Finite Element in Fluids
Assignment 1

Anil Bettadahalli Channakeshava

22/05/2016

Introduction:

This report contains the solutions of 3 exercises of given assignment related to topics like Steady transport problems (1D and 2D convection-diffusion-reaction), 2D Unsteady transport and Incompressible Stokes/Navier Stokes problems. And All the exercises are solved considering given domain Ω=(0,2)×(0,3) and given boundary set as mentioned in question.

Exercise 1: Steady Transport Problems

(a) 1D Convection-diffusion-reaction :

We have the Convection-diffusion-reaction neglecting transient term defined by,

$$\mathbf{a} \cdot \nabla u - \nabla \cdot (\nu \nabla u) - \sigma u = s, \quad \text{in} \quad \Omega \tag{1}$$

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

$$\mathbf{n} \cdot \nu \nabla = \nu \frac{\partial u}{\partial \mathbf{n}} = h \quad \text{on} \quad \Gamma_N$$

The weak form of the equation(1) is obtained by using weighted residual method ,

$$\int_\Omega w(\mathbf{a} \cdot \nabla u) d\Omega - \int_\Omega w \nabla \cdot (\nu \nabla u) d\Omega + \int_\Omega \sigma w u d\Omega = \int_{\Gamma_N} w h d\Gamma \int_\Omega w s d\Omega \quad \forall w \in \mathcal{V}$$

Using divergence theorem on the diffusion term or integrating by parts and noting that w=0 on Γ_D we obtain weak form as,

$$\boxed{\int_\Omega w(\mathbf{a} \cdot \nabla u) d\Omega + \int_\Omega \nabla w \cdot (\nu \nabla u) d\Omega + \int_\Omega \sigma w u d\Omega = \int_{\Gamma_N} w h d\Gamma + \int_\Omega w s d\Omega \quad \forall w \in \mathcal{V}}$$

Now discretizing this weak form using Galerkin's method and an approximation of the solution,

$$w_i \cong N_i$$

$$u^h(x) \cong \sum_j u_j N_j(x)$$

$$\sum_j \int_\Omega N_i(\mathbf{a} \cdot \frac{\partial N_j}{\partial x} u_j) d\Omega + \int_\Omega \frac{\partial N_i}{\partial x} \cdot (\nu \frac{\partial N_j}{\partial x} u_j) d\Omega + \int_\Omega \sigma N_i N_j u_j d\Omega = \int_{\Gamma_N} N_i h d\Gamma + \int_\Omega N_i s d\Omega$$

$$\sum_j \int_\Omega [N_i(\mathbf{a} \cdot \frac{\partial N_j}{\partial x})d\Omega + \frac{\partial N_i}{\partial x} \cdot (\nu \frac{\partial N_j}{\partial x})d\Omega + \sigma N_i N_j d\Omega]u_j = \int_{\Gamma_N} N_i h d\Gamma + \int_\Omega N_i s d\Omega$$

Hence we obtain a system of equations as,

$$[\mathbf{K} + \mathbf{C} + \mathbf{M}]\mathbf{u} = \mathbf{f}$$

$$\mathbf{K} = \int_\Omega N_i(\mathbf{a} \cdot \frac{\partial N_j}{\partial x})d\Omega \quad (Convection \quad Matrix)$$

$$\mathbf{C} = \int_\Omega \frac{\partial N_i}{\partial x} \cdot (\nu \frac{\partial N_j}{\partial x})d\Omega \quad (Diffusion \quad Matrix)$$

$$\mathbf{M} = \int_\Omega \sigma N_i N_j d\Omega \quad (Reaction \quad Matrix)$$

$$\mathbf{f} = \int_{\Gamma_N} N_i h d\Gamma + \int_\Omega N_i s d\Omega \quad (Force \quad Vector)$$

Note: Here I have considered my unknown solution as 'u' and solved part (a). Replace u=ρ as ρ is unknown term in question.

(b) 2D Convection-diffusion-reaction (Galerkin Formulation) :

As per the given question, I have modified the Matlab code to solve this problem with linear elements, a spatial discretization h=0.2, convective velocity, diffusion parameter, reaction and source term. Modified code is attached in zip file (**'exe1'**). I have assigned different cases in subroutine '**problemtype.m**'. Run this subroutine and we get plots according to the different cases we need. Using *'Galerkin formulation'* performed analysis for different cases with given BC: $\rho = 1$ in Γ_2, $\rho = 0$ in Γ_4 and obtained plots as shown below.

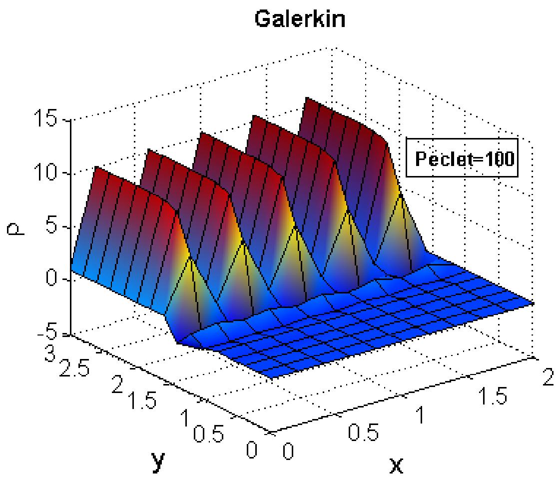

Figure 1: Case.1:($a = 1$,$\nu = 10^{-3}$,$\sigma = 10^{-3}$,$s = 0$)

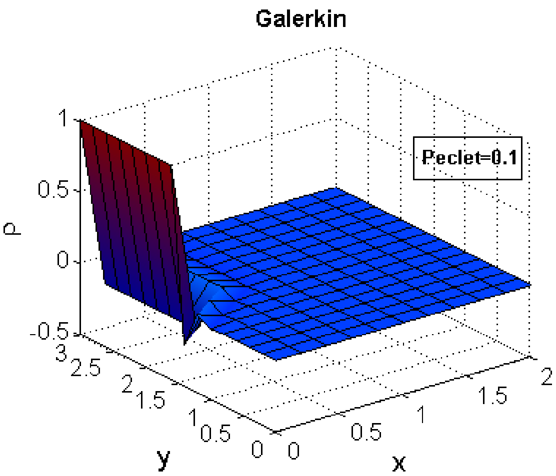

Figure 2: Case.2:($a = 10^{-3}$,$\nu = 10^{-3}$,$\sigma = 1$,$s = 0$)

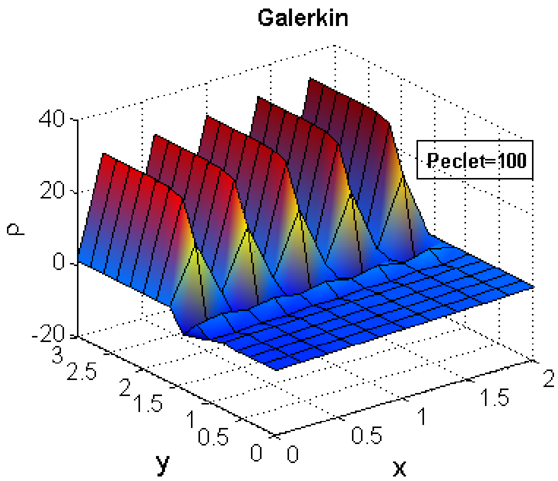

Figure 3: Case.3:($a = 1, \nu = 10^{-3}, \sigma = 0, s = 1$)

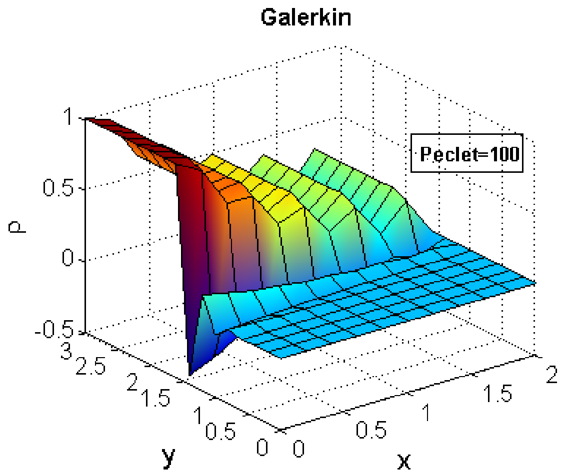

Figure 4: Case.4:($a = 1, \nu = 10^{-3}, \sigma = 1, s = 0$)

As we can see from above *Figures* 1-4, the obtained numerical solutions from Galerkin formulation for different cases , for case 3 (*Figure* 2) we are obtaining Peclet Number, Pe=0.1 which is < 1. That means the obtained solution is stable and is acceptable. And also we can observe that in this case convective(a) and diffusion term(ν) is very less, in other way we can say it is reaction dominated. Wherein for cases 1,2 and 4 we obtain Peclet Number, Pe=100 which is very high which makes solution unstable and give oscillations. And also we can observe that convective term a=1 and is dominating diffusion operator(ν) in cases 1 and 3 which makes Galerkin method to lose its approximation property and is not acceptable. But solution obtained using case 4 is relatively acceptable as reaction($\rho = 1$). Hence we can overcome this problem by just reducing mesh size,(h) keeping other parameters constant so that we can obtain low 'Pe' and reduce oscillations to obtain good approximation. I have reduced 'h' from 0.2 to 0.05 and we can see the reduction in oscillations and better solutions for case 1 which is shown in *Figure* 5 below. Similarly we obtain better solutions for case 3 and 4 by reducing 'h'.

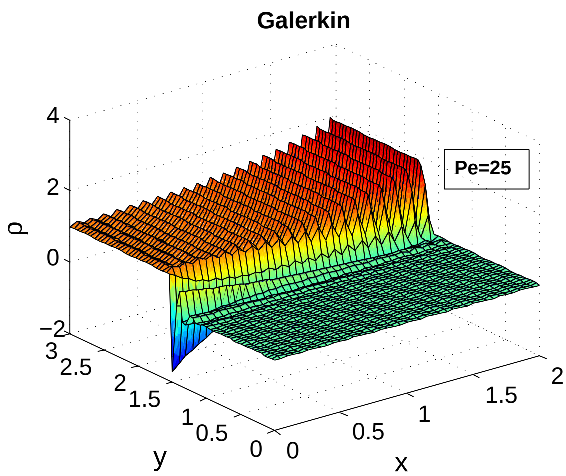

Figure 5: Case 1 with h=0.05

(c) SUPG and GLS formulation :

Here I am choosing the case 3: $a = 1, \nu = 10^{-3}, \sigma = 0, s = 1$ with mesh size, h=0.2 and obtaining the solution using SUPG and GLS formulation. As you can see, the case which I have chosen is convection-diffusion dominated (no reaction). As per practical point of view both methods (SUPG and GLS) are identical for zero reaction with linear elements. And as per the numerical solution obtained using above case mentioned with linear elements we obtain following plots shown in *Figures* 6 and 7, which shows solution obtained using SUPG and GLS formulation. We can see from *Figures* 6 and 7 that we obtained almost same plots (same solutions) for both SUPG and GLS for the case I have chosen, from which we can say that our assumptions and implementation of code is working properly.

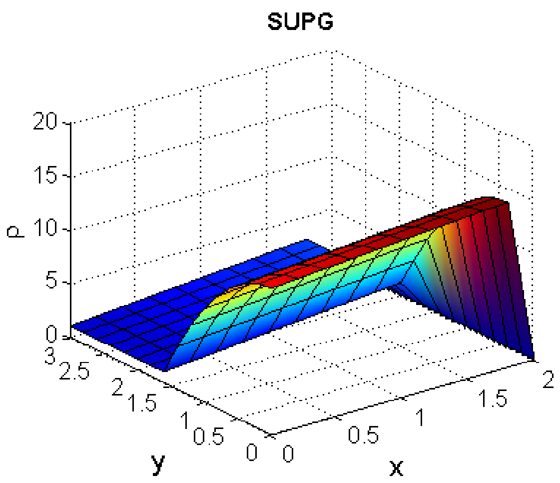

Figure 6: SUPG with case 3

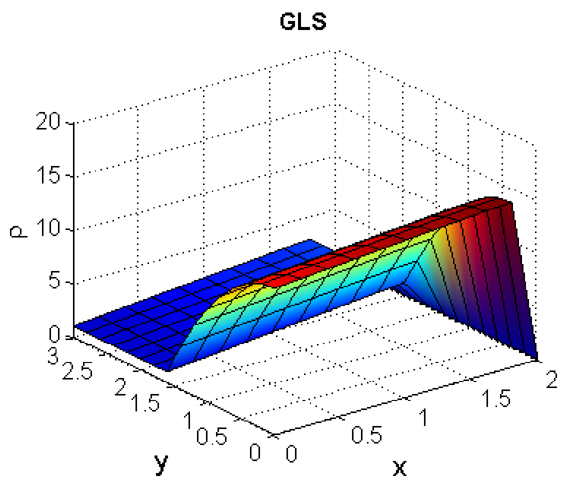

Figure 7: GLS with case 3

Here I am explaining how each methods (SUPG and GLS) is obtained and added with standard galerkin in order to increase stabilization. We know that the residual of convection-diffusion-reaction equation is,

$$R(\rho) = \mathbf{a} \cdot \nabla \rho - \nabla \cdot (\nu \nabla \rho) - \sigma \rho - s = L(\rho) - s \tag{2}$$

Where L is the differential operator associated with differential equation. And the general form of consistent formulation (with added stabilization term) is,

$$\mathbf{a}(w, \rho) + \mathbf{c}(\mathbf{a}; w, \rho) + (w, \sigma\rho) + \sum_e \int_{\Omega^e} P(w)\tau R(\rho) d\Omega = (w, s) + (w, h)_{\Gamma_N} \tag{3}$$

Here, the stabilization term is '$\sum_e \int_{\Omega^e} P(w)\tau R(\rho) d\Omega$' where, $P(w)$ is a perturbation operator applied to test function and τ is stabilization parameter.

SUPG Method:

Here we define stabilization technique by taking, $P(w) = \mathbf{a} \cdot \nabla w$. Hence after substituting $P(w)$ in equation(3) with approximation $\rho \cong \rho^h$ we obtain,

$$\mathbf{a}(w^h, \rho^h) + \mathbf{c}(\mathbf{a}; w^h, \rho^h) + (w^h, \sigma\rho^h) + \sum_e \int_{\Omega^e} (\mathbf{a} \cdot \nabla w^h)\tau[\mathbf{a} \cdot \nabla \rho^h - \nabla \cdot (\nu \nabla \rho^h) + \sigma\rho^h - s] d\Omega = (w^h, s) + (w^h, h)_{\Gamma_N} \tag{4}$$

I have implemented the same expressions with system of equations '**Ku=f**' as shown in equation(4) in the matlab code under subroutine '`FEM_system`' and I have considered stabilization parameter, $\tau = \frac{h}{2a}\left(1 + \frac{9}{Pe^2} + \left(\frac{h}{2a}\sigma\right)\right)^{\frac{-1}{2}}$, where $Pe = \frac{ah}{2\nu}$. Implementation of SUPG formulation in matlab subroutine is shown below.

```
%%% SUPG Method %%%
Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny)+ N_ig'*sigma*N_ig + ...
    tau*(ax*Nx+ay*Ny)'*(ax*Nx+ay*Ny + N_ig*sigma))*dvolu;
aux = N_ig*Xe;
f_ig = SourceTerm(aux,source);
fe = fe + (N_ig+tau*(ax*Nx+ay*Ny))'*(f_ig*dvolu);
```

GLS Method:

Here the stabilization term is an element-by-element weighted least-squares formulation of original differential equation i.e, $P(w) = L(w) = \mathbf{a} \cdot \nabla w - \nabla \cdot (\nu \nabla w) + \sigma w$. Hence after substituting $P(w)$ in equation(3) with approximation $\rho \cong \rho^h$ we obtain,

$$\mathbf{a}(w^h, \rho^h) + \mathbf{c}(\mathbf{a}; w^h, \rho^h) + (w^h, \sigma\rho^h) + \sum_e \int_{\Omega^e} [\mathbf{a}\cdot\nabla w^h - \nabla\cdot(\nu\nabla w^h) + \sigma w^h]\tau[\mathbf{a}\cdot\nabla\rho^h + \nabla\cdot(\nu\nabla\rho^h) + \sigma\rho^h - s]d\Omega =$$

$$= (w^h, s) + (w^h, h)_{\Gamma_N}$$

I have implemented the same expressions with system of equations '**Ku=f**' as shown in above equation in the matlab code under subroutine '`FEM_system`' and I have considered stabilization parameter, $\tau = \frac{h}{2a}\left(1 + \frac{9}{Pe^2} + \left(\frac{h}{2a}\sigma\right)\right)^{\frac{-1}{2}}$, where $Pe = \frac{ah}{2\nu}$. Refer attached Zip file (**'exe1'**) for modified subroutines in matlab. Implementation of GLS formulation in matlab subroutine is shown below.

```
%%%% GLS Method %%%
Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) +N_ig'*(ax*Nx+ay*Ny)+ N_ig'*sigma*N_ig + ...
    tau*(ax*Nx+ay*Ny+sigma*N_ig)'*(ax*Nx+ay*Ny + N_ig*sigma))*dvolu;
aux = N_ig*Xe;
f_ig = SourceTerm(aux,source);
fe = fe + (N_ig+tau*(ax*Nx+ay*Ny+sigma*N_ig))'*(f_ig*dvolu);
```

(d) Modified Boundary conditions in Direchlit and Neumann :

It was asked to solve the problem with Boundary conditions ρ=2 in Γ_2 and ρ=1 in Γ_4. And also to modify BC in Γ_4 to impose Neumann BC so that we should obtain same solution as previous. For this purpose I have defined two more BC's in subroutine '**main.m**' (refer zip-file **'exe1'**) : one with Direchlit BC's on both Γ_2 with ρ=2 and Γ_4 with ρ=1 and the other BC's with Direchlit BC Γ_2 with ρ=2 and Neumann BC on Γ_4.

It is to be noted that here I have considered SUPG method with 'case 1' and h=0.2 for both BC's and compared both solution. In the zip-file (**'exe1'**) refer excel document named '`exe1_partd`', where I have shown the solutions obtained using both BC's and compared. Initially we obtain unknown solution(ρ) and flux(q) imposing Direchlit BC's on both Γ_2 and Γ_4 by running subroutine '**problemtype.m**' selecting case 1. And in next step imposing Direchlit BC on Γ_2 ,Neumann BC on Γ_4 and using the obtained nodal flux from previous Direchlit BC as input for new nodal flux on Γ_4(where Neumann BC applied now), we obtain the new unknown solution (ρ_{new}) and are compared. The obtained error is very less and it is proved that almost same solutions are obtained using both BC's. Similarly we obtain same solutions for Galerkin and GLS methods imposing both Direchlit and Neumann on Γ_4 following same steps as we did for SUPG .

Exercise 2: Unsteady Transport problems

In this exercise I have included the transient term. Considered and implemented the given initial conditions and Boundary Conditions(Direchlit), convection vector field as mentioned in the question. I have attached the zip file named '(**exe2**)' where we can find the matlab codes with modified subroutines.

From exercise 1, I have found SUPG method is more appropriate compared to other methods for space discretization since it showed stable results for all cases. Hence I am using SUPG for space discretization here with CrankNicolson for *Convective-diffusion-reaction problem.* But I have used Galerkin for space discretization and TG3 2-step only for *Pure convection problem(diffusion and reaction not included).* Hence it is to be noted that here I am showing results of ***SUPG+CrankNicolson (for Convection-diffusion-reaction problem)*** and ***Galerkin+TG3 2-step (for Pure Convection problem)*** and also here I am considering unknown 'ρ=u' and solving the problem.

(a) 1. SUPG+CrankNicolson :

As I mentioned above for convection-diffusion-reaction problem, I have implemented SUPG for space discretization and CrankNicolson for time discretization. The mathematical steps of implementation of above methods are shown below,

We know that transient convection-diffusion-reaction as,

$$u_t + \mathbf{a} \cdot \nabla u - \nabla \cdot (\nu \nabla u) - \sigma u = s, \quad \text{in} \quad \Omega \tag{5}$$

After time discretization using the θ family method we obtain,

$$\frac{\Delta u}{\Delta t} + \theta[\mathbf{a} \cdot \nabla - \nabla \cdot (\nu \nabla) - \sigma]\Delta u = \theta s^{n+1} + (1-\theta)s^n - [\mathbf{a} \cdot \nabla - \nabla \cdot (\nu \nabla) - \sigma]u^n \tag{6}$$

For CrankNicolson, $\theta = \frac{1}{2}$ and source, $s = 0$. Considering this conditions and after integrating over the computational domain and imposing the boundary conditions, we obtain

$$(w, \frac{\Delta u}{\Delta t}) + \frac{1}{2}\Big[(w, \Delta u; \mathbf{a} \cdot \hat{\mathbf{n}})_\Gamma - (\Delta u, \mathbf{a} \cdot \nabla w)_\Omega + (\nabla w . \nu \nabla(\Delta u))_\Omega + (w, \sigma \Delta u)_\Omega\Big] = -\Big[(w, u^n(\mathbf{a} \cdot \hat{\mathbf{n}})_\Gamma \tag{7}$$

$$-(u^n, \mathbf{a} \cdot \nabla w)_\Omega + (\nabla w, \nu \nabla u^n)_\Omega + (w, \sigma u^n)_\Omega\Big]$$

Or we can also express as,

$$\frac{1}{\Delta t}\mathbf{M} + \frac{1}{2}\Big[\mathbf{M}_{out} - \mathbf{C} + \mathbf{K}_{diff} + \mathbf{M}_{rn}\Big]\Delta u \cdot \Delta t = (-\mathbf{M}_{out} + \mathbf{C}^T - \mathbf{K}_{diff} - \mathbf{M}_{rn})\Delta t u^n \tag{8}$$

We know that the added stabilization term($\mathbf{S}$) with equation(7) consists,

$$\mathbf{S} = \sum_e \Big(P(\mathbf{w})\tau, R(\Delta \mathbf{u})\Big)_{\Omega^e} \tag{9}$$

Now since have to implement SUPG stabilization technique, we have $P(\mathbf{w}) = \mathbf{W}(\mathbf{a} \cdot \nabla)\mathbf{w}$, where $\mathbf{W} = \frac{1}{2}$ for CrankNicolson(1step method). Now substituting $P(\mathbf{w})$ in equation(9) and multiplying with the residual ($R(\Delta \mathbf{u})$ we obtain,

$$\mathbf{S} = \frac{1}{2}(\nabla \cdot (\mathbf{aw}))\tau\Big[\frac{\Delta \mathbf{u}}{\Delta t} + \frac{1}{2}\big(\nabla \cdot (\mathbf{au}) + \sigma \mathbf{u}\big) + \big(\nabla \cdot (\mathbf{au}) + \sigma \mathbf{u}\big)\Big]$$

$$\nabla \cdot (\nu \nabla \mathbf{u}) = 0 \quad (Linear)$$

$$w_i \cong N_i, u \cong u^h = \sum N_j u_j$$

$$\implies \mathbf{S} = \frac{1}{2}(\nabla \cdot (\mathbf{a}N_i))\tau\Big[N_j + \frac{\Delta t}{2}(\nabla \cdot (\mathbf{a}N_j) + \sigma N_j)\Big]\frac{\Delta u_j}{\Delta t} + \frac{1}{2}(\nabla \cdot (\mathbf{a}N_i))\tau\Big[(\nabla \cdot (\mathbf{a}N_j) + \sigma N_j)\Big]u_j$$

$$\mathbf{S} = \frac{1}{2}\Big[(\nabla \cdot \mathbf{a})N_i + \mathbf{a} \cdot (\nabla N_i)\Big]\tau\Bigg[N_j + \frac{\Delta t}{2}\Big[(\nabla \cdot \mathbf{a})N_j + \mathbf{a} \cdot (\nabla N_j) + \sigma N_j\Big]\Bigg]\frac{\Delta u_j}{\Delta t} + \frac{1}{2}\Big[(\nabla \cdot \mathbf{a})N_i + \mathbf{a} \cdot (\nabla N_i)\Big]$$

$$\tau\Bigg[(\nabla \cdot \mathbf{a})N_j + \mathbf{a} \cdot (\nabla N_j) + \sigma N_j\Bigg]u_j$$

After collecting the identical terms from above equations we divide $\mathbf{S}$ as ,

$$\Big[\mathbf{S}_t + \frac{\Delta t}{2}\mathbf{S}_x\Big]\frac{\Delta u}{\Delta t} = -\mathbf{S}_x \cdot u^n$$

Where,

$$\mathbf{S}_t = \frac{1}{2}\tau\Big[(\nabla \cdot \mathbf{a})N_i + \mathbf{a} \cdot (\nabla N_i)\Big]N_j$$

$$\mathbf{S}_x = \frac{1}{2}\tau\Big[(\nabla \cdot \mathbf{a})N_j + \mathbf{a} \cdot (\nabla N_j) + \sigma N_j\Big]$$

$$\tau = \Big(\frac{1}{\theta \Delta t} + \frac{2a}{h} + \frac{4\nu}{h^2} + \sigma\Big)^{-1}$$

Finally the stabilization terms are added with residuals in the equation(8) with the system of equations which has to be solved to obtain solution. In the attached zip file '(**exe2**)', we can see the modified subroutines. The below lines shows the implementation of this method in matlab subroutines:

In 'FEM_matrices':

```
elseif sp_meth ==1 %%%SUPG+CrankNicolson%%%%
tau = 2/dt + 2*sqrt((ax^2+ay^2)/(hx^2+hy^2)) + 4*nu/(hx^2+hy^2) + sigma ;
tau = 1/tau ;
Se_t = Se_t + 1/2*tau*( (ax+ay)*N_ig+ ax*Nx+ay*Ny)'* N_ig *dvolu;
Se_x = Se_x + 1/2*tau*( (ax+ay)*N_ig+ ax*Nx+ay*Ny)' ...
            *((ax+ay)*N_ig+ ax*Nx+ay*Ny + N_ig*sigma)*dvolu;
end
```

In 'system.m':

```
%%% Crank-Nicolson + SUPG %%% (Convection-diffusion-reaction)
 A = M - 1/2*dt*C'+ 1/2*dt*Mout + 1/2*dt*K_diff ...
        + 1/2*dt*M_rn + S_t+ dt/2* S_x ;
 B = dt*C'-dt*Mout - dt*M_rn -dt*K_diff - dt*S_x ;
 methodName = 'CN';
```

(a) 2. Galerkin+TG3 2-step :

As I mentioned above for Pure convection problem with no source, I have implemented Galerkin for space discretization and TG3 2-step for time discretization. The mathematical steps of implementation of above methods are shown below,

We know that pure convection problem is defined as,

$$u_t + \mathbf{a} \cdot \nabla u = 0, \tag{10}$$

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad \text{in} \quad \Omega \quad \text{at} \quad \text{t=0}$$
$$u = u_D, \quad \Gamma_D^{in} \times]0, T[$$
$$-\mathbf{a}u \cdot \mathbf{n} = h$$

We have the TG3 2-step scheme as,

$$\tilde{u}^n = u^n + \frac{1}{3}\Delta t u_t^n + \alpha \Delta t^2 u_{tt}^n \quad (1^{st} step) \tag{11}$$

$$u^{n+1} = u^n + \Delta t u_t^n + \frac{1}{2}\Delta t^2 \tilde{u}_{tt}^n \quad (2^{nd} step) \tag{12}$$

where,

$$u_t^n = 0 - \nabla \cdot (\mathbf{a}u^n)$$
$$u_{tt}^n = 0 - \nabla \cdot (\mathbf{a}u_t^n)$$

Now after integrating using WRM method we get the weak form of 'u_t^n' and 'u_{tt}^n' as,

$$\int w u_t^n = -\int w \nabla \cdot (\mathbf{a}u^n) = -\int \nabla \cdot (\mathbf{a}u^n w) + \int \nabla w \cdot \mathbf{a}u^n$$
$$\implies \int w u_t^n = -\oint u^n w \mathbf{a} \cdot \hat{\mathbf{n}} d\Gamma + \int u^n \mathbf{a} \cdot \nabla w$$
$$= -\mathbf{M}_{out} + \mathbf{C}$$

$$\int w u_{tt}^n = -\int \nabla \cdot (\mathbf{a}u_t^n) = \int w \nabla \cdot [\mathbf{a} \nabla \cdot (\mathbf{a}u_t^n)] d\Omega$$
$$\implies \int w u_{tt}^n = \int \nabla \cdot [w \mathbf{a} \nabla \cdot (\mathbf{a}u_t^n)] - \int (\nabla w \cdot \mathbf{a}) \nabla \cdot (\mathbf{a}u_t^n)$$
$$\implies \int w u_{tt}^n = \oint [w \mathbf{a} \cdot \hat{n} \nabla \cdot (\mathbf{a}u_t^n)] d\Gamma - \int (\nabla w \cdot \mathbf{a}) \nabla \cdot (\mathbf{a}u_t^n) d\Omega$$
$$= \mathbf{C}_{out} - \mathbf{K}$$

Where $\mathbf{C}$ is convection matrix, $\mathbf{M}_{out}$ and $\mathbf{C}_{out}$ are outflow boundary matrices. Refer subroutine '**main.m**' under zip file '**exe2**' to see the implementation of this step.

Now after substituting the weak form of 'u_t^n' and 'u_{tt}^n' in $1^{st} step$ (equation 11) we obtain $\Delta u = \tilde{u}^n - u^n$ and substituting 'u_t^n' and 'u_{tt}^n' in $2^{nd} step$(equation 12) we obtain system of equations that has to be solved to obtain solution at u^{n+1} with $\alpha = \frac{1}{9}$. The system which has to be solved has been implemented in matlab(refer zip file '**exe2**'). The modified subroutines showing the implementations of 'Galerkin-TG3 2-step' system is shown below:

In 'system.m':

```
case 4 %% TG3 2step +Galerkin %%% (Only for Pure convection)
    alpha = 1/9;
    B = cell(3,1) ;
    A = M ;
    B{1} = dt/3*(C'-Mout)+alpha*dt^2*(-K+Cout');
    B{2} = dt*(C'-Mout) ;
    B{3} = 0.5*dt^2*(-K+Cout');
    methodName = 'TG3' ;
    warning('Diffusion and Reaction not implemented')
```

(b): Refer zip file 'exe2'

(c): Solutions using above methods and tests

Sensitivity analysis of Mesh dependency :

For SUPG+Crank Nicolson method I have considered constant values of reaction and diffusion as σ=1, ν=0.001 and end time t^n=3s and linear elements for the analysis. Performed sensitivity analysis of the mesh dependency and found that good solutions are obtained after refining mesh size (h). Below figures shows the solution for mesh size I have considered for analysis from which we can observe that as we refine mesh size the 'Courant number' is less than 1 which leads to stable solutions, which can be see in below figure.

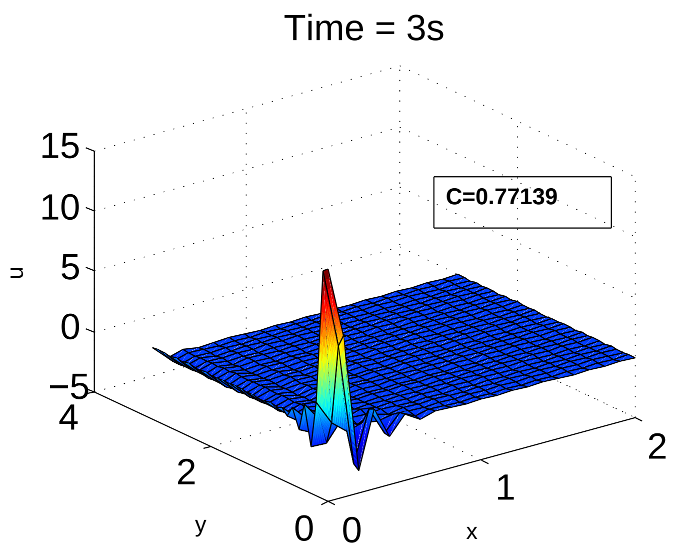

Figure 8: Solution after t=3s,Δt = 110 h=0.1

For Galerkin+TG3 2-step method I have considered constant values of reaction and diffusion as σ=0, ν=0 since it is implemented only for pure convection and end time t^n=1s and linear elements for the analysis. Performed sensitivity analysis of the mesh dependency and found that good solutions are obtained after refining mesh size (h). Below figures shows the solution for mesh size I have considered for analysis from which we can observe that as we refine mesh size the 'Courant number' is less than 1 which leads to stable solutions, which can be see in below figure.

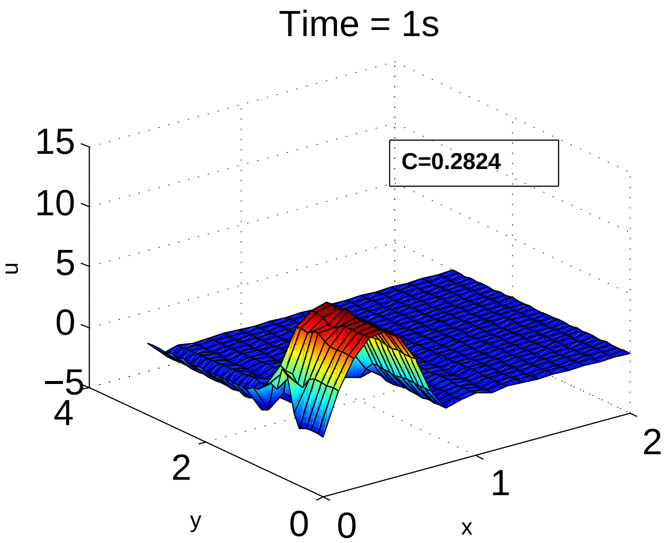

Figure 9: Solution after t=1s,Δt = 100 h=0.1

Analysis on time discretization :

For SUPG+Crank Nicolson method I have considered constant values of reaction and diffusion as σ=1, ν=0.001 and end time t^n=3s and linear elements of size h=0.1 for the analysis. I have considered initial discretization time of Δt= 110 and later changed the time steps to Δt=60 and Δt=300. Below table shows the influence of time discretization after performing the analysis. We can observe that as we increase the time discretization Δt we obtain Courant number lesser which leads to stabilized solutions. Hence to obtain better solutions we need to increase the time discretization values.

Δt(s)	**mesh size (h)**	**End time t^n(s)**	**Courant Number (C)**
110	0.1	3	0.77139
60	0.1	3	1.4142
300	0.1	3	0.28284

Table 1: Influence of time discretization

For Galerkin+TG3 2-step method I have considered constant values of reaction and diffusion as σ=0, ν=0 since it is implemented only for pure convection and end time t^n=2s and linear elements with mesh size h=0.1 for the analysis. I have considered initial discretization time of Δt= 110 and later changed the time steps to Δt=60 and Δt=300. Below table shows the influence of time discretization after performing the analysis. We can observe that as we increase the time discretization Δt we obtain Courant number lesser which leads to stabilized solutions. Hence to obtain better solutions we need to increase the time discretization values.

Δt(s)	**mesh size (h)**	**End time t^n(s)**	**Courant Number (C)**
110	0.1	3	0.51426
60	0.1	3	0.94281
300	0.1	3	0.11886

Table 2: Influence of time discretization

We can observe that for both the methods,(SUPG+Crank Nicolson and Galerkin+TG3 2-step)even though problems are different , influence of time discretization on stability of solution is same. Computational cost is actually reduced using these methods compared to other methods.

(c): Comprison between quadratic and linear elements :

SUPG+Crank Nicolson : I have considered constant values of reaction and diffusion as σ=1, ν=0.001, end time t^n=3s, Δt = 110 for both quadratic and linear elements with mesh size h=0.1. Below figure shows plots quadratic and linear elements. We can observe that there are no much variation in solutions using both the elements.

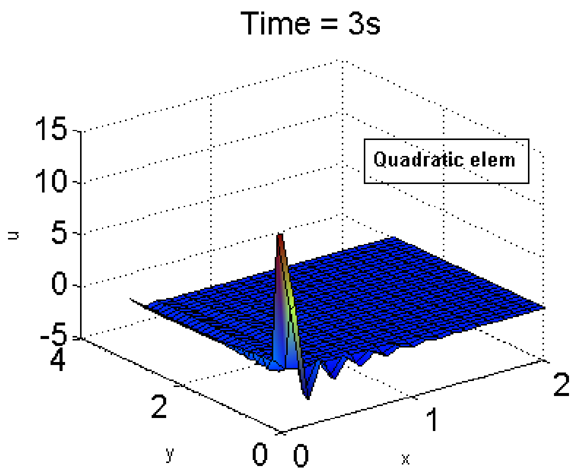

Figure 10: Quadratic elements

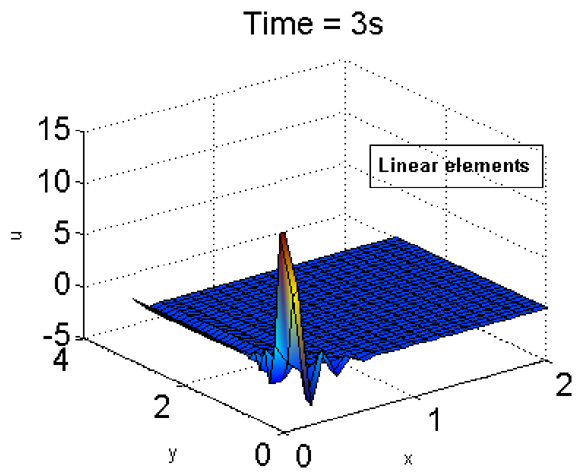

Figure 11: Linear elements

Galerkin+TG3 2-step : I have considered constant values of reaction and diffusion as σ=0, ν=0 (pure convection), end time t^n=1s, Δt = 110 for both quadratic and linear elements with mesh size h=0.1. Below figure shows plots quadratic and linear elements. We can observe that there are no much variation in solutions using both the elements.

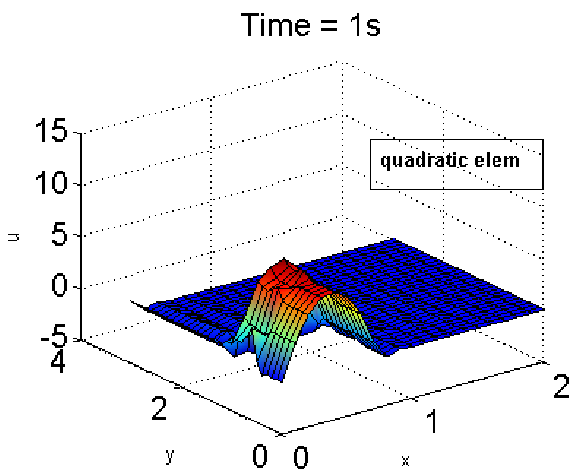

Figure 12: Quadratic elements

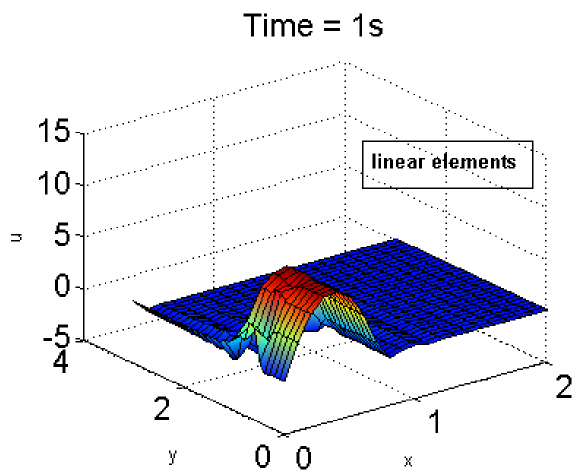

Figure 13: Linear elements

Exercise 3: Stokes and Navier Stokes Problems

(a)Incompressible Viscous Stokes problem :

We have Incompressible viscous stokes equation as,

$$-\nu\nabla^2\mathbf{v} + \nabla p = \mathbf{b} \quad \text{in} \quad \Omega \tag{13}$$
$$\nabla.\mathbf{v} = 0 \quad \text{in} \quad \Omega$$

With given Direchlit boundary conditions as ,

$$\mathbf{v} = 0 \quad \text{in} \quad \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_5$$
$$\mathbf{v}_y = -1 \quad \text{in} \quad \Gamma_4$$

Where ν is Kinematic viscosity parameter and p is kinematic pressure. And the weak form of the equation(5) is obtained by using weighted residual method ,

$$\int_\Omega \mathbf{w}(-\nu\nabla^2\mathbf{v})d\Omega + \int_\Omega \mathbf{w}\nabla p d\Omega = \int_\Omega \mathbf{w}\cdot\mathbf{b}d\Omega \quad \forall\mathbf{w}\in\mathcal{V}$$

By applying divergence theorem and integrating by parts to above equation we obtain weak form as,

$$\int_\Omega \nabla\mathbf{w} : \nu\nabla\mathbf{v} - \int_\Omega p\nabla\cdot\mathbf{w}d\Omega = \int_\Omega \mathbf{w}\cdot\mathbf{b}d\Omega \quad \forall\mathbf{w}\in\mathcal{V}$$
$$\int_\Omega q\nabla\cdot\mathbf{v}d\Omega = 0 \quad \forall q\in\mathcal{Q}$$

We can also express the above weak form as,

$$\begin{cases}\mathbf{a}(\mathbf{w},\mathbf{v}) + b(\mathbf{w},p) = (\mathbf{w},\mathbf{b}) \\ b(\mathbf{v},q) = 0\end{cases}$$
$$\mathcal{V} := \mathcal{H}^1_{\Gamma_D}(\Omega) = \Big\{\mathbf{w}\in\mathcal{H}^1(\Omega)|\mathbf{w} = \mathbf{0} \quad on \quad \Gamma_D\Big\}$$
$$\mathcal{Q} := \Big\{p\in\mathcal{L}_2(\Omega)\Big\}$$

Using Galerkin approximation, approximating $\mathbf{v}\cong\mathbf{v}^h = \mathbf{u}^h + \mathbf{v}^h_D$ where $\mathbf{u}^h\in\mathcal{V}^h$ is auxiliary velocity ,$p\cong p^h$ as well as for their associated weighting functions $\mathbf{w}\cong\mathbf{w}^h$ and $q\cong q^h$, we obtain

$$\begin{cases}\mathbf{a}(\mathbf{w}^h,\mathbf{u}^h) + b(\mathbf{w}^h,p^h) = (\mathbf{w}^h,\mathbf{b}^h) \\ b(\mathbf{u^h},q^h) = 0\end{cases}$$

Where $\mathbf{u}^h$ and p^h are the unknowns which has to be find out.

Fintie element discretization :

From the above equations we have to find the matrix system which governs the discrete Stokes problem assumes the following partitioned form:

$$\begin{bmatrix}\mathbf{K} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{u} \\ \mathbf{p}\end{bmatrix} = \begin{bmatrix}\mathbf{f} \\ \mathbf{h}\end{bmatrix}$$

Where matrix $\mathbf{K}$ is viscosity matrix, results from discretization of $\mathbf{a}(\cdot,\cdot)$, matrix $\mathbf{G}$ is the discrete gradient operator and $\mathbf{G}^T$ is is the discrete divergence operator. Matrix $\mathbf{G}$ arises from the discretization of the term $b(\mathbf{w}^h,p^h)$ in Galerkin variational form. Vectors $\mathbf{f}$ and $\mathbf{h}$ incorporate the effect of the velocity $\mathbf{v}_D$ prescribed on the Dirichlet portion Γ_D of the boundary. And the unknown $\mathbf{u}$ and $\mathbf{p}$ is obtained by solving the above system.

As per the given question applied the Boundary conditions as mentioned and I have considered the viscous fluid and included the viscosity in the matlab program. I have chosen kinematic viscosity ν=0.0001 $\frac{m^2}{s}$ and density of oil as ρ=1000 $\frac{kg}{m^3}$ For the analysis I have considered Taylor hood element($Q2Q1$ element) as it satisfies LBB compatibility condition and also we know that Taylor-Hood elements exhibit optimal quadratic convergence and considered mesh size of h=0.09, as we know mesh size is an important aspect of the quality of the solution. Hence I have chosen the refined mesh to obtain better solutions. After analysis using above parameters following velocity and pressure distributions are obtained as shown below,

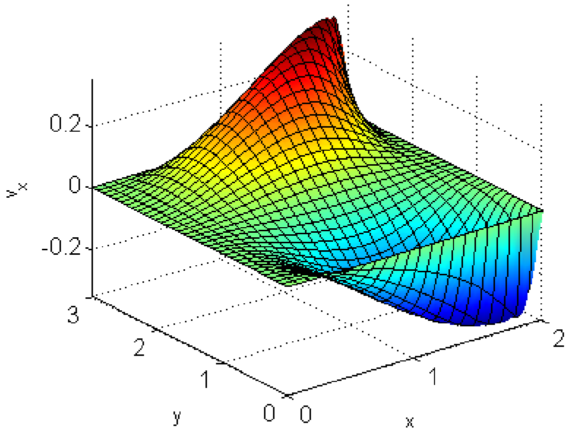

Figure 14: Velocity along x-direction

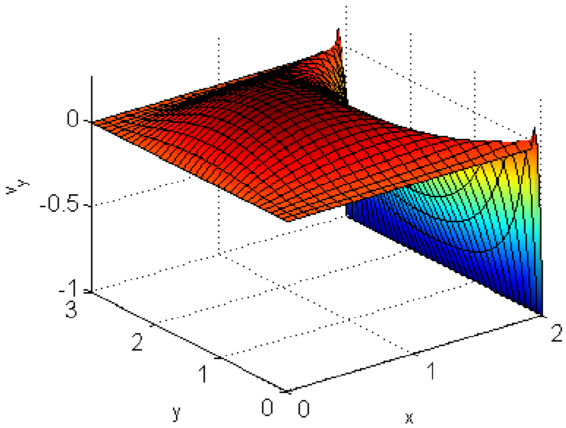

Figure 15: Velocity along y-direction

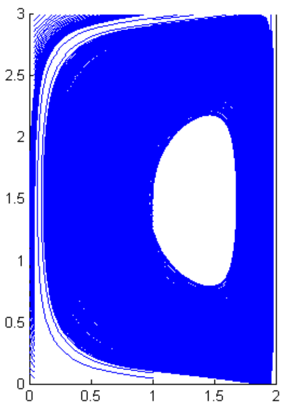

Figure 16: Streamlines

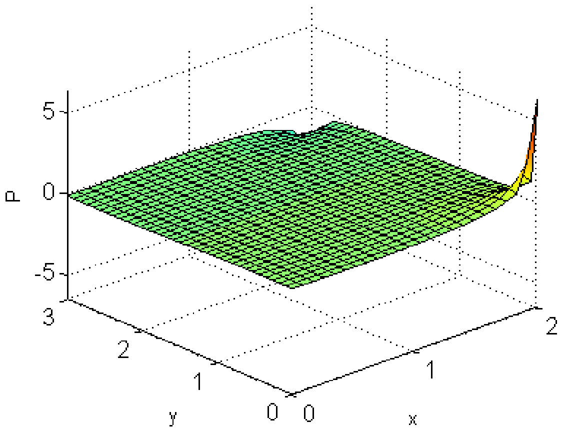

Figure 17: Pressure distribution

As we can see from the above plots as per the given data Dirichlet conditions for the velocity are imposed. Horizontal component of velocity $\mathbf{v}_x$ is prescribed to be 0 on Γ_1,Γ_2,Γ_3, and Γ_5 and Vertical component of velocity $\mathbf{v}_y$=-1 on Γ_4 . As we included the viscosity ,velocity field is not much changed but the pressure field is increased. As the fluid is more viscous pressure field also increases. The matlab code with modified subroutines is attached in zip file ('**exe3**') where we can see the modified boundary conditions and implementation of viscosity parameter.

(a)Incompressible Navier Stokes problem :

As per the given question implemented the boundary conditions as mentioned and considered Galerkin spatial discretization and same type of element as previous, Taylor hood element($Q2Q1$ element) for the analysis of this problem as it satisfies LBB compatibility condition and also we know that Taylor-Hood elements exhibit optimal quadratic convergence . Performed the analysis with different values of Reynolds number, Re=1,100,1000,2000 and obtained results as shown below.

For Re=1 :

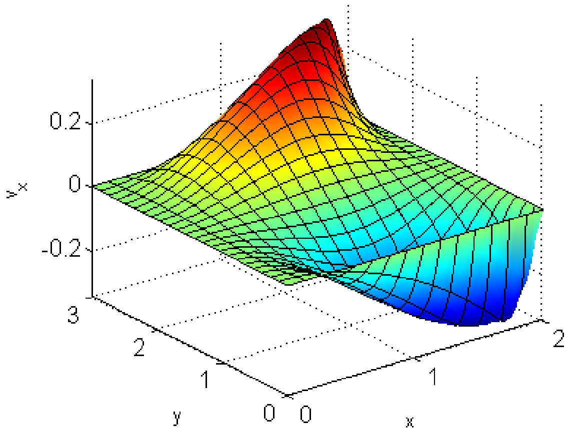

Figure 18: Velocity along x-direction

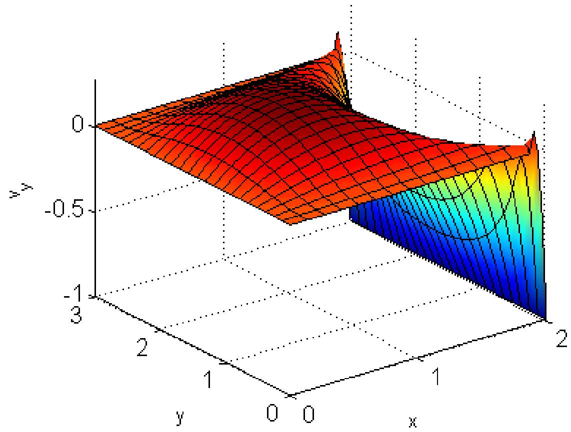

Figure 19: Velocity along y-direction

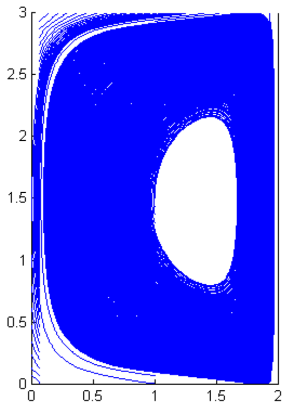

Figure 20: Streamlines

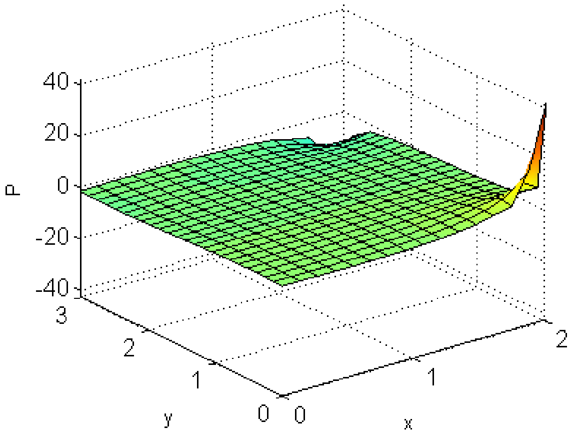

Figure 21: Pressure distribution

For Re=100 :

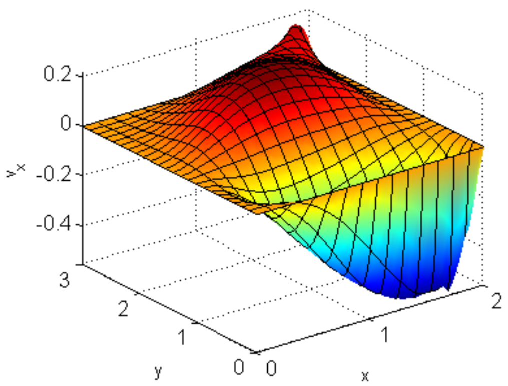

Figure 22: Velocity along x-direction

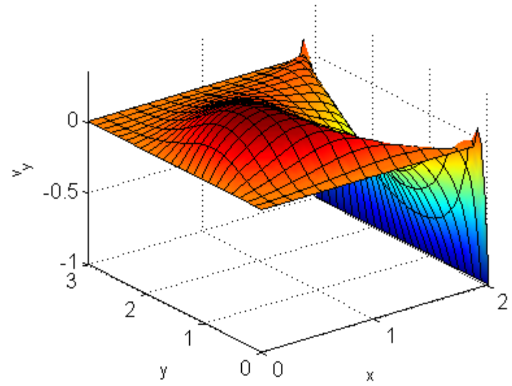

Figure 23: Velocity along y-direction

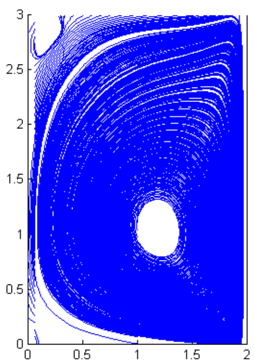

Figure 24: Streamlines

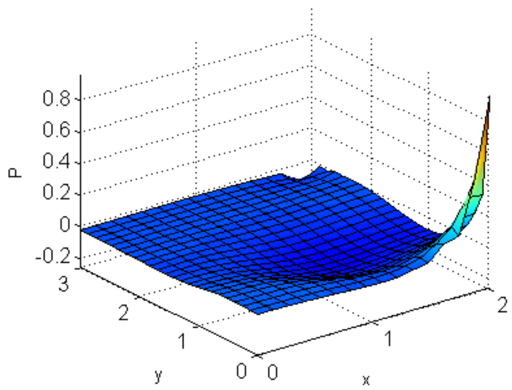

Figure 25: Pressure distribution

For Re=1000 :

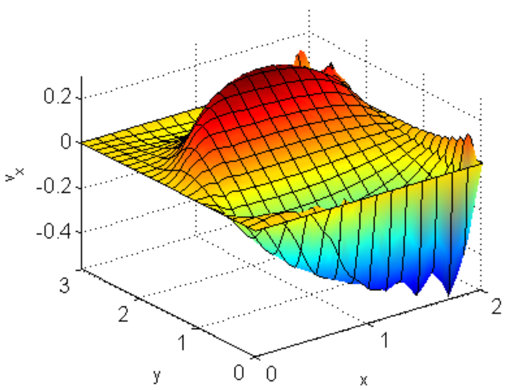

Figure 26: Velocity along x-direction

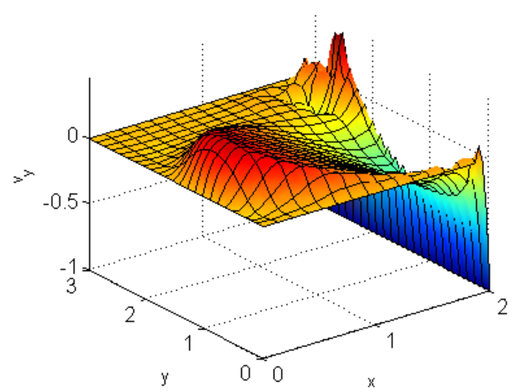

Figure 27: Velocity along y-direction

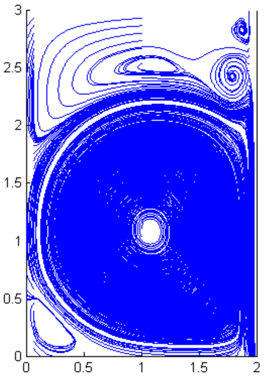

Figure 28: Streamlines

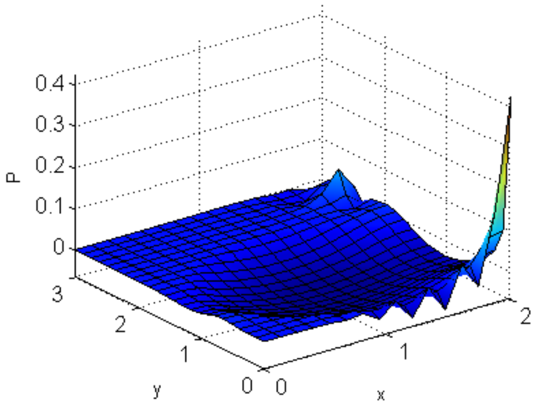

Figure 29: Pressure distribution

For Re=2000 :

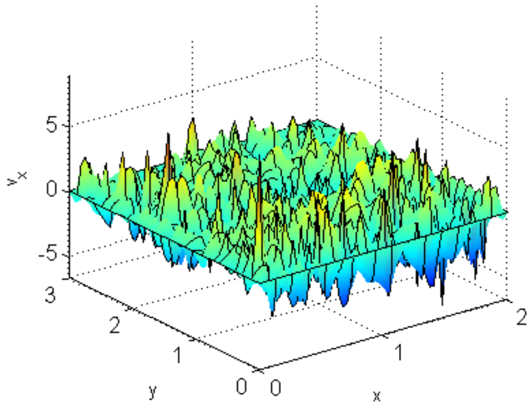

Figure 30: Velocity along x-direction

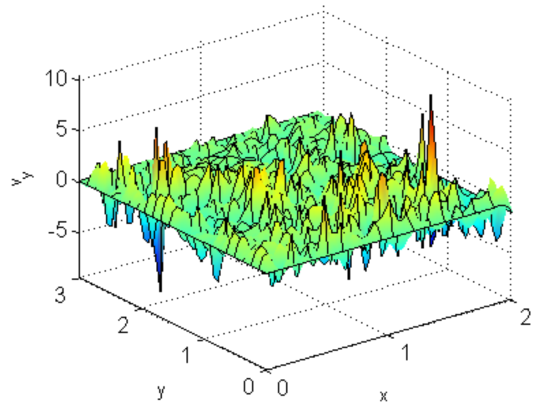

Figure 31: Velocity along y-direction

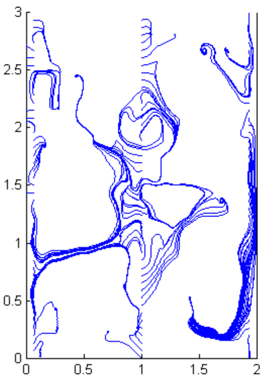

Figure 32: Streamlines

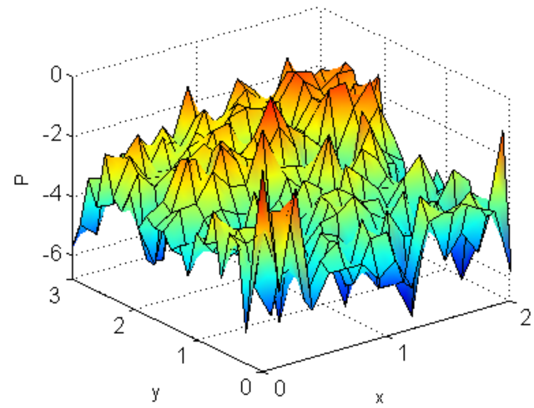

Figure 33: Pressure distribution

We know that Reynolds Number(Re) is a dimensionless quantity that is used to help predict similar flow patterns in different fluid flow situations. We also know that laminar flow occurs at low Reynolds numbers, where viscous forces are dominant, and is characterized by smooth, constant fluid motion and Turbulant flow occurs at high Reynolds numbers and is dominated by inertial forces, which tend to produce chaotic eddies, vortices and other flow instabilities. This we can observe from the analysis we performed for different Reynolds number. As we can see for Re=1, viscous forces are dominant which leads to smooth flow which can be seen from Figure 20 above. And we achieve convergence in iteration number=3. In case of Re=100, we can obtain convergence in iteration number=11 and also the flow is still smooth and we can observe laminar flow. But for higher reynolds number Re=1000,2000 we could not obtain convergence and also flow will be free. As we can see from the streamlines of Re=1000 and 2000(Figures 28,32) it is not smooth flow, since there is no domination of viscous forces which is leading to chaotic solutions in velocity and pressure fields, which can be seen in Figures 30,31,and 33 . The matlab code for Navier-stokes problem is attached in zip file '**exe3**' under which we need to run the subroutine '`mainNavierStokes.m`' to obtain the results for different Reynolds number.