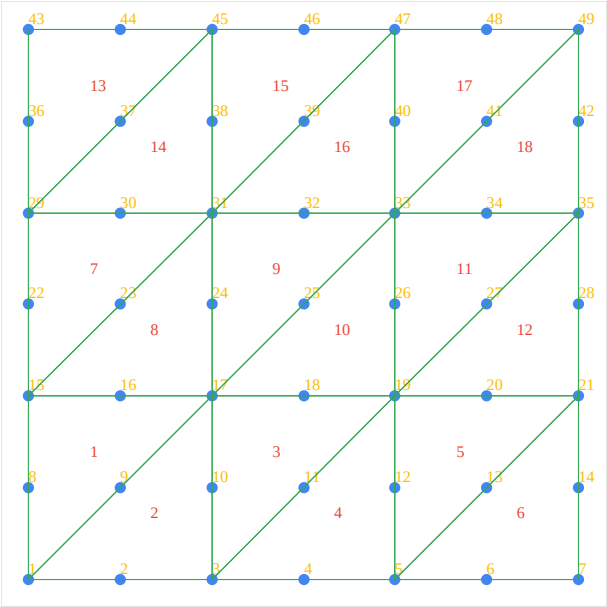


Figure 1.4: A sample exported mesh using the developed software

Table 1.1: Elements with associated nodes for a sample generated gird

Elements	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6
1	1	17	15	9	16	8
2	1	3	17	2	10	9
3	3	19	17	11	18	10
4	3	5	19	4	12	11
5	5	21	19	13	20	12
6	5	7	21	6	14	13
7	15	31	29	23	30	22
8	15	17	31	16	24	23
9	17	33	31	25	32	24
10	17	19	33	18	26	25
11	19	35	33	27	34	26
12	19	21	35	20	28	27
13	29	45	43	37	44	36
14	29	31	45	30	38	37
15	31	47	45	39	46	38
16	31	33	47	32	40	39
17	33	49	47	41	48	40
18	33	35	49	34	42	41

1.5 Numerical Procedure

1.5.1 Affine mapping

In order to facilitate the numerical procedure, each element is mapped into a reference element using a mapping function (Fig. 3.4).

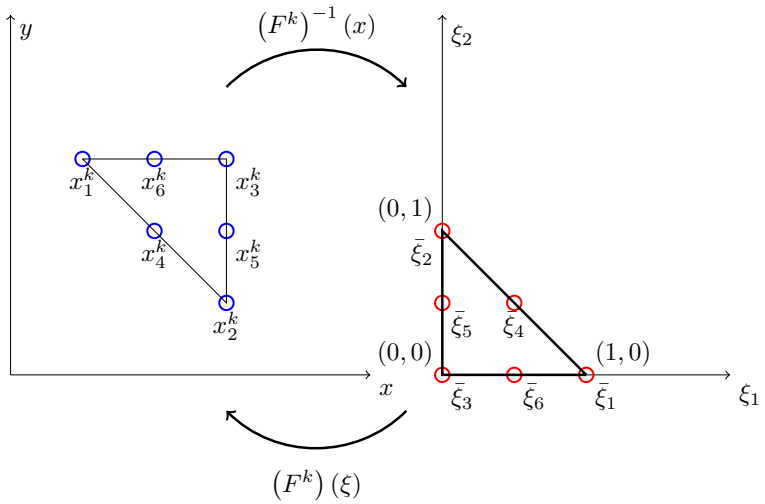


Figure 1.5: Affine Mapping

The following relation can be used to transform every node in original space to reference space:

$$\begin{pmatrix}\xi_1\\ \xi_2\end{pmatrix} = \begin{pmatrix}C_1^k\\ C_2^k\end{pmatrix} + \begin{pmatrix}D_{11} & D_{12}\\ D_{21} & D_{22}\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} \tag{1.7}$$

where

$$C_i^k = \frac{1}{2area^k}\{\hat{x}_{[i+1]}^k \hat{y}_{[i+2]}^k - \hat{x}_{[i+2]}^k \hat{y}_{[i+1]}^k\} \tag{1.8}$$

$$D_{i1}^k = \frac{1}{2area^k}\{\hat{y}_{[i+1]}^k - \hat{y}_{[i+2]}^k\} \tag{1.9}$$

$$D_{i2}^k = \frac{1}{2area^k}\{\hat{x}_{[i+2]}^k - \hat{x}_{[i+1]}^k\} \tag{1.10}$$

Any function defined on a triangle can be approximated by the quadratic function:

$$u(x,y) = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2 \tag{1.11}$$

and in the transformed system we obtain

$$u(\xi_1,\xi_2) = h_1 + h_2\xi_1 + h_3\xi_2 + h_4\xi_1^2 + h_5\xi_1\xi_2 + h_6\xi_2^2 \tag{1.12}$$

where

$$h_1 = \xi_1(-1+2\xi_1) \tag{1.13}$$
$$h_2 = \xi_2(-1+2\xi_2) \tag{1.14}$$
$$h_3 = (1-\xi_1-\xi_2)(1-2\xi_2-2\xi_2) \tag{1.15}$$
$$h_4 = 4\xi_1\xi_2 \tag{1.16}$$
$$h_5 = 4\xi_2(1-\xi_1-\xi_2) \tag{1.17}$$
$$h_6 = 4\xi_1(1-\xi_1-\xi_2) \tag{1.18}$$

Reference elemental besis has the following property:

$$h_i(\xi_i) = \delta_{ij} \tag{1.19}$$

For example considering Fig. 3.4, for $\bar{\xi_1} = (1,0)$, $h_1(\bar{\xi_1}) = 1$, $h_1(\bar{\xi_2}) = 0$,$h_1(\bar{\xi_3}) = 0$, $h_1(\bar{\xi_4}) = 0$, ... Also this is true for other points, $h_2(\bar{\xi_1}) = 0$, $h_2(\bar{\xi_2}) = 1$,$h_2(\bar{\xi_3}) = 0$, $h_2(\bar{\xi_4}) = 0$, ...

1.5.2 Stiffness Matrix

In order to solve the problem we need to calculate stiffness matrix for each element. Stiffness Matrix ($\hat{A}^k_{\alpha\beta}$) for each element which is a 6×6 elemental matrix can be constructed using the following equation:

$$\hat{A}_{\alpha\beta} = \int_{\Omega^k} \frac{\partial h_\alpha}{\partial x_1}\frac{\partial h_\beta}{\partial x_1} + \frac{\partial h_\alpha}{\partial x_2}\frac{\partial h_\beta}{\partial x_2} dxdy \tag{1.20}$$

It will be convenient to write $\hat{A}^k_{\alpha\beta}$ as follows:

$$\hat{A}^k_{\alpha\beta} = 2area^k \left(\sum_{m=1}^{2} \sum_{n=1}^{2} \sum_{n'=1}^{2} D^k_{nm} D^k_{n'm} G_{\alpha\beta nn'} \right) \tag{1.21}$$

where

$$G^k_{\alpha\beta} = \int_0^1 \int_0^{1-\xi_2} \frac{\partial h_\alpha}{\partial \xi_n} \frac{\partial h_\beta}{\partial \xi_{n'}} d\xi_1 d\xi_2 \tag{1.22}$$

1.5.3 Mass Matrix

Asides from stiffness matrix, mass matrix should also be calculated from the following equation for each element:

$$\hat{M}^k_{\alpha\beta} = \int_{\Omega^k} h_\alpha h_\beta dx dy \tag{1.23}$$

It will be convenient to write $\hat{M}^k_{\alpha\beta}$ as follows:

$$\hat{M}^k_{\alpha\beta} = 2area^k \int_{\Omega^k} h_\alpha h_\beta d\xi_1 d\xi_2 \tag{1.24}$$

There is a Maple code to calculate $G^k_{\alpha\beta}$:

Listing 1.1: A sample Maple code for calculation $G_{\alpha\beta nb'}$

```
z := array(1 .. 3);
h := array(1 .. 6);
g := array(1 .. 6, 1 .. 6, 1 .. 2, 1 .. 2);
h[1] := z[1]*(-1+2*z[1]);
h[2] := z[2]*(-1+2*z[2]);
h[3] := z[3]*(-1+2*z[3]);
h[4] := 4*z[1]*z[2];
h[5] := 4*z[2]*z[3];
h[6] := 4*z[1]*z[3];
for alpha from 1 to 6 do:
for beta from 1 to 6 do:
```

```
for n1 from 1 to 2 do:
for n2 from 1 to 2 do:
z[3] := 1-z[1]-z[2];
eq1 := diff(h[alpha], z[n1]);
eq2 := diff(h[beta], z[n2]);
eq3 := eq1*eq2;
eq4 := evalf(int(eq3, z[1] = 0 .. 1-z[2]));
g[alpha, beta, n1, n2] := int(eq4, z[2] = 0 .. 1);
unassign('z'); z := array(1 .. 3)
end do end do end do end do;
 eval(g)
```

The final step is to assemble the system forcing vector. The following algorithm would be effective for constructing global stiffness and mass matrices.

forall the *Elements(k)* **do**
  for $\alpha = 1,\ldots,6$ **do**
    $i = g_\alpha^k$
    for $\beta = 1,\ldots,6$ **do**
      $j = g_\beta^k$
      $\tilde{A}_{ij} = \tilde{A}_{ij} + \tilde{A}_{\alpha\beta}^k$
      $\tilde{M}_{ij} = \tilde{M}_{ij} + \tilde{M}_{\alpha\beta}^k$
    end
  end
end

Algorithm 1: Constructing global stiffness and mass matrices

Afte assembling mass matrix, right hand side of the equation can be calculated by multiplying mass matrix and force vector (obtained from function f in Eq. 1.3). So at this step that global stiffness matrix and right hand side of the equation are constructed, boundary conditions should be applied and final system of equations should be solved with an appropriate algorithm.

1.5.4 Boundary Conditions

Applying boundary condition at the last step can be done with different methods. One of the most useful ways of applying boundary conditions is to eliminate boundary nodes from final system of equations and solve the system just for internal nodes.

1.5.5 Solver

After applying boundary conditions the system of equations is obtained:

$$\mathbf{Ax} = \mathbf{b} \tag{1.25}$$

LU decomposition is used to solve the obtain linear system of equations. In order to validate the accuracy of the solver, a sample linear system of equations is solved as following:

$$\begin{bmatrix} 3 & 2 & -1 \\ 2 & -2 & 4 \\ -1 & \frac{1}{2} & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix} \tag{1.26}$$

The resulting **x** is [1,âĹŠ2,âĹŠ2], hence the solution $x = 1, y =$ âĹŠ2, $z =$ âĹŠ2. So as demonstrated the solver is capable of solving system of equations accurately and will be used for solving current system of equations.

1.6 Validation

Validation of the code will be performed by comparing the result with an exact solution of the problem and three different errors would be calculated to check the convergence of the numerical solution. If we assume $u^* = \sin(\pi x)\sin(\pi y)$ as an exact solution of poisson equation which is plotted in Fig. 3.5 by substituting it into Eq. 1.3 we obtain:

$$f = 2\pi^2 \sin(\pi x)\sin(\pi y) \tag{1.27}$$

Now by solving the poisson equation using the developed finite element code we will get the results shown in Fig. 1.7. The contours of solution are illustrated in the Fig. 3.6.

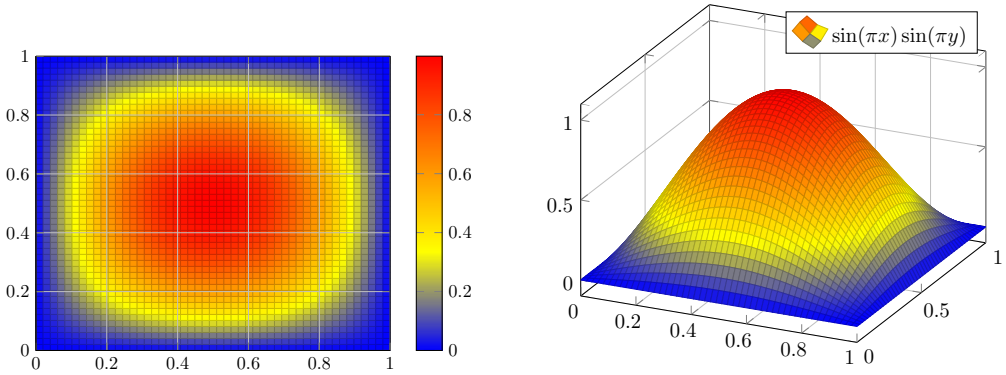


Figure 1.6: Exact solution of poisson equation

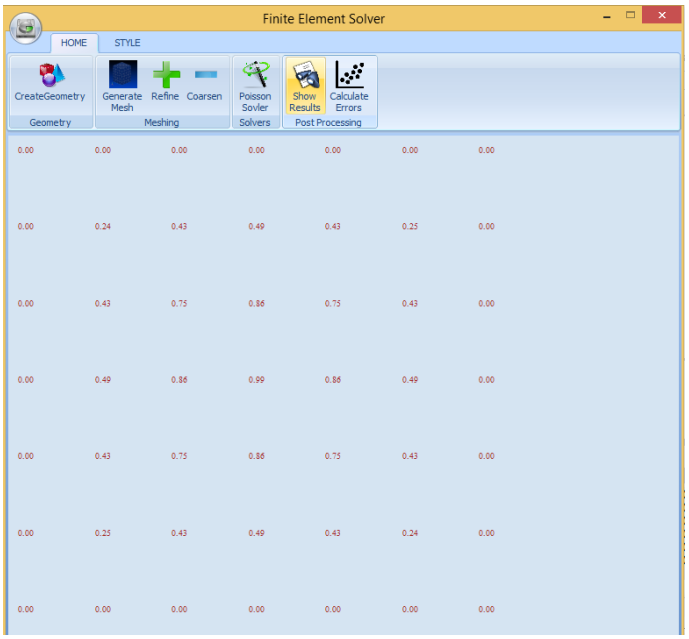


Figure 1.7: Solution of poisson equation using developed software

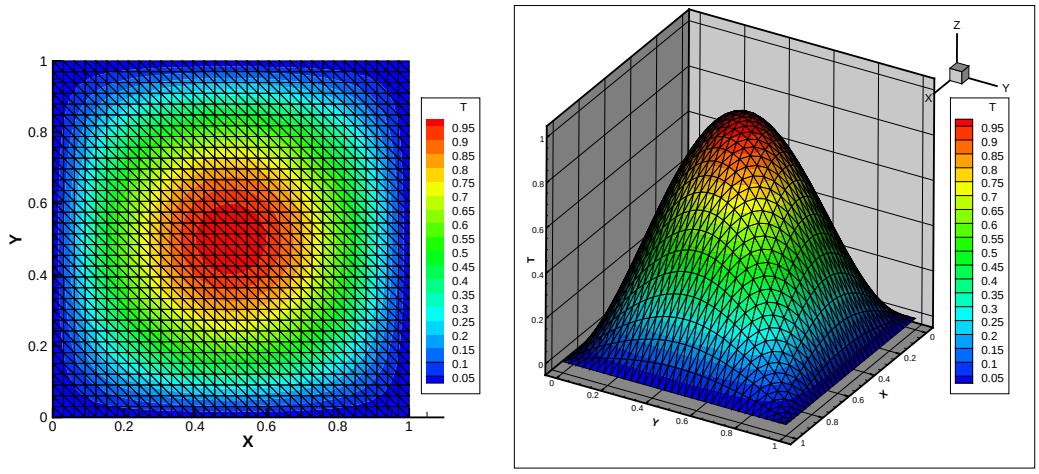


Figure 1.8: Contours of the numerical solution of poisson equation

Three different types of errors can be defined to check the convergence of the solution.

$$E_1 = |u^*(0.5,0.5) - u_h(0.5,0.5)| \tag{1.28}$$

$$E_2 = \sqrt{(u^* - u_h, u^* - u_h)} \tag{1.29}$$

$$E_3 = \sqrt{a(u^* - u_h, u^* - u_h)} \tag{1.30}$$

In order to check the convergence, three error type is plotted in Fig. 1.9. As it is shown in Fig. 1.9 the errors are reasonable and the problem is converged.

One of the most important errors encountered in numerical simulations corresponds to spatial discretization, so, more refined discretization would lead to more accurate numerical results. Besides, decreasing the difference between exact and numerical solution of the problem will be interpreted as getting more accurate results. Having considered aforementioned concepts it would be clear that by increasing the number of elements (decreasing h) errors should be decreased accordingly. In order to check the convergence of the problem and its relation to refinement of grids it would be beneficial to compare three different errors which are plotted in Fig. 1.9 together. Comparing these results reveals that difference between numerical simulation and exact simulation decreases with more refined meshes which is expected in numerical simulations.

Three different plotted errors have different behaviors. E_2 and E_3 are decreasing

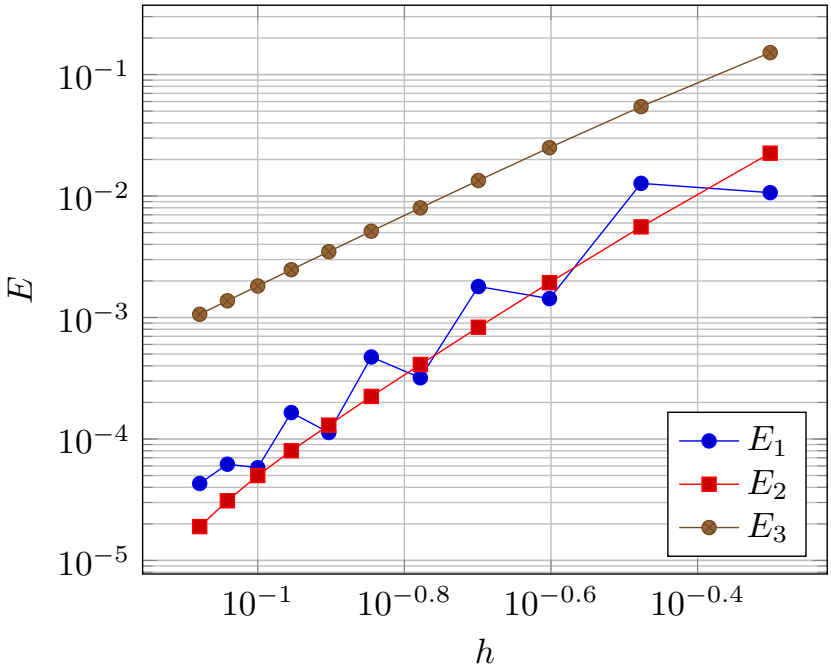


Figure 1.9: Plots of three different errors for obtained from solution of poisson equation

regularly by refining the mesh but E_1 which is defined the difference between exact and numerical value at the center of the domain exhibits another behavior as can be seen in Fig. 1.9. Following the plotted errors, it can be realized that E_1 is not decreasing regularly and there is a step like pattern which at the first glance may seem strange or even unreasonable. However, by considering our element structure and the fact that our elements have two different types of nodes (main and middle) center of the domain would coincide with on of these node types. Considering different meshes with different horizontal and vertical node numbers ,N_1, it can be realized that for even node numbers center of the domain would coincide with the main node of the element and on the other hand for odd node numbers center node would coincide with the middle node of the element. Having considered all these facts the step-like irregularity in decreasing E_1 can be justified by the fact that center node of the mesh is associated with main node of the element when number of N_1 is even, and middle node of the element when the number of N_1 is odd. As expected the value of error is lower when center of the domain domain is located in the main point of the element.

Some more useful information could be extracted from Fig. 1.9 by comparing

three type of plotted errors. Before any comparison it should be kept in mind that both value of the errors and decreasing rate of them with refining the mesh are important and should be considered when comparing them. So, having considered the data provided in Fig. 1.9, it would be realized that E_1 and E_2 are at the same order and less than E_3. However the rate of decreasing of E_1 and E_2 are nearly identical and greater than E_3.

Chapter 2

Stokes Solver

2.1 Introduction

The main purpose of this project is to develop a code to solve sotkes equation with dirichlet boundary conditions.To this end a stokes equation solver is added to the developed software(Fig. 3.1)

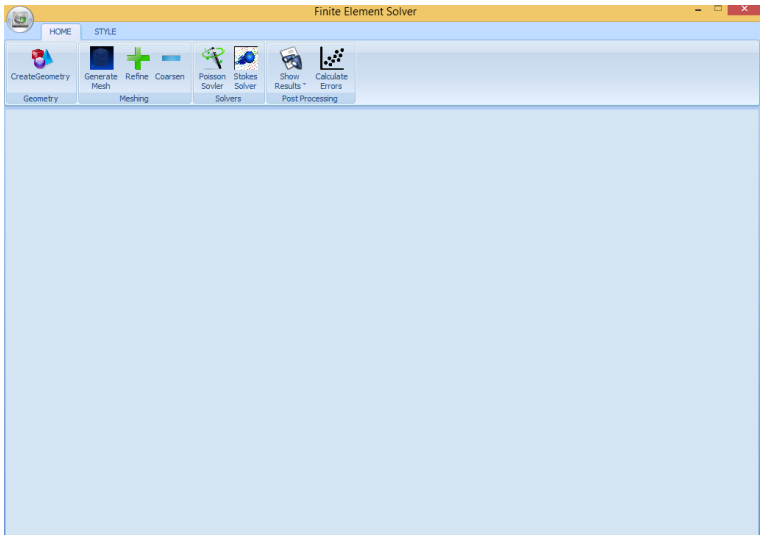


Figure 2.1: General view of software developed for solving stokes equation

As it is shown in Fig. 3.1, the software is capable of generation domain and mesh. Also it is possible to refine the mesh in order to obtain better results. After mesh generation stokes solver will solve the problem and results will be shown for each node.

Finally, errors of the calculation can be plotted to get an insight about convergence of the solver.

2.2 Problem

In this project we're solving stokes equation

$$-\nabla^2 u + \nabla p = f \quad \text{in} \quad \Omega \tag{2.1}$$
$$-\nabla u = 0 \quad \text{in} \quad \Omega \tag{2.2}$$
$$u = 1\hat{x} \quad \text{on} \quad \Omega_{top} \tag{2.3}$$
$$u = 0\hat{x} \quad \text{on} \quad \Omega \backslash \Omega_{top} \tag{2.4}$$
$$\tag{2.5}$$

The following system of equations could be obtained from weak formulation of the problem:

$$\begin{pmatrix} A & 0 & -D_1^T \\ 0 & A & -D_2^T \\ -D_1 & -D_2 & 0 \end{pmatrix} \begin{pmatrix} u_h \\ v_h \\ p_h \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ 0 \end{pmatrix} \tag{2.6}$$

where A is globals stiffness matrix.

2.3 Geometry

The geometry of this project is a unit square which is specified with four nodes located on the corners of the geometry. In order to generate the geometry there is a command button (Fig. 3.1) by which the domain of the solution would be created.

2.4 Mesh generation

In order to obtain accurate numerical results it is of the utmost importance to choose an appropriate mesh type. Finite element method has several advantages over other methods and mesh generation is one of the most important of them. Finite element

is able to solve equations using triangular elements which is a crucial choice when dealing with complex geometries.

Not every finite element can be used to solve stokes equation. One of the choices to solve the stokes equation is Taylor and Hood elements which is a combination of P2 and P1 triangular elements as shown in Fig. 3.2

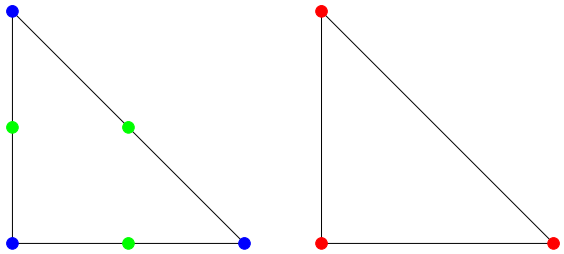


Figure 2.2: P2 and P1 Triangular elements used in Taylor and Hood elements

Clicking on the "Generate Mesh" button the software will regularly discretize the domain into P2 elements. To obtain better and more accurate results the mesh should be fine enough, so, after mesh generation by clicking on the "Refine" button the mesh will be refined. In Fig. 3.3 a sample of generated and refined meshes is shown. After mesh generation there is a possibility to export the mesh to different formats which Fig. 3.3 illustrated an exported mesh from the software. As it is shown in Table. 2.1, each element encompasses six points and each point has two different numbering; local and global indicators.

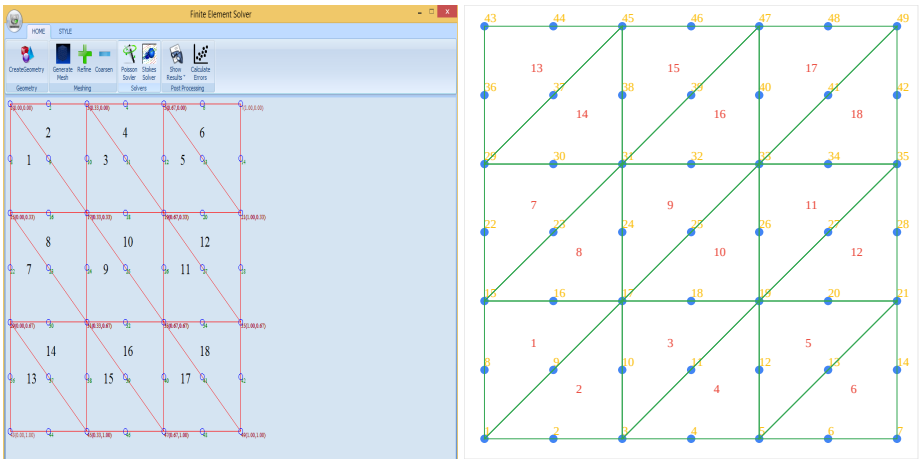


Figure 2.3: A sample generated mesh using the developed software

Table 2.1: Elements with associated nodes for a sample generated gird

Elements	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6
1	1	17	15	9	16	8
2	1	3	17	2	10	9
3	3	19	17	11	18	10
4	3	5	19	4	12	11
5	5	21	19	13	20	12
6	5	7	21	6	14	13
7	15	31	29	23	30	22
8	15	17	31	16	24	23
9	17	33	31	25	32	24
10	17	19	33	18	26	25
11	19	35	33	27	34	26
12	19	21	35	20	28	27
13	29	45	43	37	44	36
14	29	31	45	30	38	37
15	31	47	45	39	46	38
16	31	33	47	32	40	39
17	33	49	47	41	48	40
18	33	35	49	34	42	41

2.5 Numerical Procedure

2.5.1 Affine mapping

In order to facilitate the numerical procedure, each element is mapped into a reference element using a mapping function (Fig. 3.4).

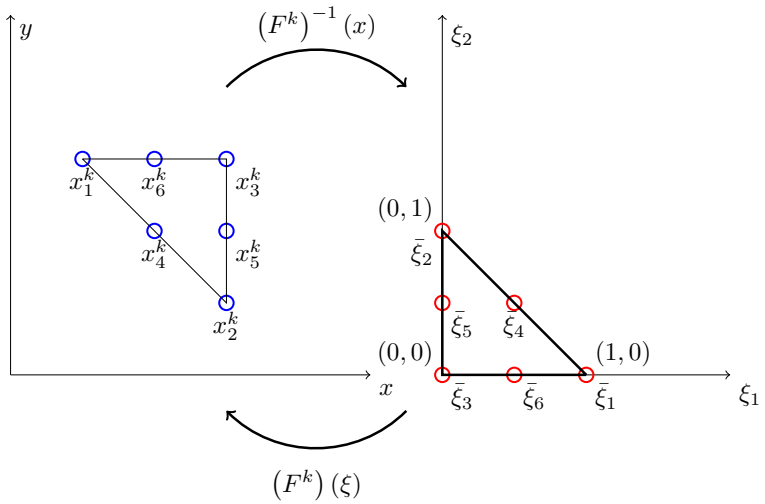


Figure 2.4: Affine Mapping

The following relation can be used to transform every node in original space to reference space:

$$\begin{pmatrix}\xi_1\\ \xi_2\end{pmatrix} = \begin{pmatrix}C_1^k\\ C_2^k\end{pmatrix} + \begin{pmatrix}D_{11} & D_{12}\\ D_{21} & D_{22}\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} \tag{2.7}$$

where

$$C_i^k = \frac{1}{2area^k}\{\hat{x}_{[i+1]}^k \hat{y}_{[i+2]}^k - \hat{x}_{[i+2]}^k \hat{y}_{[i+1]}^k\} \tag{2.8}$$

$$D_{i1}^k = \frac{1}{2area^k}\{\hat{y}_{[i+1]}^k - \hat{y}_{[i+2]}^k\} \tag{2.9}$$

$$D_{i2}^k = \frac{1}{2area^k}\{\hat{x}_{[i+2]}^k - \hat{x}_{[i+1]}^k\} \tag{2.10}$$

2.5.2 Basis function

2.5.2.1 P1 Elements

Any function defined on a P1 triangle can be approximated by the linear function:

$$u(x,y) = a_1 + a_2 x + a_3 y \tag{2.11}$$

and in the transformed system we obtain

$$u(\xi_1,\xi_2) = h_1 + h_2\xi_1 + h_3\xi_2 \tag{2.12}$$

where

$$h_1 = \xi_1 \tag{2.13}$$
$$h_2 = \xi_2 \tag{2.14}$$
$$h_3 = (1-\xi_1-\xi_2) \tag{2.15}$$
$$\tag{2.16}$$

Reference elemental basis has the following property:

$$h_i(\xi_i) = \delta_{ij} \tag{2.17}$$

For example considering Fig. 3.4, for $\bar{\xi_1} = (1,0)$, $h_1(\bar{\xi}_1) = 1$, $h_1(\bar{\xi}_2) = 0$,$h_1(\bar{\xi}_3) = 0$. Also this is true for other points, $h_2(\bar{\xi}_1) = 0$, $h_2(\bar{\xi}_2) = 1$,$h_2(\bar{\xi}_3) = 0$.

2.5.2.2 P2 Elements

Any function defined on a triangle can be approximated by the quadratic function:

$$u(x,y) = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2 \tag{2.18}$$

and in the transformed system we obtain

$$u(\xi_1,\xi_2) = h_1 + h_2\xi_1 + h_3\xi_2 + h_4\xi_1^2 + h_5\xi_1\xi_2 + h_6\xi_2^2 \tag{2.19}$$

where

$$h_1 = \xi_1(-1+2\xi_1) \tag{2.20}$$
$$h_2 = \xi_2(-1+2\xi_2) \tag{2.21}$$
$$h_3 = (1-\xi_1-\xi_2)(1-2\xi_2-2\xi_2) \tag{2.22}$$
$$h_4 = 4\xi_1\xi_2 \tag{2.23}$$
$$h_5 = 4\xi_2(1-\xi_1-\xi_2) \tag{2.24}$$
$$h_6 = 4\xi_1(1-\xi_1-\xi_2) \tag{2.25}$$

Reference elemental basis has the following property:

$$h_i(\xi_i) = \delta_{ij} \tag{2.26}$$

For example considering Fig. 3.4, for $\bar{\xi_1} = (1,0)$, $h_1(\bar{\xi_1}) = 1$, $h_1(\bar{\xi_2}) = 0$,$h_1(\bar{\xi_3}) = 0$, $h_1(\bar{\xi_4}) = 0$, ... Also this is true for other points, $h_2(\bar{\xi_1}) = 0$, $h_2(\bar{\xi_2}) = 1$,$h_2(\bar{\xi_3}) = 0$, $h_2(\bar{\xi_4}) = 0$, ...

2.5.3 Stiffness Matrix

In order to solve the problem we need to calculate stiffness matrix for each element. Stiffness Matrix ($\hat{A}^k_{\alpha\beta}$) for each element which is a 6×6 elemental matrix can be constructed using the following equation:

$$\hat{A}_{\alpha\beta} = \int_{\Omega^k} \frac{\partial h^v_\alpha}{\partial x_1}\frac{\partial h^v_\beta}{\partial x_1} + \frac{\partial h^v_\alpha}{\partial x_2}\frac{\partial h^v_\beta}{\partial x_2} dxdy \tag{2.27}$$

It will be convenient to write $\hat{A}^k_{\alpha\beta}$ as follows:

$$\hat{A}^k_{\alpha\beta} = 2area^k \left(\sum_{m=1}^{2}\sum_{n=1}^{2}\sum_{n'=1}^{2} D^k_{nm} D^k_{n'm} G_{\alpha\beta nn'} \right) \tag{2.28}$$

where

$$G^k_{\alpha\beta} = \int_0^1 \int_0^{1-\xi_2} \frac{\partial h^\nu_\alpha}{\partial \xi_n} \frac{\partial h^\nu_\beta}{\partial \xi_{n'}} d\xi_1 d\xi_2 \tag{2.29}$$

There is a Maple code to calculate $G^k_{\alpha\beta}$:

Listing 2.1: A sample Maple code for calculation $G_{\alpha\beta nb'}$

```
z := array(1 .. 3);
hv := array(1 .. 6);
g := array(1 .. 6, 1 .. 6, 1 .. 2, 1 .. 2);
hv[1] := z[1]*(-1+2*z[1]);
hv[2] := z[2]*(-1+2*z[2]);
hv[3] := z[3]*(-1+2*z[3]);
hv[4] := 4*z[1]*z[2];
hv[5] := 4*z[2]*z[3];
hv[6] := 4*z[1]*z[3];
for alpha from 1 to 6 do:
for beta from 1 to 6 do:
for n1 from 1 to 2 do:
for n2 from 1 to 2 do:
z[3] := 1-z[1]-z[2];
eq1 := diff(hv[alpha], z[n1]);
eq2 := diff(hv[beta], z[n2]);
eq3 := eq1*eq2;
eq4 := evalf(int(eq3, z[1] = 0 .. 1-z[2]));
g[alpha, beta, n1, n2] := int(eq4, z[2] = 0 .. 1);
unassign('z'); z := array(1 .. 3)
end do end do end do end do;
 eval(g)
```

2.5.4 Mass Matrix

Asides from stiffness matrix, mass matrix should also be calculated from the following equation for each element:

$$\hat{M}^k_{\alpha\beta} = \int_{\Omega^k} h^v_\alpha h^v_\beta dxdy \tag{2.30}$$

It will be convenient to write $\hat{M}^k_{\alpha\beta}$ as follows:

$$\hat{M}^k_{\alpha\beta} = 2area^k \int_{\Omega^k} h^v_\alpha h^v_\beta d\xi_1 d\xi_2 \tag{2.31}$$

There is a Maple code to calculate $M^k_{\alpha\beta}$:

Listing 2.2: A sample Maple code for calculation $M_{\alpha\beta}$

```
z := array(1 .. 3);
hv := array(1 .. 6);
g := array(1 .. 6, 1 .. 6, 1 .. 2, 1 .. 2);
hv[1] := z[1]*(-1+2*z[1]);
hv[2] := z[2]*(-1+2*z[2]);
hv[3] := z[3]*(-1+2*z[3]);
hv[4] := 4*z[1]*z[2];
hv[5] := 4*z[2]*z[3];
hv[6] := 4*z[1]*z[3];
for alpha to 6 do for beta to 6 do
z[3] := 1-z[1]-z[2];
eq4 := eval(int(hv[alpha]*hv[beta], z[1] = 0 .. 1-z[2]));
M[alpha, beta] := int(eq4, z[2] = 0 .. 1);
unassign('z'); z := array(1 .. 3)
end do end do;
M = eval(M)
```

2.5.5 D_1 and D_2 Matrices

For solving stokes equations we also need to calculate D_i matrices for each element. The following relation could be used to solve the equations.

$$\hat{D_i}^k_{\alpha\beta} = \int_{\Omega^k} h^p_\alpha \frac{h^v_\beta}{x_i} d\mathbf{x} \tag{2.32}$$

It will be convenient to write $\hat{D_1}^k_{\alpha\beta}$ and $\hat{D_2}^k_{\alpha\beta}$ as follows:

$$\hat{D_1}^k_{\alpha\beta} = 2area^k \left(\sum_{m=1}^{2} D^k_{m1} H^k_{m\alpha\beta} \right) \tag{2.33}$$

$$\hat{D_2}^k_{\alpha\beta} = 2area^k \left(\sum_{m=1}^{2} D^k_{m2} H^k_{m\alpha\beta} \right) \tag{2.34}$$

where

$$H^k_{1\alpha\beta} = \int_0^1 \int_0^{1-\xi_2} h^p_\alpha \frac{\partial h^v_\beta}{\partial \xi_1} d\xi_1 d\xi_2 \tag{2.35}$$

$$H^k_{2\alpha\beta} = \int_0^1 \int_0^{1-\xi_2} h^p_\alpha \frac{\partial h^v_\beta}{\partial \xi_2} d\xi_1 d\xi_2 \tag{2.36}$$

$$\tag{2.37}$$

There is a Maple code to calculate $H^k_{1\alpha\beta}$ and $H^k_{2\alpha\beta}$:

Listing 2.3: A sample Maple code for calculation $H_{1\alpha\beta}$ and $H_{2\alpha\beta}$

```
z := array(1 .. 3);
hv := array(1 .. 6);
g := array(1 .. 6, 1 .. 6, 1 .. 2, 1 .. 2);
hv[1] := z[1]*(-1+2*z[1]);
hv[2] := z[2]*(-1+2*z[2]);
hv[3] := z[3]*(-1+2*z[3]);
hv[4] := 4*z[1]*z[2];
hv[5] := 4*z[2]*z[3];
hv[6] := 4*z[1]*z[3];
H_1 := array(1 .. 3, 1 .. 6);
```

```
H_2 := array(1 .. 3, 1 .. 6);
hp := array(1 .. 3);
hp[1] := z[1];
hp[2] := z[2];
hp[3] := 1-z[1]-z[2];
for alpha to 3 do for beta to 6 do
z[3] := 1-z[1]-z[2];
eq1 := hp[alpha];
eq2 := diff(hv[beta], z[1]);
eq3 := eq1*eq2;
eq4 := eval(int(eq3, z[1] = 0 .. 1-z[2]));
eq5 :=diff(hv[beta], z[2]);
eq6 := eval(int(eq3, z[1] = 0 .. 1-z[2]));
H_1[alpha, beta] := int(eq4, z[2] = 0 .. 1);
H_2[alpha, beta] := int(eq6, z[2] = 0 .. 1)
unassign('z'); z := array(1 .. 3)
end do end do;
 H_1 = eval(H_1);
 H_2 = eval(H_2);
```

2.5.6 Constructing global matrices

Before assembling the system forcing vector global matrices should be constructed from local ones. The following algorithms would be effective to construct global

stiffness, mass, D_1 and D_2 matrices.

forall the *Elements(k)* **do**
  for $\alpha = 1, \ldots, 6$ **do**
    $i = g_\alpha^k$
    for $\beta = 1, \ldots, 6$ **do**
      $j = g_\beta^k$
      $\tilde{A}_{ij} = \tilde{A}_{ij} + \tilde{A}_{\alpha\beta}^k$
      $\tilde{M}_{ij} = \tilde{M}_{ij} + \tilde{M}_{\alpha\beta}^k$
    end
  end
end

Algorithm 2: Constructing global stiffness and mass matrices

forall the *Elements(k)* **do**
  for $\alpha = 1, \ldots, 3$ **do**
    $i = (g^p)_\alpha^k$
    for $\beta = 1, \ldots, 6$ **do**
      $j = (g^v)_\beta^k$
      $\tilde{D}_{1ij} = \tilde{D}_{1ij} + \tilde{D}_{1\alpha\beta}^{\ k}$
      $\tilde{D}_{2ij} = \tilde{D}_{2ij} + \tilde{D}_{2\alpha\beta}^{\ k}$
    end
  end
end

Algorithm 3: Constructing global D_1 and D_2 matrices

2.5.7 Assembling the block form

After assembling mass matrix, right hand side of the equation can be calculated by multiplying mass matrix and force vector (obtained from function f in Eq. 3.5). For velocity that has two component, two distinct f_1 and f_2 should multiplied by the global mass matrix. So at this step we should assemble constructed global stiffness, D_1 and D_2 matrices as Eq.3.5. Finally our system of equations is constructed and boundary conditions should be applied and final system of equations should be solved with an appropriate algorithm.

2.5.8 Boundary Conditions

Applying boundary condition at the last step can be done with different methods. One of the most useful ways of applying boundary conditions is to eliminate boundary nodes from final system of equations and solve the system just for internal nodes. One important point that should be kept in mind is that treating zero and nonzero boundary conditions should be done carefully. For applying boundary conditions columns and rows which corresponds to boundary nodes will be removed from the coefficient matrix and the value of boundary condition will be multiplied by removed column and subtracted from right hand side of the equation. One more point in dealing boundary nodes is treating corners which will be discussed in the following sections.

2.5.9 Solver

After applying boundary conditions the system of equations is obtained:

$$\mathbf{Ax} = \mathbf{b} \tag{2.38}$$

LU decomposition is used to solve the obtain linear system of equations. In order to validate the accuracy of the solver, a sample linear system of equations is solved as following:

$$\begin{bmatrix} 3 & 2 & -1 \\ 2 & -2 & 4 \\ -1 & \frac{1}{2} & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix} \tag{2.39}$$

The resulting **x** is [1,âĹŠ2,âĹŠ2], hence the solution $x = 1, y =$ âĹŠ2, $z =$ âĹŠ2. So as demonstrated the solver is capable of solving system of equations accurately and will be used for solving current system of equations.

2.6 Validation

Validation of the code will be performed by comparing the result with an exact solution of the problem and two different error norms would be calculated to check the convergence of the numerical solution. If we assume the following solutions as an

exact solutions of stokes equation Eq. 3.5:

$$u^\star = \pi \sin(2\pi y) \sin^2(\pi x) \tag{2.40}$$

$$v^\star = -\pi \sin(2\pi x) \sin^2(\pi y) \tag{2.41}$$

$$p^\star = \cos(\pi x) \sin(\pi y) \tag{2.42}$$

Now after plugging in exact solutions into Eq. 3.5, f_1 and f_2 can be calculated as follows:

$$f_1 = -\nabla^2 u^\star + \nabla p^\star = 2\pi^3 \sin(2\pi y) \cos(\pi x)^2 - 6\pi^3 \sin(2\pi y) \sin(\pi x)^2 - \pi \sin(\pi x) \sin(\pi y) \tag{2.43}$$

$$f_2 = -\nabla^2 v^\star + \nabla p^\star = 6\pi^3 \sin(2\pi x) \sin(\pi y)^2 - 2\pi^3 \sin(2\pi x) \cos(\pi y)^2 + \pi \cos(\pi x) \cos(\pi y) \tag{2.44}$$

Exact solutions of this trumped up problem is plotted in Fig. 3.5. Now by solving the trumped-up stokes equation using the developed finite element code we will get the results shown in Fig. 3.6. Two different error norms can be used to check the convergence of the solution.

$$L_2 = \sqrt{(u^* - u_h, u^* - u_h)} \tag{2.45}$$

$$H_1 = \sqrt{a(u^* - u_h, u^* - u_h)} \tag{2.46}$$

In order to check the convergence, two error norms is plotted in Fig. 3.7 for x direction velocities and in Fig. 3.8 for y direction velocities. As it is shown in these figures the errors are reasonable and the problem is converged. One of the most important errors encountered in numerical simulations corresponds to spatial discretization, so, more refined discretization would lead to more accurate numerical results. Besides, decreasing the difference between exact and numerical solution of the problem will be interpreted as getting more accurate results. In order to check the convergence of the problem and its relation to refinement of grids it would be beneficial to compare two different error norms which are plotted in Fig. 3.7 for both x velocity and Fig.3.8 y velocity directions. Comparing these results reveals that difference between numerical

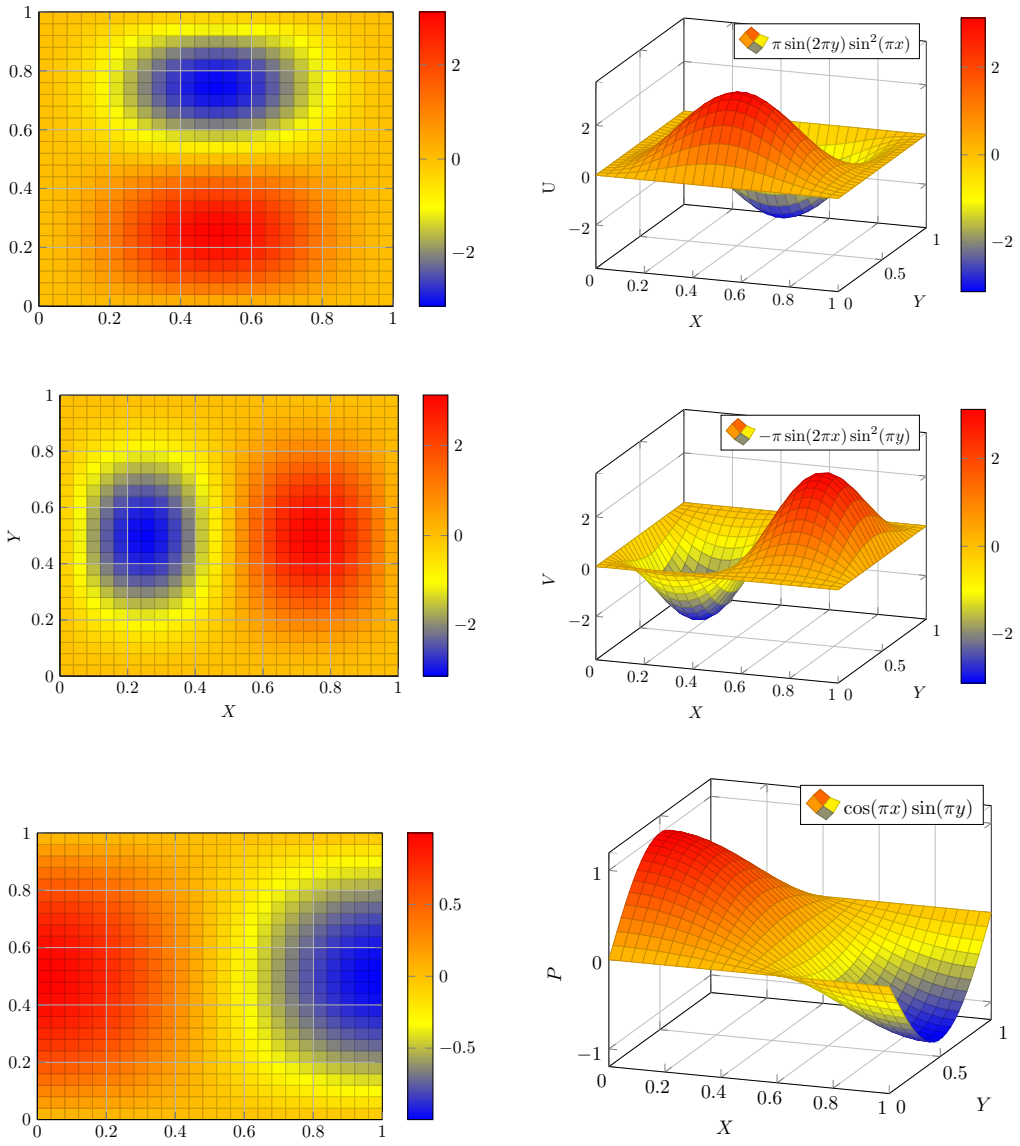


Figure 2.5: Exact solution of stoke equation

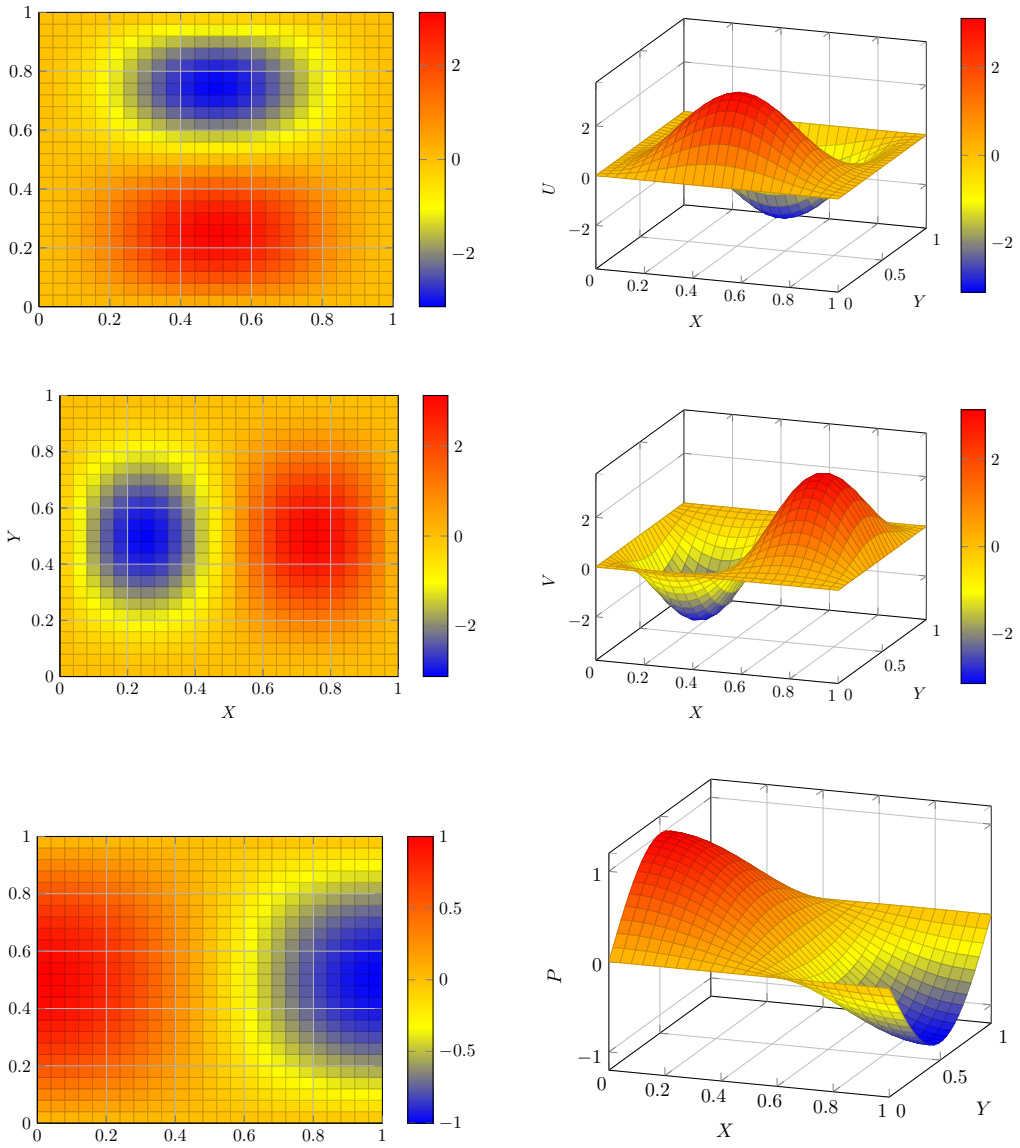


Figure 2.6: Numerical solution of trumped-up stokes equation

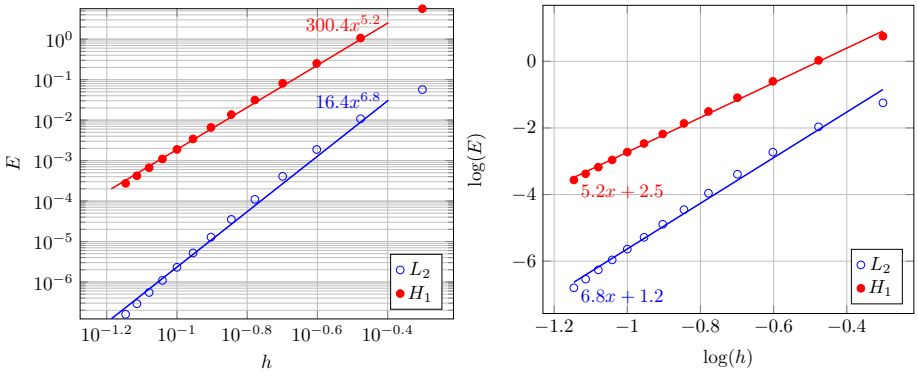


Figure 2.7: Plots of error norms obtained from solution of stokes equation for x direction velocities

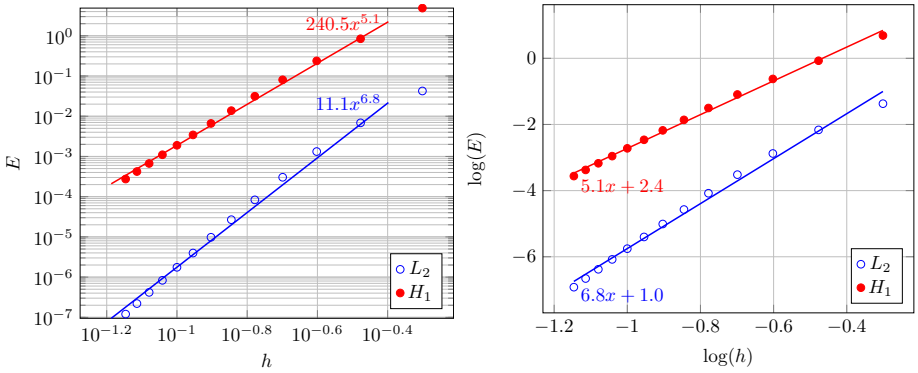


Figure 2.8: Plots of error norms obtained from solution of stokes equation for y direction velocities

simulation and exact simulation decreases with more refined meshes which is expected in numerical simulations. Error norms for different grid sizes are summarized in Table. 2.2.

Table 2.2: Error norms for different grid sizes for X and Y direction velocities

	Y direction Velocity		X direction Velocity	
h	L_2	H_1	L_2	H_1
1/2	0.04	4.886503	0.056690567	5.662406
1/3	0.007	0.844839	0.010798582	1.067564
1/4	0.001	0.238007	0.001877319	0.251471
1/5	0.0003	0.080615	0.000407452	0.080956
1/6	8.33E-05	0.031423	0.000110022	0.031304
1/7	2.68E-05	0.013769	3.52E-05	0.013697
1/8	9.79E-06	0.006623	1.28E-05	0.006588
1/9	3.97E-06	0.003436	5.21E-06	0.003418
1/10	1.76E-06	0.001897	2.31E-06	0.001887
1/11	8.33E-07	0.001103	1.09E-06	0.001097
1/12	4.18E-07	0.00067	5.50E-07	0.000666
1/13	2.20E-07	0.000422	2.90E-07	0.00042
1/14	1.20E-07	0.000275	1.59E-07	0.000273

2.7 Results

After validation, it's time to simulate a benchmark problem named 'Lid Driven Cavity'. The geometry of the problem consists of a unit square with homogeneous dirichlet boundary conditions on three walls which is zero and dirichlet boundary condition on the top edge which it's velocity is 1. In order to solve the problem we need to solve

stokes equation:

$$-\nabla^2 u + \nabla p = 0 \quad \text{in} \quad \Omega \tag{2.47}$$

$$-\nabla u = 0 \quad \text{in} \quad \Omega \tag{2.48}$$

$$u = 1\hat{x} \quad \text{on} \quad \Omega_{top} \tag{2.49}$$

$$u = 0\hat{x} \quad \text{on} \quad \Omega \backslash \Omega_{top} \tag{2.50}$$

$$\tag{2.51}$$

using the validated code in the previous section, the problems is solved and velocity field and streamlines are shown in Fig. 2.9 and 2.10 for $N = 8$ and $N = 16$, respectively. Contours of pressure, x-direction velocity and y-direction velocity are shown in Fig. 2.11.

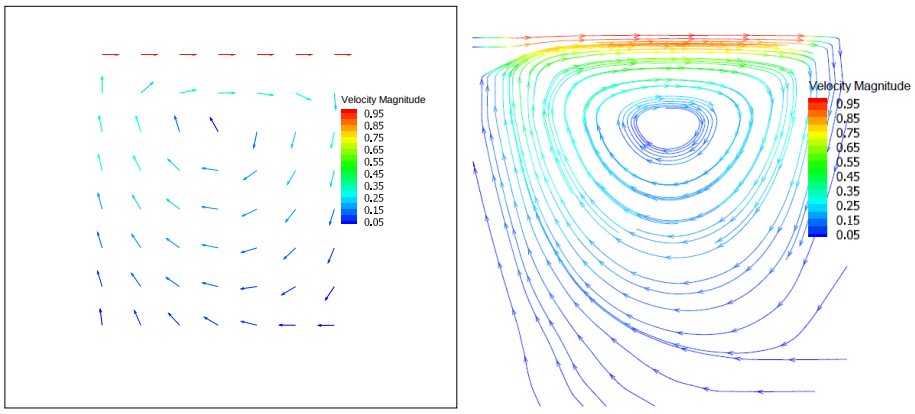


Figure 2.9: Velocity field and streamlines of lid driven cavity for $N = 8$

2.7.1 Discontinous Dirichlet BCs

In this problem the x-component of velocity is zero along the side walls and nonzero along the top lid. The x velocity is discontinuous at the two top corners, and it is necessary to choose a scheme for assigning the boundary value at the points of discontinuity. Several ideas suggested :

- Pick one of the two boundary conditions and apply it at the corner point
- Average the two boundary conditions

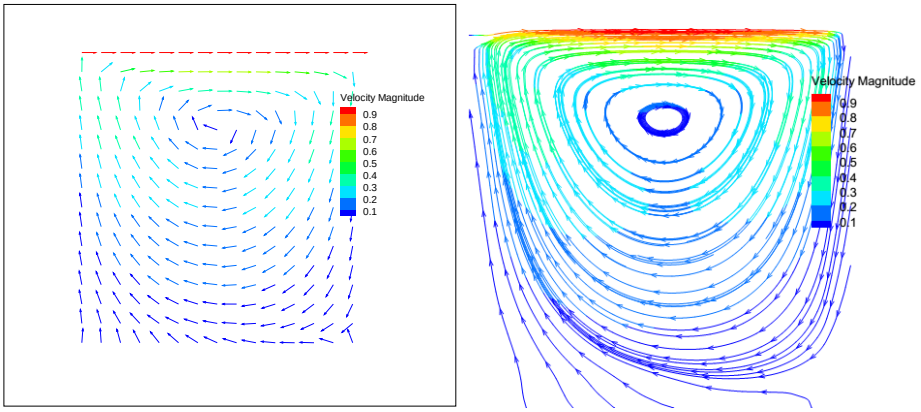
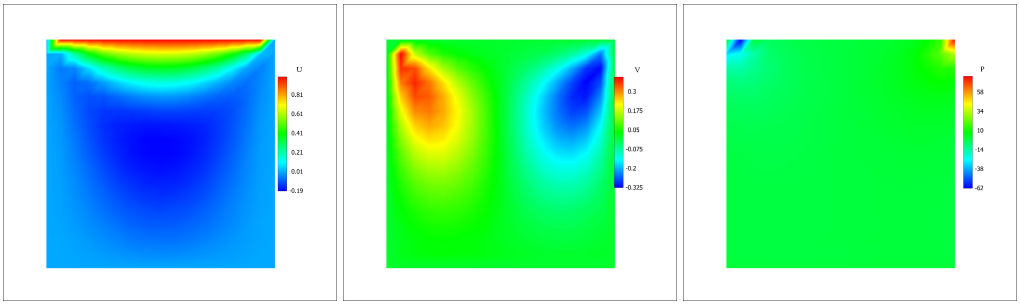


Figure 2.10: Velocity field and streamlines of lid driven cavity for $N = 16$



(a) X direction velocity **(b)** Y direction velocity **(c)** Pressure

Figure 2.11: Contours of lid driven cavity for $N = 16$

- Weighted average of two boundary conditions

In the following figures different approaches are compared to each other.

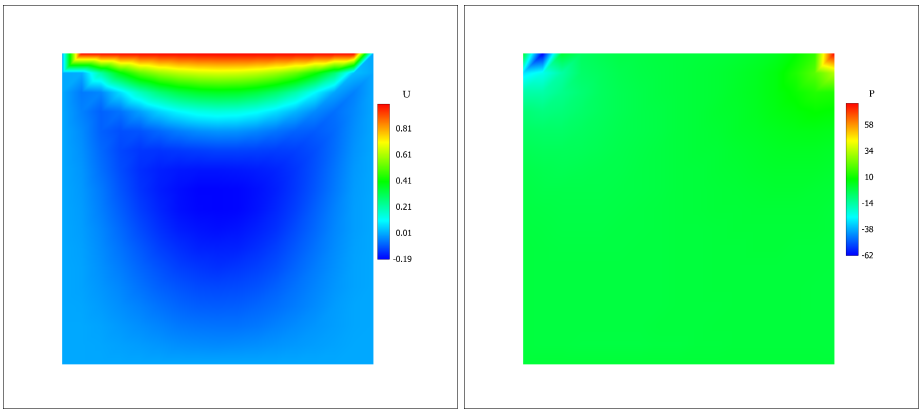


Figure 2.12: Picked $U = 0$ at corners

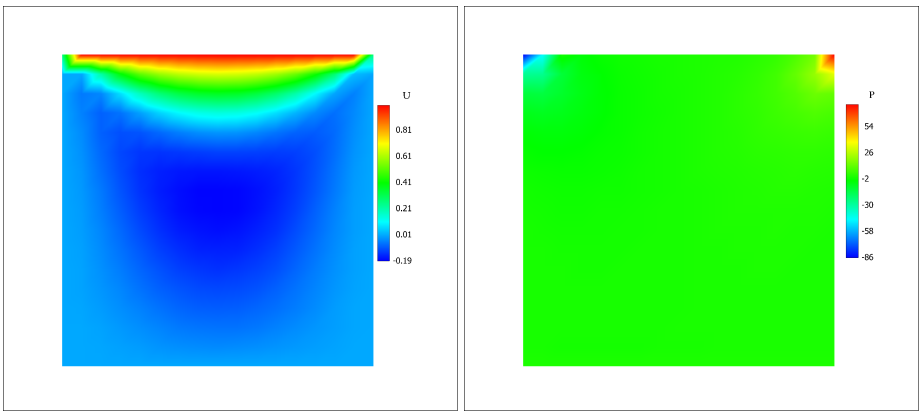


Figure 2.13: Picked weighted average, $U = 0.25$, at corners

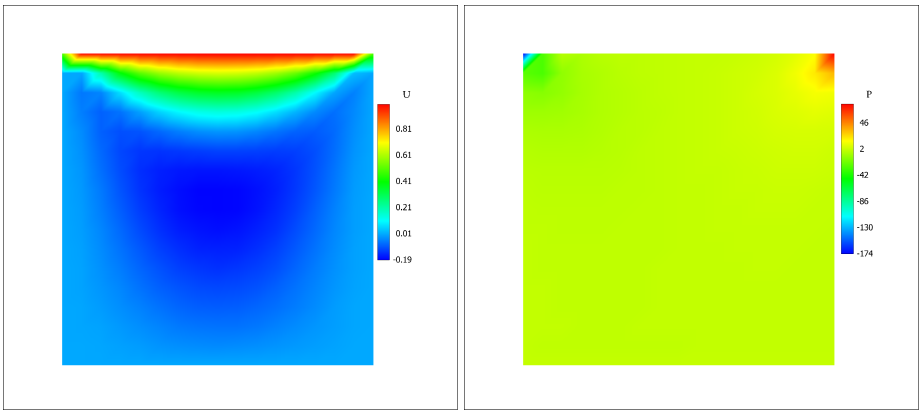


Figure 2.14: Picked average, $U = 0.5$, at corners

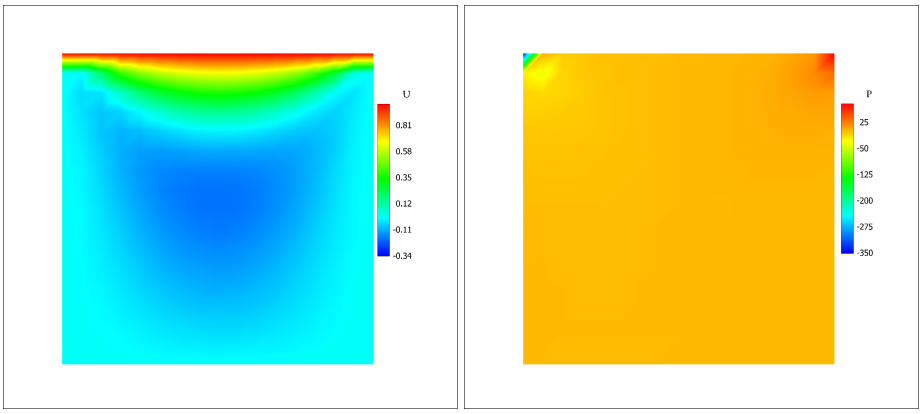


Figure 2.15: Picked $U = 1$ at corners

Chapter 3

Navier-Stokes Solver

3.1 Introduction

The main purpose of this project is to develop a code to solve transient sotkes equation with dirichlet boundary conditions.To this end a stokes equation solver is added to the developed software(Fig. 3.1)

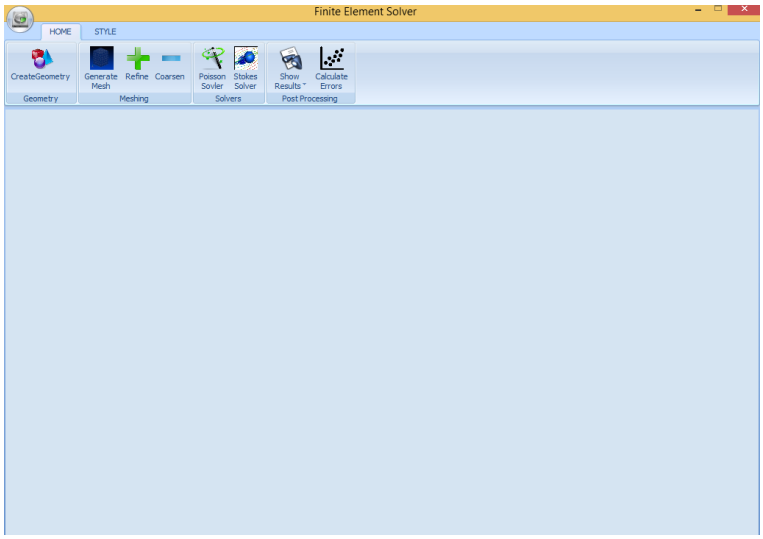


Figure 3.1: General view of software developed for solving stokes equation

As it is shown in Fig. 3.1, the software is capable of generation domain and mesh. Also it is possible to refine the mesh in order to obtain better results. After mesh generation stokes solver will solve the problem and results will be shown for each node.

Finally, errors of the calculation can be plotted to get an insight about convergence of the solver.

3.2 Problem

In this project we're solving stokes equation

$$\frac{\partial u}{\partial t} = \nabla^2 u - \nabla p + f \quad \text{in} \quad \Omega \tag{3.1}$$

$$\nabla u = 0 \quad \text{in} \quad \Omega \tag{3.2}$$

$$u = 1\hat{x} \quad \text{on} \quad \Omega_{top} \tag{3.3}$$

$$u = 0\hat{x} \quad \text{on} \quad \Omega \backslash \Omega_{top} \tag{3.4}$$

$$\tag{3.5}$$

The following system of equations could be obtained from weak formulation of the problem:

$$\begin{pmatrix} A + \frac{M}{dt} & 0 & -D_1^T \\ 0 & A + \frac{M}{dt} & -D_2^T \\ -D_1 & -D_2 & 0 \end{pmatrix} \begin{pmatrix} u_h \\ v_h \\ p_h \end{pmatrix}^{n+1} = \begin{pmatrix} \frac{M}{dt} u \\ \frac{M}{dt} v \\ 0 \end{pmatrix}^{n} + \begin{pmatrix} Mf_1 \\ Mf_2 \\ 0 \end{pmatrix}^{n+1} \tag{3.6}$$

where A is globals stiffness matrix and n corresponds to time step.

3.3 Geometry

The geometry of this project is a unit square which is specified with four nodes located on the corners of the geometry. In order to generate the geometry there is a command button (Fig. 3.1) by which the domain of the solution would be created.

3.4 Mesh generation

In order to obtain accurate numerical results it is of the utmost importance to choose an appropriate mesh type. Finite element method has several advantages over other methods and mesh generation is one of the most important of them. Finite element is able to solve equations using triangular elements which is a crucial choice when dealing with complex geometries.

Not every finite element can be used to solve stokes equation. One of the choices to solve the stokes equation is Taylor and Hood elements which is a combination of P2 and P1 triangular elements as shown in Fig. 3.2

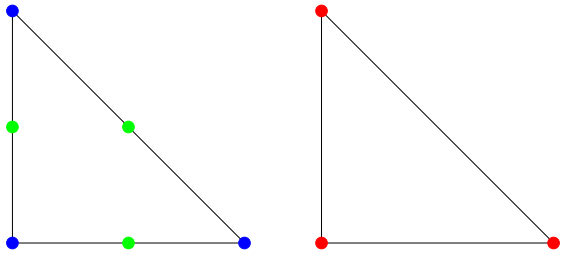


Figure 3.2: P2 and P1 Triangular elements used in Taylor and Hood elements

Clicking on the "Generate Mesh" button the software will regularly discretize the domain into P2 elements. To obtain better and more accurate results the mesh should be fine enough, so, after mesh generation by clicking on the "Refine" button the mesh will be refined. In Fig. 3.3 a sample of generated and refined meshes is shown. After mesh generation there is a possibility to export the mesh to different formats which Fig. 3.3 illustrated an exported mesh from the software.

3.5 Numerical Procedure

3.5.1 Affine mapping

In order to facilitate the numerical procedure, each element is mapped into a reference element using a mapping function (Fig. 3.4).

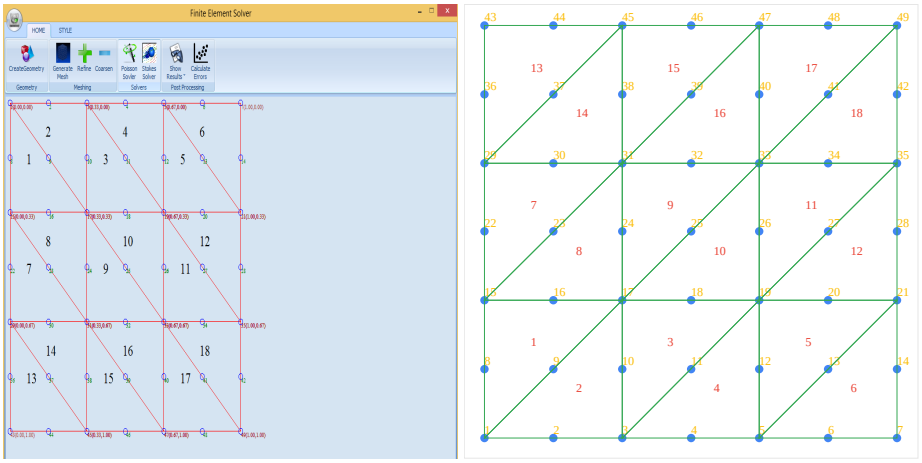


Figure 3.3: A sample generated mesh using the developed software

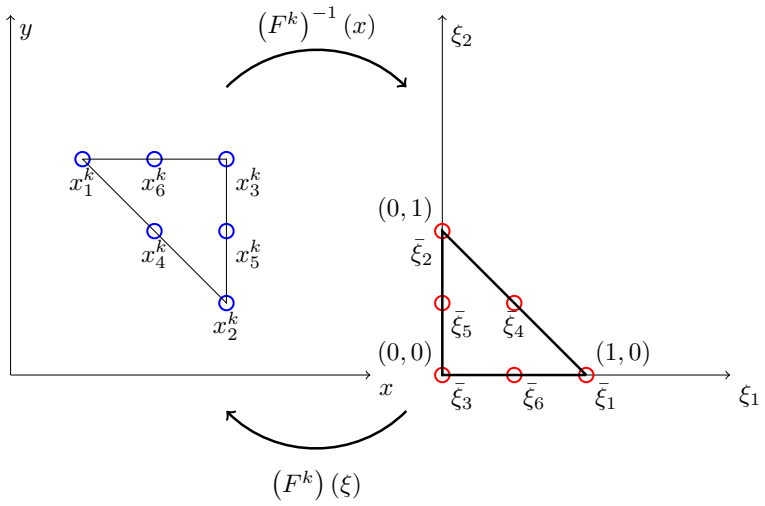


Figure 3.4: Affine Mapping

The following relation can be used to transform every node in original space to reference space:

$$\begin{pmatrix}\xi_1\\ \xi_2\end{pmatrix} = \begin{pmatrix}C_1^k\\ C_2^k\end{pmatrix} + \begin{pmatrix}D_{11} & D_{12}\\ D_{21} & D_{22}\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} \tag{3.7}$$

where

$$C_i^k = \frac{1}{2area^k}\{\hat{x}_{[i+1]}^k \hat{y}_{[i+2]}^k - \hat{x}_{[i+2]}^k \hat{y}_{[i+1]}^k\} \tag{3.8}$$

$$D_{i1}^k = \frac{1}{2area^k}\{\hat{y}_{[i+1]}^k - \hat{y}_{[i+2]}^k\} \tag{3.9}$$

$$D_{i2}^k = \frac{1}{2area^k}\{\hat{x}_{[i+2]}^k - \hat{x}_{[i+1]}^k\} \tag{3.10}$$

3.5.2 Basis function

P1 Elements Any function defined on a P1 triangle can be approximated by the linear function:

$$u(x,y) = a_1 + a_2 x + a_3 y \tag{3.11}$$

and in the transformed system we obtain

$$u(\xi_1,\xi_2) = h_1 + h_2\xi_1 + h_3\xi_2 \tag{3.12}$$

where

$$h_1 = \xi_1 \tag{3.13}$$

$$h_2 = \xi_2 \tag{3.14}$$

$$h_3 = (1 - \xi_1 - \xi_2) \tag{3.15}$$

$$\tag{3.16}$$

Reference elemental basis has the following property:

$$h_i(\xi_i) = \delta_{ij} \tag{3.17}$$

For example considering Fig. 3.4, for $\bar{\xi_1} = (1,0)$, $h_1(\bar{\xi_1}) = 1$, $h_1(\bar{\xi_2}) = 0$,$h_1(\bar{\xi_3}) = 0$. Also this is true for other points, $h_2(\bar{\xi_1}) = 0$, $h_2(\bar{\xi_2}) = 1$,$h_2(\bar{\xi_3}) = 0$.

P2 Elements Any function defined on a triangle can be approximated by the quadratic function:

$$u(x,y) = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2 \tag{3.18}$$

and in the transformed system we obtain

$$u(\xi_1,\xi_2) = h_1 + h_2\xi_1 + h_3\xi_2 + h_4\xi_1^2 + h_5\xi_1\xi_2 + h_6\xi_2^2 \tag{3.19}$$

where

$$h_1 = \xi_1(-1+2\xi_1) \tag{3.20}$$
$$h_2 = \xi_2(-1+2\xi_2) \tag{3.21}$$
$$h_3 = (1-\xi_1-\xi_2)(1-2\xi_2-2\xi_2) \tag{3.22}$$
$$h_4 = 4\xi_1\xi_2 \tag{3.23}$$
$$h_5 = 4\xi_2(1-\xi_1-\xi_2) \tag{3.24}$$
$$h_6 = 4\xi_1(1-\xi_1-\xi_2) \tag{3.25}$$

Reference elemental basis has the following property:

$$h_i(\xi_i) = \delta_{ij} \tag{3.26}$$

For example considering Fig. 3.4, for $\bar{\xi_1} = (1,0)$, $h_1(\bar{\xi_1}) = 1$, $h_1(\bar{\xi_2}) = 0$,$h_1(\bar{\xi_3}) = 0$, $h_1(\bar{\xi_4}) = 0$, ... Also this is true for other points, $h_2(\bar{\xi_1}) = 0$, $h_2(\bar{\xi_2}) = 1$,$h_2(\bar{\xi_3}) = 0$, $h_2(\bar{\xi_4}) = 0$, ...

3.5.3 Stiffness Matrix

In order to solve the problem we need to calculate stiffness matrix for each element. Stiffness Matrix ($\hat{A}^k_{\alpha\beta}$) for each element which is a 6×6 elemental matrix can be

constructed using the following equation:

$$\hat{A}_{\alpha\beta} = \int_{\Omega^k} \frac{\partial h_\alpha^\nu}{\partial x_1}\frac{\partial h_\beta^\nu}{\partial x_1} + \frac{\partial h_\alpha^\nu}{\partial x_2}\frac{\partial h_\beta^\nu}{\partial x_2} dxdy \tag{3.27}$$

It will be convenient to write $\hat{A}^k_{\alpha\beta}$ as follows:

$$\hat{A}^k_{\alpha\beta} = 2area^k \left(\sum_{m=1}^{2}\sum_{n=1}^{2}\sum_{n'=1}^{2} D^k_{nm} D^k_{n'm} G_{\alpha\beta nn'} \right) \tag{3.28}$$

where

$$G^k_{\alpha\beta} = \int_0^1 \int_0^{1-\xi_2} \frac{\partial h_\alpha^\nu}{\partial \xi_n}\frac{\partial h_\beta^\nu}{\partial \xi_{n'}} d\xi_1 d\xi_2 \tag{3.29}$$

3.5.4 Mass Matrix

Asides from stiffness matrix, mass matrix should also be calculated from the following equation for each element:

$$\hat{M}^k_{\alpha\beta} = \int_{\Omega^k} h_\alpha^\nu h_\beta^\nu dxdy \tag{3.30}$$

It will be convenient to write $\hat{M}^k_{\alpha\beta}$ as follows:

$$\hat{M}^k_{\alpha\beta} = 2area^k \int_{\Omega^k} h_\alpha^\nu h_\beta^\nu d\xi_1 d\xi_2 \tag{3.31}$$

3.5.5 D_1 and D_2 Matrices

For solving stokes equations we also need to calculate D_i matrices for each element. The following relation could be used to solve the equations.

$$\hat{D}_{i\,\alpha\beta}^{\;k} = \int_{\Omega^k} h_\alpha^p \frac{h_\beta^\nu}{x_i} d\mathbf{x} \tag{3.32}$$

It will be convenient to write $\hat{D_1}^k_{\alpha\beta}$ and $\hat{D_2}^k_{\alpha\beta}$ as follows:

$$\hat{D_1}^k_{\alpha\beta} = 2area^k \left(\sum_{m=1}^{2} D^k_{m1} H^k_{m\alpha\beta} \right) \tag{3.33}$$

$$\hat{D_2}^k_{\alpha\beta} = 2area^k \left(\sum_{m=1}^{2} D^k_{m2} H^k_{m\alpha\beta} \right) \tag{3.34}$$

where

$$H^k_{1\alpha\beta} = \int_0^1 \int_0^{1-\xi_2} h^p_\alpha \frac{\partial h^\nu_\beta}{\partial \xi_1} d\xi_1 d\xi_2 \tag{3.35}$$

$$H^k_{2\alpha\beta} = \int_0^1 \int_0^{1-\xi_2} h^p_\alpha \frac{\partial h^\nu_\beta}{\partial \xi_2} d\xi_1 d\xi_2 \tag{3.36}$$

$$\tag{3.37}$$

3.5.6 Constructing global matrices

Before assembling the system forcing vector global matrices should be constructed from local ones. The following algorithms would be effective to construct global

stiffness, mass, D_1 and D_2 matrices.

forall the *Elements(k)* **do**
  for $\alpha = 1, \ldots, 6$ **do**
    $i = g_\alpha^k$
    for $\beta = 1, \ldots, 6$ **do**
      $j = g_\beta^k$
      $\tilde{A}_{ij} = \tilde{A}_{ij} + \tilde{A}_{\alpha\beta}^k$
      $\tilde{M}_{ij} = \tilde{M}_{ij} + \tilde{M}_{\alpha\beta}^k$
    end
  end
end

Algorithm 4: Constructing global stiffness and mass matrices

forall the *Elements(k)* **do**
  for $\alpha = 1, \ldots, 3$ **do**
    $i = (g^p)_\alpha^k$
    for $\beta = 1, \ldots, 6$ **do**
      $j = (g^v)_\beta^k$
      $\tilde{D}_{1ij} = \tilde{D}_{1ij} + \tilde{D}_{1\alpha\beta}^{\ k}$
      $\tilde{D}_{2ij} = \tilde{D}_{2ij} + \tilde{D}_{2\alpha\beta}^{\ k}$
    end
  end
end

Algorithm 5: Constructing global D_1 and D_2 matrices

3.5.7 Boundary Conditions

Applying boundary condition at the last step can be done with different methods. One of the most useful ways of applying boundary conditions is to eliminate boundary nodes from final system of equations and solve the system just for internal nodes. One important point that should be kept in mind is that treating zero and nonzero boundary conditions should be done carefully. For applying boundary conditions columns and rows which corresponds to boundary nodes will be removed from the coefficient matrix and the value of boundary condition will be multiplied by removed column and subtracted from right hand side of the equation. One more point in

dealing boundary nodes is treating corners which will be discussed in the following sections.

3.5.8 Solver

After applying boundary conditions the system of equations is obtained:

$$\mathbf{Ax} = \mathbf{b} \tag{3.38}$$

LU decomposition is used to solve the obtain linear system of equations. In this problem since the coeficient matrix is the same in all timesteps, L and U matrices are stored and used in each iteration to avoid calculating them in each time step because its an time consuming process.

3.6 Validation

Validation of the code will be performed by comparing the result with an exact solution of the problem and two different error norms would be calculated to check the convergence of the numerical solution. If we assume the following solutions as an exact solutions of stokes equation Eq. 3.5:

$$u^\star = \pi \sin(2\pi y) \sin^2(\pi x) \sin(t) \tag{3.39}$$

$$v^\star = -\pi \sin(2\pi x) \sin^2(\pi y) \sin(t) \tag{3.40}$$

$$p^\star = \cos(\pi x) \sin(\pi y) \sin(t) \tag{3.41}$$

Now after plugging in exact solutions into Eq. 3.5, f_1 and f_2 can be calculated as follows:

$$
\begin{aligned}
f_1 &= \frac{\partial u}{\partial t} - \nabla^2 u^\star + \nabla p^\star = \\
&\pi \sin\left(2\pi y\right) \left(\sin\left(\pi x\right)\right)^2 \cos\left(t\right) \\
&- 2\pi^3 \sin\left(2\pi y\right) \left(\cos\left(\pi x\right)\right)^2 \sin\left(t\right) \\
&+ 6\pi^3 \sin\left(2\pi y\right) \left(\sin\left(\pi x\right)\right)^2 \sin\left(t\right) \\
&- \sin\left(\pi x\right) \pi \sin\left(\pi y\right) \sin\left(t\right) \\
f_2 &= \frac{\partial v}{\partial t} - \nabla^2 v^\star + \nabla p^\star = \\
&- \pi \sin\left(2\pi x\right) \left(\sin\left(\pi y\right)\right)^2 \cos\left(t\right) \\
&- 6\pi^3 \sin\left(2\pi x\right) \left(\sin\left(\pi y\right)\right)^2 \sin\left(t\right) \\
&+ 2\pi^3 \sin\left(2\pi x\right) \left(\cos\left(\pi y\right)\right)^2 \sin\left(t\right) \\
&+ \cos\left(\pi x\right) \cos\left(\pi y\right) \pi \sin\left(t\right)
\end{aligned}
$$

Exact solutions of this trumped up problem is plotted in Fig. 3.5. Now by solving the trumped-up unsteady stokes equation using the developed finite element code we will get the results shown in Fig. 3.6. Two different error norms can be used to check the convergence of the solution.

$$L_2 = \sqrt{(u^* - u_h, u^* - u_h)} \tag{3.42}$$

$$H_1 = \sqrt{a\,(u^* - u_h, u^* - u_h)} \tag{3.43}$$

In order to check the temporal convergence, two error norms is plotted in Fig. 3.7 and Fig. 3.8 for x and y direction velocities. As it is shown in these figures the results are converge with a slope of nearly 1 which is spected.

One of the most important errors encountered in numerical simulations corresponds to spatial discretization, so, more refined discretization should lead to more accurate numerical results. Besides, decreasing the difference between exact and numerical solution of the problem will be interpreted as getting more accurate results. The spatial convergence of the problem is shown in Fig. 3.9 for x direction velocities and in Fig. 3.10 for y direction velocities. Errors also are summrized in Table. 3.2.

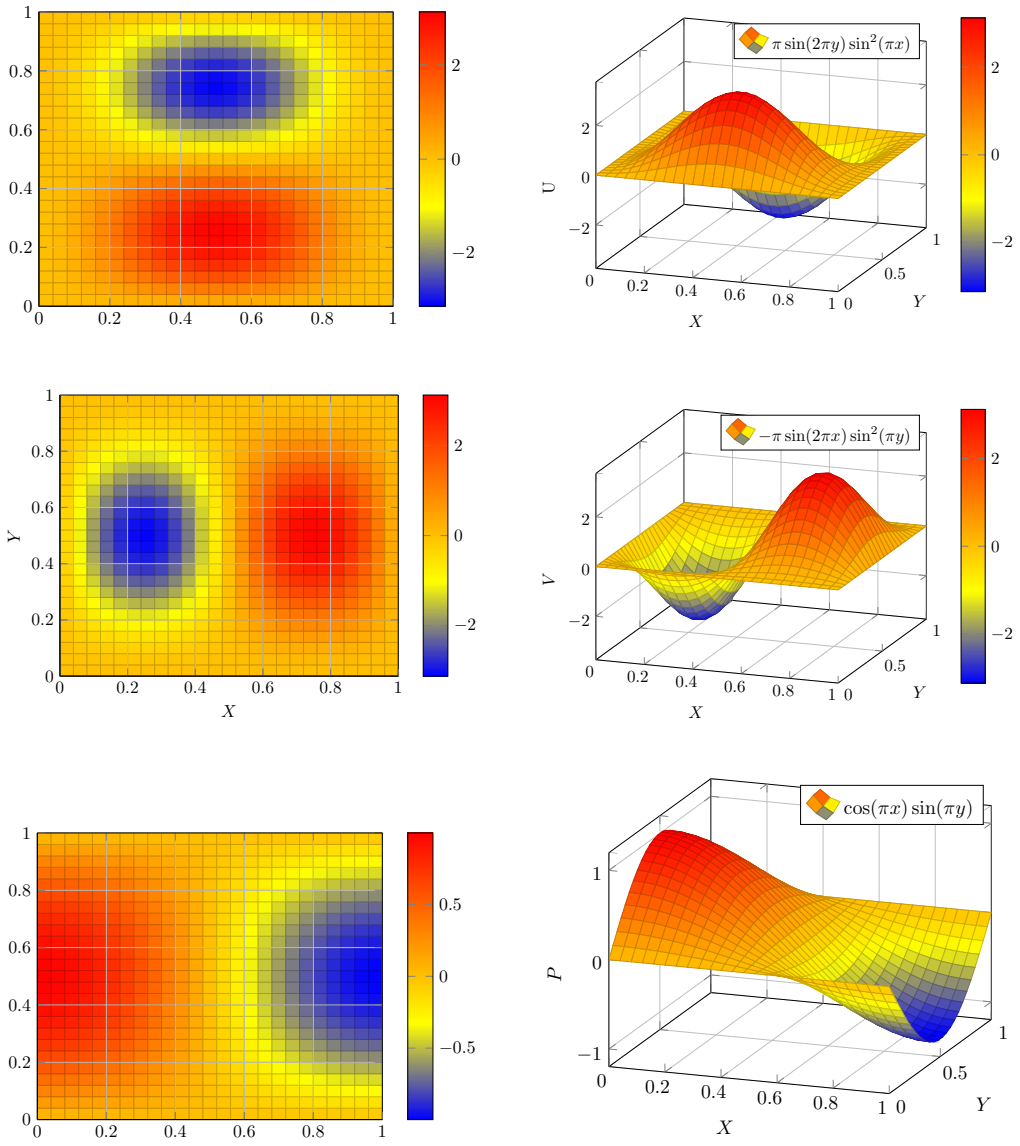


Figure 3.5: Exact solution of transient stoke equation at time=1

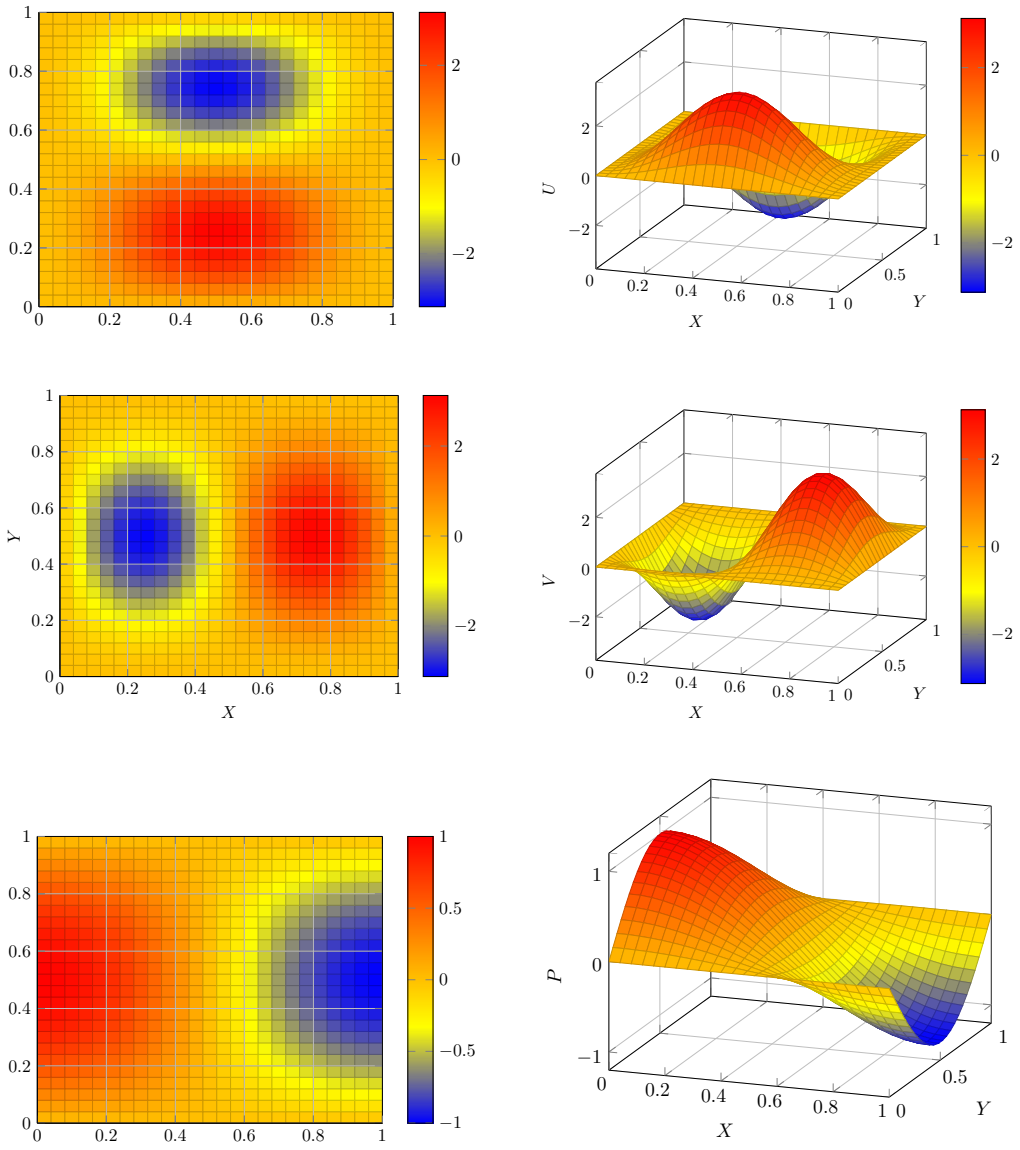


Figure 3.6: Numerical solution of trumped-up transient stokes equation at time = 1

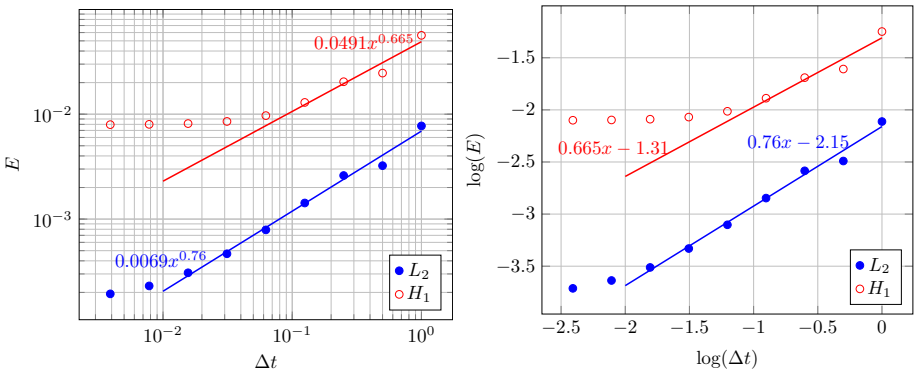


Figure 3.7: Temporal convergence of solution of the transient stokes equation for x direction velocities

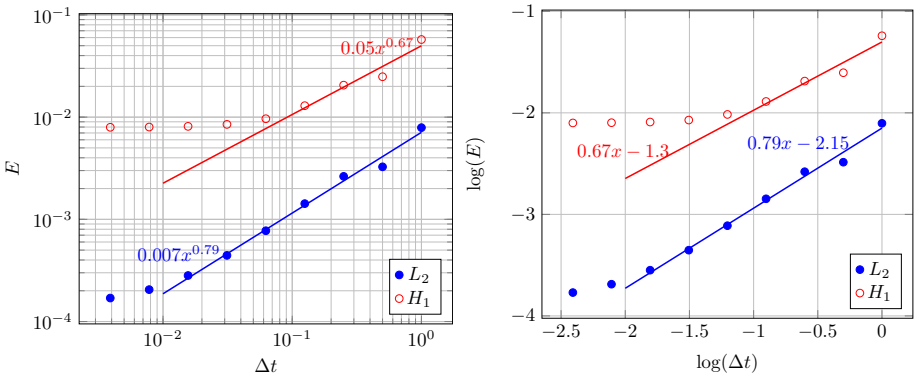


Figure 3.8: Temporal convergence of solution of the transient stokes equation for y direction velocities

Table 3.1: TemporalConvergence of transient stokes equation

	X Velocity		Y Velocity	
Δt	L_2	H_1	L_2	H_1
0.5	0.003231	0.024651	0.003265	0.024845
0.25	0.002602	0.020387	0.002633	0.020556
0.125	0.001424	0.01294	0.001424	0.012942
0.0625	0.000789	0.009691	0.000774	0.009646
0.03125	0.000467	0.008526	0.000444	0.008492
0.015625	0.000308	0.008137	0.000282	0.008122
0.007813	0.000231	0.008002	0.000205	0.008
0.003906	0.000194	0.007951	0.00017	0.007956

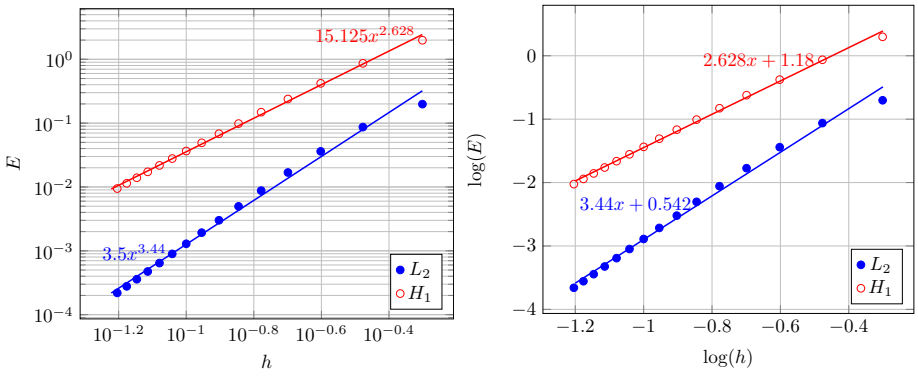


Figure 3.9: Spatial convergence of solution of the transient stokes equation for x direction velocities

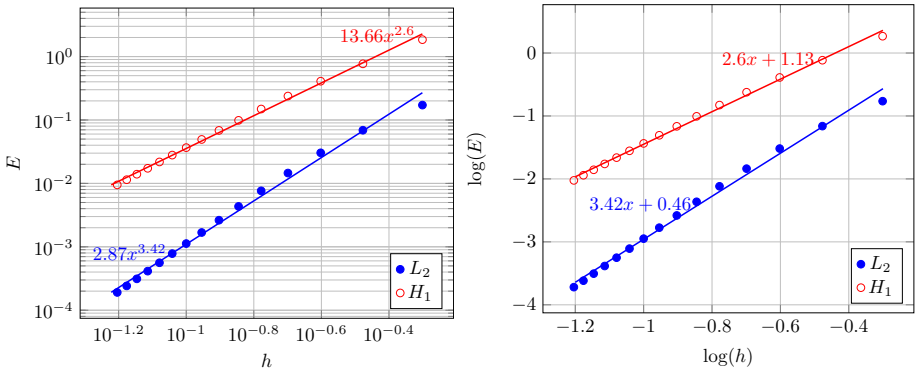


Figure 3.10: Spatial convergence of solution of the transient stokes equation for y direction velocities

Table 3.2: Spatial convergence of transient stokes equations solver

	X Velocity		Y Velocity	
h	L_2	H_1	L_2	H_1
0.5	0.198419	1.98945	0.171774	1.849419
0.333333	0.086648	0.864269	0.068963	0.768837
0.25	0.036153	0.4202	0.030355	0.408875
0.2	0.016857	0.238777	0.014559	0.238305
0.166667	0.00877	0.148629	0.007633	0.148918
0.142857	0.004967	0.09838	0.004335	0.098637
0.125	0.00301	0.068266	0.002628	0.068444
0.111111	0.001926	0.049196	0.001682	0.049317
0.1	0.001289	0.036569	0.001125	0.036653
0.090909	0.000896	0.027895	0.000781	0.027954
0.083333	0.000643	0.021748	0.00056	0.02179
0.076923	0.000474	0.017275	0.000413	0.017306
0.071429	0.000359	0.013944	0.000313	0.013967
0.066667	0.000278	0.011416	0.000242	0.011433
0.0625	0.000219	0.009462	0.000191	0.009475

3.7 Unsteady Navier-Stokes equation

3.7.1 Problem

In this section we're solving unsteady navier stokes equation

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = \frac{1}{Re}\nabla^2 u - \nabla p + f \quad \text{in} \quad \Omega \tag{3.44}$$

$$\nabla \cdot u = 0 \quad \text{in} \quad \Omega \tag{3.45}$$

$$u = 1\hat{x} \quad \text{on} \quad \Omega_{top} \tag{3.46}$$

$$u = 0\hat{x} \quad \text{on} \quad \Omega \backslash \Omega_{top} \tag{3.47}$$

$$\tag{3.48}$$

The following system of equations could be obtained from weak formulation of

the problem:

$$\begin{pmatrix} \frac{A}{Re}+\frac{M}{dt} & 0 & -D_1^T \\ 0 & \frac{A}{Re}+\frac{M}{dt} & -D_2^T \\ -D_1 & -D_2 & 0 \end{pmatrix} \begin{pmatrix} u_h \\ v_h \\ p_h \end{pmatrix}^{n+1} = \begin{pmatrix} \frac{M}{dt}u \\ \frac{M}{dt}v \\ 0 \end{pmatrix}^{n} + \begin{pmatrix} Mf_1 \\ Mf_2 \\ 0 \end{pmatrix}^{n+1} + \frac{1}{12} \begin{pmatrix} 23C^n u^n - 16C^{n-1}u^{n-1} + 5C^{n-2}u^{n-2} \\ 23C^n v^n - 16C^{n-1}v^{n-1} + 5C^{n-2}v^{n-2} \\ 0 \end{pmatrix}$$

where A is globals stiffness matrix and n corresponds to time step and C is convection matrix which will be describe in the following section.

3.7.2 Convection Matrix

Elemental convection matrix could be calculated from the following equation.

$$\hat{C}^k(i,j) = \sum_{n=1}^{Nnodes} \hat{G}_{nx}(i,j)u_n + \hat{G}_{ny}(i,j)v_n$$

where $\hat{G_{nx}}$ and $\hat{G_{ny}}$ are obtained from the following equations:

$$\hat{G_{nx}}(i,j) = \int_\Omega h_i^v h_n^v \frac{\partial h_j^v}{\partial x} d\mathbf{x} = 2area^k \sum_{m=1}^{2} D_{m1}K_{nm} \tag{3.49}$$

$$\hat{G_{ny}}(i,j) = \int_\Omega h_i^v h_n^v \frac{\partial h_j^v}{\partial y} d\mathbf{x} = 2area^k \sum_{m=1}^{2} D_{m2}K_{nm} \tag{3.50}$$

where K_{nm} can be calculated from the following equation.

$$K_{nm} = \int_0^1 \int_0^{1-\xi_2} h_i^v h_n^v \frac{\partial h_j^v}{\partial \xi_m} d\xi_1 d\xi_2 \tag{3.51}$$

Finally the global convection matrix would be evaluated from the following algorithm.

forall the *Elements(k)* **do**
  for $\alpha = 1, \ldots, 6$ **do**
    $i = (g^p)^k_\alpha$
    for $\beta = 1, \ldots, 6$ **do**
      $j = (g^v)^k_\beta$
      $\tilde{C}_{ij} = \tilde{C}_{ij} + \tilde{C}^k_{\alpha\beta}$
    end
  end
end

Algorithm 6: Constructing global convection matrix

3.8 Validation

Validation of the code will be performed by comparing the result with an exact solution of the problem and two different error norms would be calculated to check the convergence of the numerical solution. If we assume the following solutions as an exact solutions of navier stokes equation Eq. 3.48:

$$u^\star = \pi \sin(2\pi y) \sin^2(\pi x) \sin(t) \tag{3.52}$$

$$v^\star = -\pi \sin(2\pi x) \sin^2(\pi y) \sin(t) \tag{3.53}$$

$$p^\star = \cos(\pi x) \sin(\pi y) \sin(t) \tag{3.54}$$

Now after plugging in exact solutions into Eq. 3.48, f_1 and f_2 can be calculated as follows:

$$
\begin{aligned}
f_1 &= \pi \sin\left(2\pi y\right) \left(\sin\left(\pi x\right)\right)^2 \cos\left(t\right) \\
&+ 2\pi^3 \left(\sin\left(2\pi y\right)\right)^2 \left(\sin\left(\pi x\right)\right)^3 \left(\sin\left(t\right)\right)^2 \cos\left(\pi x\right) \\
&- 2\pi^3 \sin\left(2\pi x\right) \left(\sin\left(\pi y\right)\right)^2 \left(\sin\left(t\right)\right)^2 \cos\left(2\pi y\right) \left(\sin\left(\pi x\right)\right)^2 \\
&- 2\pi^3 \sin\left(2\pi y\right) \left(\cos\left(\pi x\right)\right)^2 \sin\left(t\right) \\
&+ 6\pi^3 \sin\left(2\pi y\right) \left(\sin\left(\pi x\right)\right)^2 \sin\left(t\right) \\
&- \sin\left(\pi x\right) \pi \sin\left(\pi y\right) \sin\left(t\right) \\
f_2 &= -\pi \sin\left(2\pi x\right) \left(\sin\left(\pi y\right)\right)^2 \cos\left(t\right) \\
&- 2\pi^3 \sin\left(2\pi y\right) \left(\sin\left(\pi x\right)\right)^2 \left(\sin\left(t\right)\right)^2 \cos\left(2\pi x\right) \left(\sin\left(\pi y\right)\right)^2 \\
&+ 2\pi^3 \left(\sin\left(2\pi x\right)\right)^2 \left(\sin\left(\pi y\right)\right)^3 \left(\sin\left(t\right)\right)^2 \cos\left(\pi y\right) - 6\pi^3 \sin\left(2\pi x\right) \left(\sin\left(\pi y\right)\right)^2 \sin\left(t\right) \\
&+ 2\pi^3 \sin\left(2\pi x\right) \left(\cos\left(\pi y\right)\right)^2 \sin\left(t\right) + \cos\left(\pi x\right) \cos\left(\pi y\right) \pi \sin\left(t\right)
\end{aligned}
$$

Now by solving the trumped-up unsteady navier-stokes equation using the developed finite element code we will get the results shown in Fig. 3.11. Two different error norms can be used to check the convergence of the solution.

$$L_2 = \sqrt{(u^* - u_h, u^* - u_h)} \tag{3.55}$$

$$H_1 = \sqrt{a\,(u^* - u_h, u^* - u_h)} \tag{3.56}$$

In order to check the temporal convergence, two error norms is plotted in Fig. 3.12 and Fig. 3.13 for x and y direction velocities.

One of the most important errors encountered in numerical simulations corresponds to spatial discretization, so, more refined discretization should lead to more accurate numerical results. Besides, decreasing the difference between exact and numerical solution of the problem will be interpreted as getting more accurate results. The spatial convergence of the problem is shown in Fig. 3.14 for x direction velocities and in Fig. 3.15 for y direction velocities. Errors also are summrized in Table. 3.4.

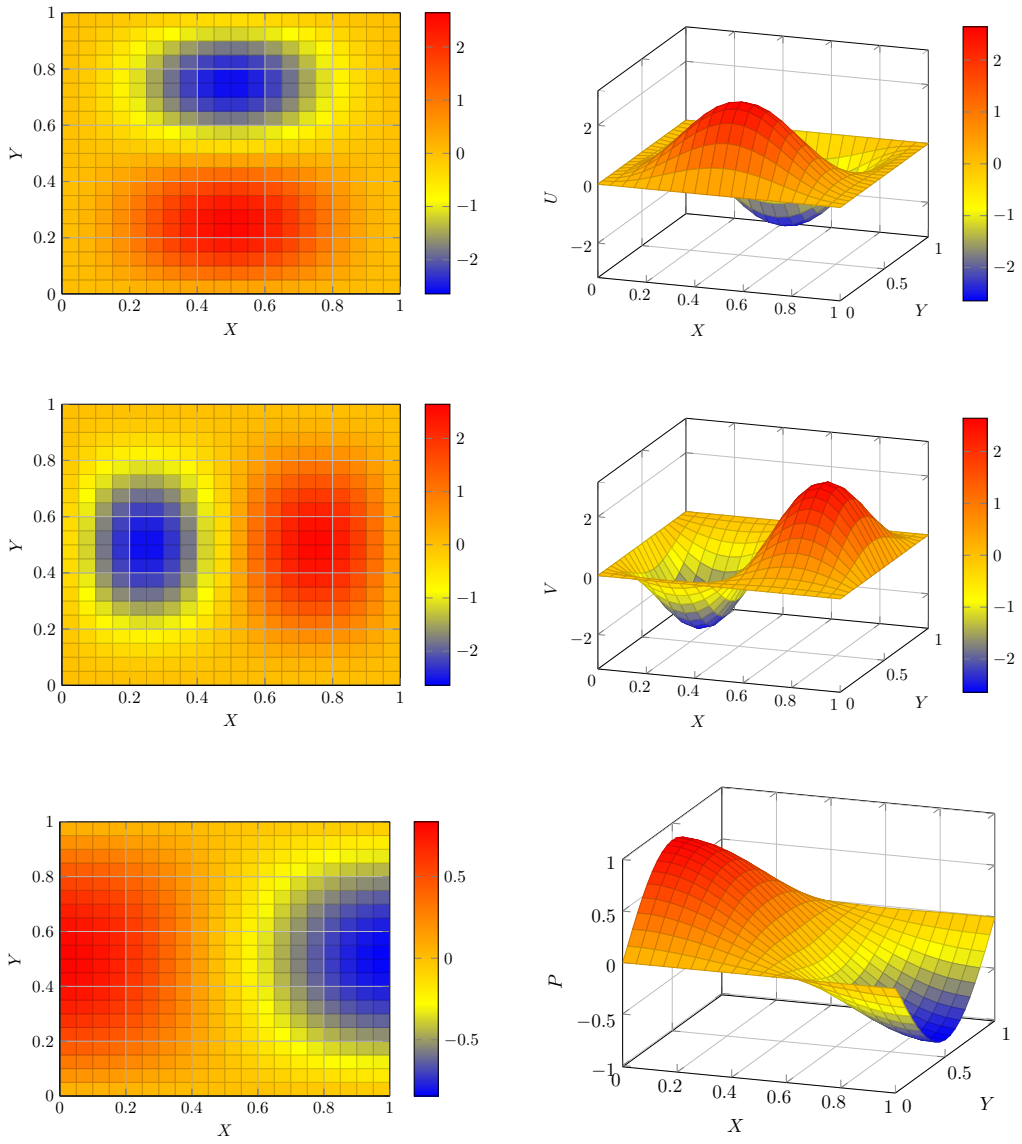


Figure 3.11: Numerical solution of trumped-up transient navier-stokes equation at time = 1

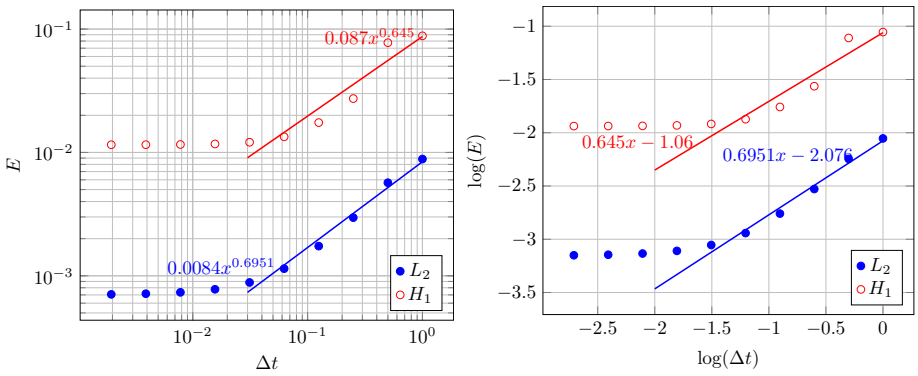


Figure 3.12: Temporal convergence of solution of the transient navier-stokes equation for x direction velocities

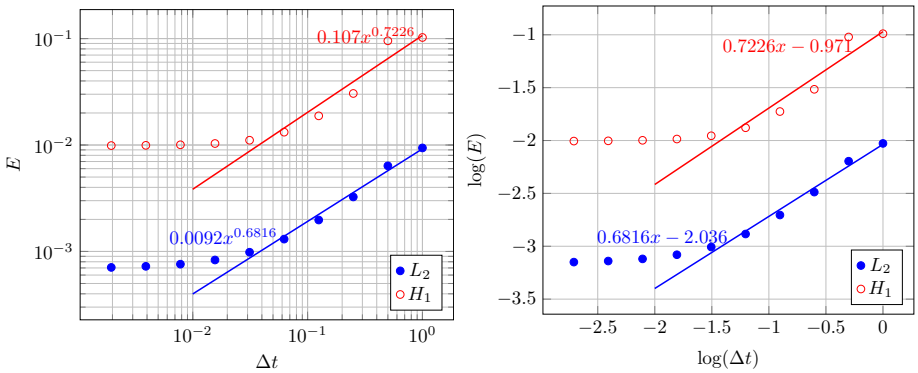


Figure 3.13: Temporal convergence of solution of the transient navier-stokes equation for y direction velocities

Table 3.3: TemporalConvergence of transient navier stokes equation

	X Velocity		Y Velocity	
Δt	L_2	H_1	L_2	H_1
0.5	0.005683	0.077591	0.006376	0.095318
0.25	0.002962	0.02734	0.003252	0.030505
0.125	0.001741	0.017461	0.001975	0.018798
0.0625	0.001143	0.013397	0.001306	0.013205
0.03125	0.000883	0.0121	0.000984	0.01109
0.015625	0.000778	0.011716	0.000831	0.010331
0.007813	0.000734	0.011599	0.000759	0.010047
0.003906	0.000716	0.011559	0.000725	0.009931
0.001953	0.000707	0.011545	0.000708	0.00988

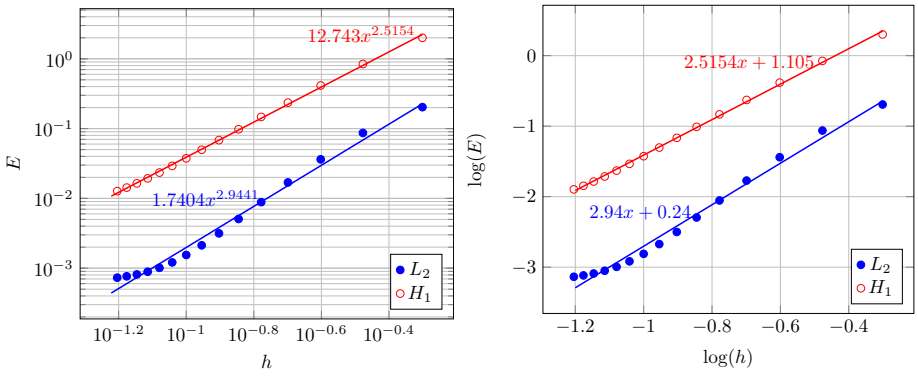


Figure 3.14: Spatial convergence of solution of the transient stokes equation for x direction velocities

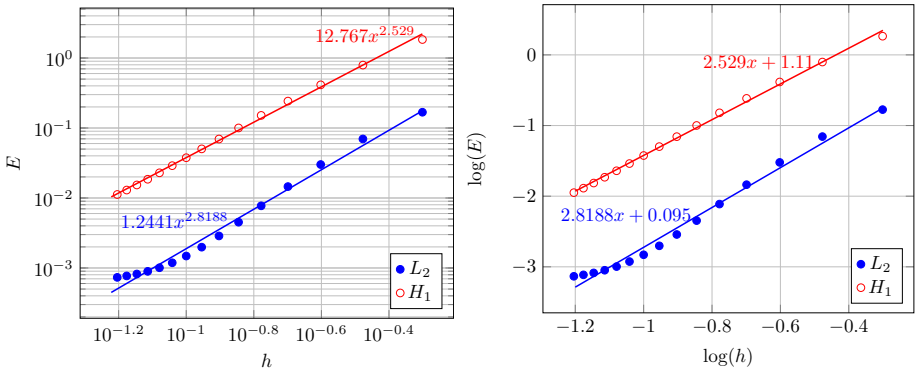


Figure 3.15: Spatial convergence of solution of the transient navier stokes equation for y direction velocities

Table 3.4: Spatial convergence of transient navier-stokes equations solver

	X Velocity		Y Velocity	
h	L_2	H_1	L_2	H_1
0.5	0.202907	1.999232	0.168005	1.838788
0.333333	0.086643	0.842564	0.069752	0.792218
0.25	0.036257	0.414217	0.030073	0.413751
0.2	0.016918	0.235443	0.01455	0.242232
0.166667	0.008833	0.146999	0.007731	0.151323
0.142857	0.005066	0.097781	0.004509	0.100197
0.125	0.003155	0.068316	0.002869	0.069609
0.111111	0.002125	0.049722	0.001983	0.05032
0.1	0.001545	0.037499	0.001481	0.037621
0.090909	0.001209	0.029203	0.001187	0.028963
0.083333	0.00101	0.023431	0.001008	0.02289
0.076923	0.000889	0.019339	0.000895	0.018533
0.071429	0.000813	0.016398	0.000821	0.015349
0.066667	0.000764	0.014261	0.00077	0.012988
0.0625	0.000731	0.012697	0.000734	0.011218

3.8.1 Results

3.8.1.1 Problem

After validation, it's time to simulate a benchmark problem named 'Lid Driven Cavity'. The geometry of the problem consists of a unit square with homogeneous dirichlet boundary conditions on three walls which is zero and dirichlet boundary condition on the top edge which it's velocity is 1. In order to solve the problem we need to solve stokes equation:

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = \frac{1}{Re}\nabla^2 u - \nabla p \quad \text{in} \quad \Omega \tag{3.57}$$

$$\nabla \cdot u = 0 \quad \text{in} \quad \Omega \tag{3.58}$$

$$u = 1\hat{x} \quad \text{on} \quad \Omega_{top} \tag{3.59}$$

$$u = 0\hat{x} \quad \text{on} \quad \Omega \backslash \Omega_{top} \tag{3.60}$$

$$\tag{3.61}$$

3.8.1.2 Stability analysis

As discussed earlier, we used Euler Backward for temporal descritization and AB-3 for convection descritization. In order to have stable results we should choose a time step that satisfies the following condition:

$$dt \leq 0.723 \min\left(\frac{h}{u}\right) \tag{3.62}$$

since in this problem maximum velocity equals one and occures at the top of the cavity we can write:

$$dt \leq \frac{0.723}{N} \tag{3.63}$$

where N is the grid number.

3.8.1.3 Simulated Results

Simulation of the lid driven cavity is illustrated in Fig. 3.16 for $Re = 200$ and $N = 20$.

Simulations for $Re = 1000$ and $N = 20$ and $dt = 0.005$ also is illustrated in Fig. 3.17 As it can be seen in this figure, the velocity vector magnitude at the upper right hand side of the domain becomes greater than one which suggests that the solution starts to become inaccurate.

3.8.1.4 Steady state analysis

In this section we will consider steady state analysis of the problem. The solution will be considered steady state when the following situation occures.

$$\| u^n - u^{n-1} \|_{L_\infty} \leq \epsilon \tag{3.64}$$

steady state condition will be reached in different iteration numbers based on the value of the ϵ. It means for smaller values of ϵ higher iteration number is required and vice versa. Number of iterations for reaching the steady state condistions for different node numbers is illustrated in Fig. 3.18. The cavity problem in steady state condition for $Re = 200$, $N = 20$, $\Delta t = 0.01$ is also shown in Fig. 3.19. The steady state condition for this problem and for $\epsilon = 10^{-5}$ is reached after 1525 iteration.

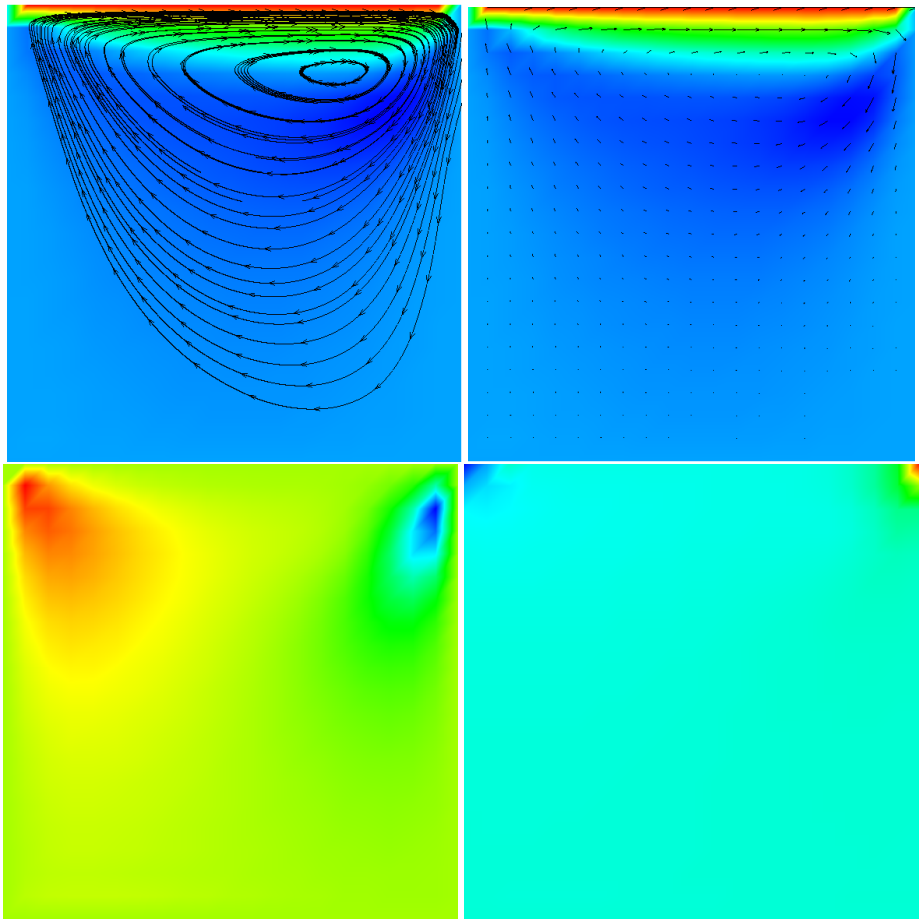


Figure 3.16: Results of lid driven cavity for $Re = 200$ and $N = 20$ and $dt = 0.01$

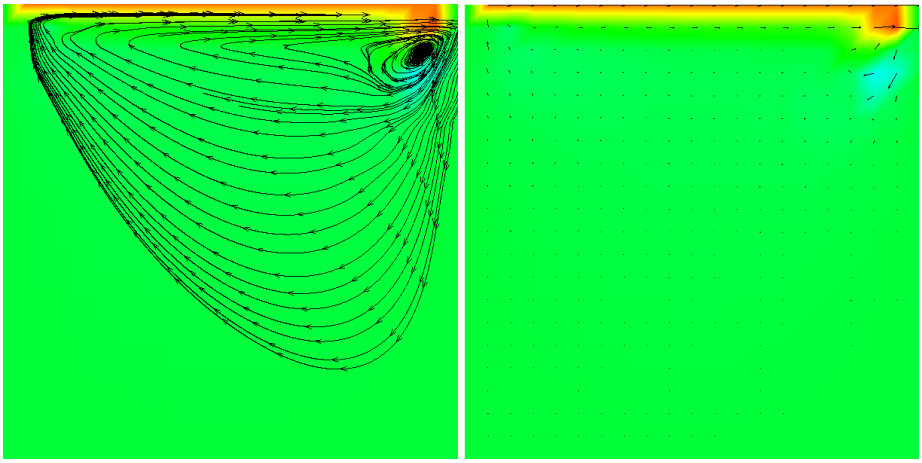


Figure 3.17: Results of lid driven cavity for $Re = 1000$ and $N = 20$ and $dt = 0.002$

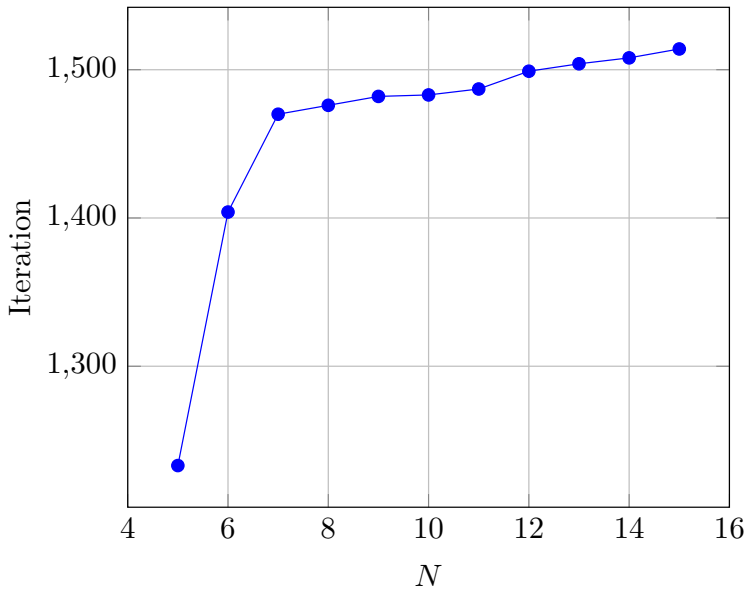


Figure 3.18: Number of iterations for reaching staty state for $\epsilon = 10^{-5}$

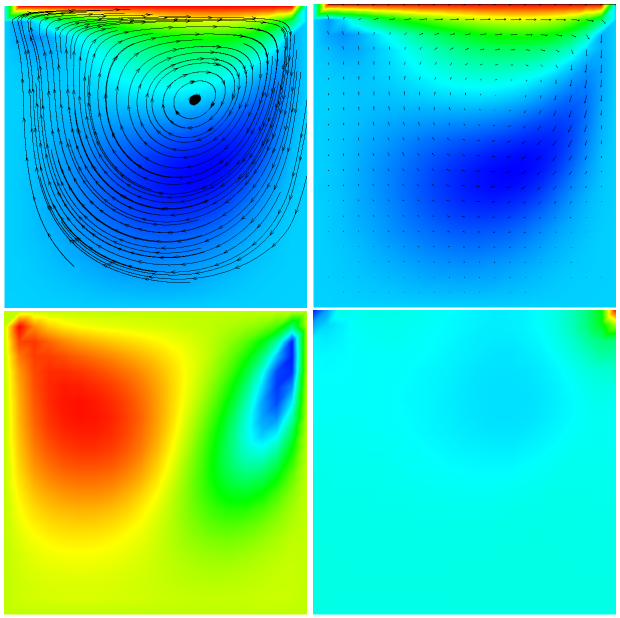


Figure 3.19: Steady state solution of cavity problem for $Re = 200$, $N = 20$, $\Delta t = 0.01$ after 1525 iteration for $\epsilon = 10^{-5}$