Course Assignment
Finite Elements in Fluids
Master of Science in Computational Mechanics 2016

Paris Dilip Mulye

May 23, 2016

Exercise 1A

The problem is in a 1D domain since $u^h = f(x)$. Neglecting transient term, the strong form of a 1D convection-diffusion-reaction problem is,

$$a\frac{\partial \rho}{\partial x} - \frac{\partial}{\partial x}\left(\nu \frac{\partial \rho}{\partial x}\right) + \sigma\rho = s$$

Multiplying by arbitrary weight function w and converting equation into weak form,

$$\int_\Omega wa\frac{\partial \rho}{\partial x}\,\mathrm{dx} - \int_\Omega w\frac{\partial}{\partial x}\left(\nu\frac{\partial \rho}{\partial x}\right)\,\mathrm{dx} + \int_\Omega w\sigma\rho\,\mathrm{dx} = \int_\Omega ws\,\mathrm{dx}$$

Splitting second term, by performing integration by parts,

$$\int_\Omega wa\frac{\partial \rho}{\partial x}\,\mathrm{dx} - \int_{\Gamma_N+\Gamma_D} w\nu\frac{\partial \rho}{\partial x}\,\mathrm{d\Gamma} + \int_\Omega \frac{\partial w}{\partial x}\nu\frac{\partial \rho}{\partial x}\,\mathrm{dx} + \int_\Omega w\sigma\rho\,\mathrm{dx} = \int_\Omega ws\,\mathrm{dx}$$

Since, w is arbitrary, it is chosen such that $w = 0$ on Dirichlet boundary (Γ_D). This modifies the equation,

$$\int_\Omega wa\frac{\partial \rho}{\partial x}\,\mathrm{dx} + \int_\Omega \frac{\partial w}{\partial x}\nu\frac{\partial \rho}{\partial x}\,\mathrm{dx} + \int_\Omega w\sigma\rho\,\mathrm{dx} = \int_\Omega ws\,\mathrm{dx} + \int_{\Gamma_N} w\nu\frac{\partial \rho}{\partial x}\,\mathrm{d\Gamma}$$

Finite element discretization, $\rho \approx \rho^h = \sum N_j(x)\rho_j$, $s \approx s^h = \sum N_j(x)s_j$, and Galerkin method, $w_i = N_i$, taking $\nu\dfrac{\partial \rho}{\partial x} = h$,

$$\left[\int_\Omega N_i a\sum\frac{\partial N_j}{\partial x}\,\mathrm{dx} + \int_\Omega \frac{\partial N_i}{\partial x}\nu\sum\frac{\partial N_j}{\partial x}\,\mathrm{dx} + \int_\Omega N_i\sigma\sum N_j\,\mathrm{dx}\right]\rho_j = \left[\int_\Omega N_i\sum N_j\,\mathrm{dx}\right]s_j + \int_{\Gamma_N} N_i h\,\mathrm{d\Gamma}$$

This can be written in matrix form (terms in respective order)

$$(C_1 + K_1 + M_1)_{ij}\rho_j = M_{ij}s_j + \int_{\Gamma_N} N_i h\,\mathrm{d\Gamma}$$

Which can be further represented as,

$$K\rho = F$$

Application of Dirichlet boundary condition can be done either by substituting known ρ in above matrix and reducing the variables, or by the eigenvalue approach. Thus, the above linear system can be solved.

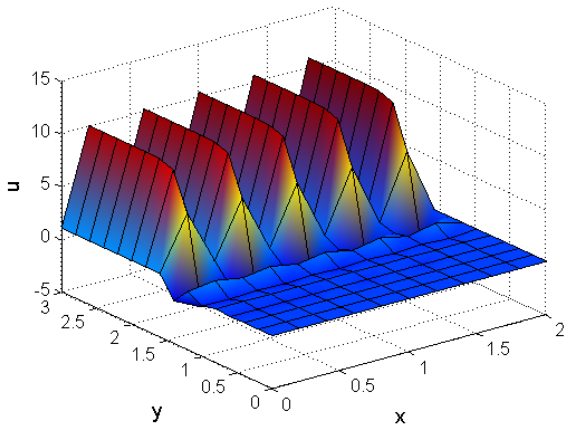

Figure 1: Case 1, Galerkin Solution, h=0.2

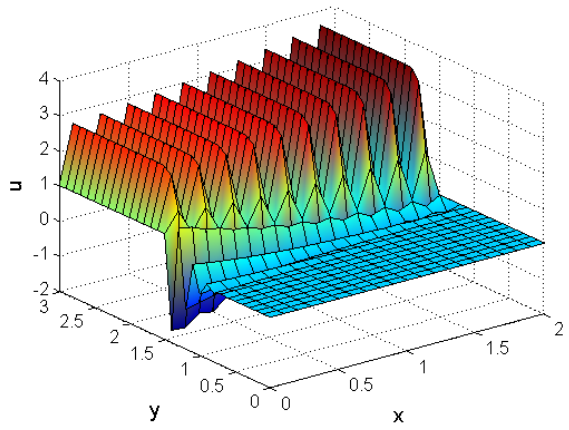

Figure 2: Case 1, Galerkin Solution, h=0.1

Exercise 1B

The Galerkin solution was obtained by following MATLAB code lines, with $\tau = 0$.

```
Ke   = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + N_ig'*sigma*N_ig)*dvolu;
aux  = N_ig*Xe;
f_ig = SourceTerm(aux,example);
fe   = fe + N_ig'*(f_ig*dvolu);
```

Because elemental matrices in weak form, are of form,

$$\begin{aligned} K^e\rho^e &= F^e \\ K^e &= \nu\boldsymbol{\nabla\rho}\cdot\boldsymbol{\nabla w} + (\boldsymbol{a}\cdot\boldsymbol{\nabla\rho})w + \sigma\rho w \\ F^e &= ws \end{aligned}$$

The boundary conditions were given as follows,

```
gamma2     = find(X(:,1) ==0 & X(:,2) >=1.5);
gamma4     = find(X(:,1) ==2);
nodesDir1  = gamma2;
nodesDir0  = gamma4;
C          = [nodesDir1, ones(length(nodesDir1),1);
              nodesDir0, zeros(length(nodesDir0),1)];
```

Case 1: Galerkin Method (a=1, ν=1e-3, σ=1e-3, s=0, h=0.2)

Since, ν and σ are very small compared to a, this is a convection dominated problem. Due to the dominant unsymmetrical $\mathbf{C}$ term ($N\boldsymbol{a}\cdot\boldsymbol{\nabla N}$), Galerkin solution is expected to be unstable. As per expectations, results (Figure 1) show spurious oscillations in the solution.

One idea to reduce oscillations without changing a, ν, σ and s is to reduce mesh size (h). The Galerkin solution with $h = 0.1$ (Figure 2). The amplitude of oscillations reduced with refinement in mesh, but stability is still not achieved. From theory, it is known that Galerkin solution is stable for Peclet Number, $Pe \leq 1$. Pe for $h = 0.2$ is 100. Even after reduction in h, Pe is 50. For stable solution, $Pe = 1$, h = 2e-3. This is too small and would result in a very big system of equations. Therefore, a stabilization method is needed for this case.

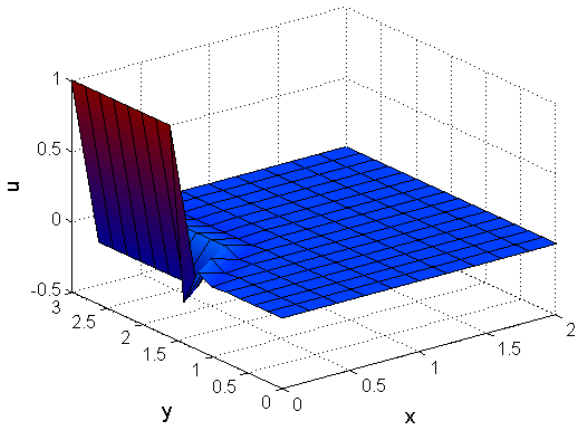

Figure 3: Case 2, Galerkin Solution, h=0.2

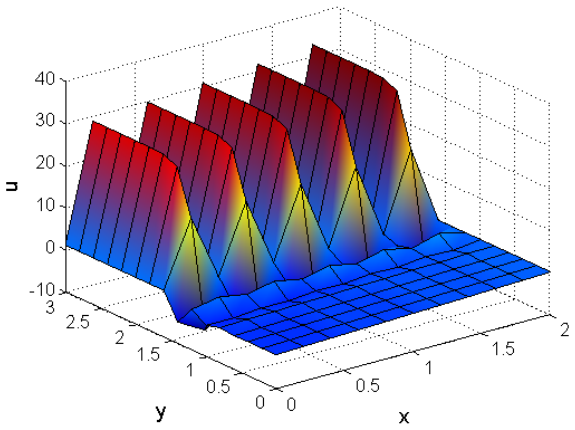

Figure 4: Case 3, Galerkin Solution, h=0.2

Case 2: Galerkin Method (a=1e-3, ν=1e-3, σ=1, s=0, h=0.2)

The Peclet Number, $Pe = 0.1$. Therefore, Galerkin solution is expected to be stable which is also seen from solution plot (Figure 3).

Case 3: Galerkin Method (a=1, ν=1e-3, σ=0, s=1, h=0.2)

The Peclet Number, $Pe = 100$. Therefore, Galerkin solution is expected to be unstable (Figure 4). Similar to Case 1, refinement can reduce the oscillations, but again that would results in a really small h. So in this case as well, a stabilization method is needed.

Case 4: Galerkin Method (a=1, ν=1e-3, σ=1, s=0, h=0.2)

The Peclet Number, $Pe = 100$. Therefore, Galerkin solution is expected to be unstable (Figure 5). Similar to Case 1 and 3, refinement can reduce the oscillations, but again that would results in a really small h. So in this case as well, a stabilization method is needed.

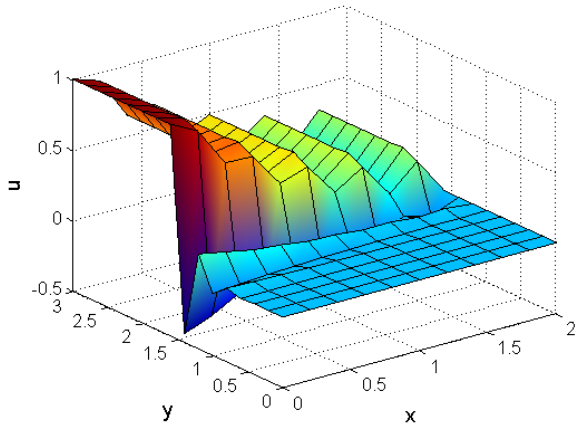

Figure 5: Case 4, Galerkin Solution, h=0.2

Since, Galerkin solution is good for Case 2, a stabilization technique is not needed for this case. But for other 3 cases, a stabilization is required.

Exercise 1C

To consider SUPG and GLS methods, it is important to know value of stabilization parameter, τ for all the relevant cases (1,3 and 4). The second order accurate formula for τ is given as,

$$\tau = \frac{h}{2a}\left(1 - \frac{1}{Pe} + \frac{h}{2a}\sigma\right)^{-1} \tag{1}$$

$\tau_{Case1} = 0.101$, $\tau_{Case3} = 0.101$ and $\tau_{Case4} = 0.092$. Since, all τ are very close, the weight of stabilization term is almost same for both methods. Also, for $\sigma = 0$, GLS solution would be equivalent to that of SUPG (for linear elements). So In case 3, both methods would give exactly same solution. Similarly, for Case 1, where, $\sigma \approx 0$, the solutions of SUPG and GLS would be very close. The case where stabilized solutions would differ is case 4. Thus, if SUPG and GLS methods are to be compared, Case 4 would be appropriate. Therefore, for this Exercise, Case 4 was chosen. (a=1, ν=1e-3, σ=1, s=0, h=0.2).

The stabilization term for all methods is of the form (at elemental level),

$$\int_e P(w)\tau R(\rho)\,\mathrm{d}\Omega$$

Where, $P(w)$ differs from method to method, stabilization parameter τ is given by (1) and $R(\rho)$ is the residual of PDE. Since,the stabilization term contains product with residual, it ensures consistency of method.

SUPG Method

For SUPG method, $P(w) = \boldsymbol{a}\cdot\boldsymbol{\nabla}w$ and $R(\rho) = \boldsymbol{a}\cdot\boldsymbol{\nabla}\rho - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}\rho) + \sigma\rho - s$. Since, elements are linear, during discretization ($\rho = \rho^h$), the term, $\boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}\rho)$ in residual would be 0, since it contains second derivatives of ρ^h which are zero. Thus, $R(\rho) = \boldsymbol{a}\cdot\boldsymbol{\nabla}\rho + \sigma\rho - s = L(\rho) - s$. Therefore, the weak form of PDE at elemental level (with stabilization term) becomes,

$$\int_{\Omega_e} w\boldsymbol{a}\cdot\boldsymbol{\nabla}\rho\,\mathrm{d}\Omega + \int_{\Omega_e}\nu\boldsymbol{\nabla}w\cdot\boldsymbol{\nabla}\rho\,\mathrm{d}\Omega + \int_{\Omega_e} w\sigma\rho\,\mathrm{d}\Omega + \int_{\Omega_e}(\boldsymbol{a}\cdot\boldsymbol{\nabla}w)\tau L(\rho)\,\mathrm{d}\Omega = \int_{\Omega_e} ws\,\mathrm{d}\Omega + \int_{\Omega_e}(\boldsymbol{a}\cdot\boldsymbol{\nabla}w)\tau s\,\mathrm{d}\Omega$$

Note that the Neumann term is not considered here, since while assembling the internal fluxes will get canceled and global boundary fluxes would be added to the global matrix directly after assembly. Combining terms under one integral,

$$\int_{\Omega_e}\left[w\boldsymbol{a}\cdot\boldsymbol{\nabla}\rho + \nu\boldsymbol{\nabla}w\cdot\boldsymbol{\nabla}\rho + w\sigma\rho + \tau(\boldsymbol{a}\cdot\boldsymbol{\nabla}w)(\boldsymbol{a}\cdot\boldsymbol{\nabla}\rho + \sigma\rho)\right]\,\mathrm{d}\Omega = \int_{\Omega_e}\left[w\ + \tau(\boldsymbol{a}\cdot\boldsymbol{\nabla}w)\right]s\,\mathrm{d}\Omega$$

This equation after discretization ($\rho = \rho^h$), using, Galerkin approximation, $w_i = N_i$, and using same shape functions for source term s, the terms in above equation are expressed as follows,

$$\begin{aligned}
w\boldsymbol{a}\cdot\boldsymbol{\nabla}\rho &= \texttt{N_ig'*(ax*Nx+ay*Ny)}\\
\nu\boldsymbol{\nabla}w\cdot\boldsymbol{\nabla}\rho &= \texttt{nu*(Nx'*Nx+Ny'*Ny)}\\
w\sigma\rho &= \texttt{N_ig'*sigma*N_ig}\\
\tau(\boldsymbol{a}\cdot\boldsymbol{\nabla}w)(\boldsymbol{a}\cdot\boldsymbol{\nabla}\rho + \sigma\rho) &= \texttt{tau*(ax*Nx+ay*Ny)'*(ax*Nx + ay*Ny + N_ig*sigma)}\\
[w + \tau(\boldsymbol{a}\cdot\boldsymbol{\nabla}w)]s &= \texttt{(N_ig+tau*(ax*Nx+ay*Ny))'*(f_ig)}
\end{aligned}$$

Therefore, the completed MATLAB code is,

```
Ke   = Ke + (N_ig'*(ax*Nx+ay*Ny) + nu*(Nx'*Nx+Ny'*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,example);
fe   = fe + (N_ig+tau*(ax*Nx+ay*Ny))'*(f_ig*dvolu);
```

GLS Method

For GLS method, $P(w) = \boldsymbol{a} \cdot \boldsymbol{\nabla} w - \boldsymbol{\nabla} \cdot (\nu \boldsymbol{\nabla} w) + \sigma w$ and $R(\rho) = \boldsymbol{a} \cdot \boldsymbol{\nabla} \rho - \boldsymbol{\nabla} \cdot (\nu \boldsymbol{\nabla} \rho) + \sigma \rho - s$. Since, elements are linear, during discretization ($\rho = \rho^h$), the term, $\boldsymbol{\nabla} \cdot (\nu \boldsymbol{\nabla} \rho)$ in residual would be 0, since it contains second derivatives of ρ^h which are zero. Thus, $R(\rho) = \boldsymbol{a} \cdot \boldsymbol{\nabla} \rho + \sigma \rho - s = L(\rho) - s$. Also, during Galerkin approximation, $w_i = N_i$, second derivatives of w would be zero. Thus, $P(w) = \boldsymbol{a} \cdot \boldsymbol{\nabla} w + \sigma w$ Therefore, the weak form of PDE at elemental level (with stabilization term) becomes,

$$\int_{\Omega_e} w \boldsymbol{a} \cdot \boldsymbol{\nabla} \rho \, \mathrm{d}\Omega + \int_{\Omega_e} \nu \boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} \rho \, \mathrm{d}\Omega + \int_{\Omega_e} w \sigma \rho \, \mathrm{d}\Omega + \int_{\Omega_e} (\boldsymbol{a} \cdot \boldsymbol{\nabla} w + \sigma w) \tau L(\rho) \, \mathrm{d}\Omega = \int_{\Omega_e} w s \, \mathrm{d}\Omega + \int_{\Omega_e} (\boldsymbol{a} \cdot \boldsymbol{\nabla} w + \sigma w) \tau s \, \mathrm{d}\Omega$$

Note that the Neumann term is not considered here, since while assembling the internal fluxes will get canceled and global boundary fluxes would be added to the global matrix directly after assembly. Combining terms under one integral,

$$\int_{\Omega_e} [w \boldsymbol{a} \cdot \boldsymbol{\nabla} \rho + \nu \boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} \rho + w \sigma \rho + \tau (\boldsymbol{a} \cdot \boldsymbol{\nabla} w + \sigma w)(\boldsymbol{a} \cdot \boldsymbol{\nabla} \rho + \sigma \rho)] \, \mathrm{d}\Omega = \int_{\Omega_e} [w \, + \tau (\boldsymbol{a} \cdot \boldsymbol{\nabla} w + \sigma w)] \, s \, \mathrm{d}\Omega$$

This equation after discretization ($\rho = \rho^h$), using, Galerkin approximation, $w_i = N_i$, and using same shape functions for source term s, the terms in above equation are expressed as follows,

$$w \boldsymbol{a} \cdot \boldsymbol{\nabla} \rho = \texttt{N\_ig'*(ax*Nx+ay*Ny)}$$
$$\nu \boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} \rho = \texttt{nu*(Nx'*Nx+Ny'*Ny)}$$
$$w \sigma \rho = \texttt{N\_ig'*sigma*N\_ig}$$
$$\tau (\boldsymbol{a} \cdot \boldsymbol{\nabla} w + \sigma w)(\boldsymbol{a} \cdot \boldsymbol{\nabla} \rho + \sigma \rho) = \texttt{tau*(ax*Nx+ay*Ny+sigma*N\_ig)'*(ax*Nx + ay*Ny + N\_ig*sigma)}$$
$$[w + \tau (\boldsymbol{a} \cdot \boldsymbol{\nabla} w + \sigma w)] s = \texttt{(N\_ig+tau*(ax*Nx+ay*Ny+sigma*N\_ig))'*(f\_ig)}$$

Therefore, the completed MATLAB code is,

```
Ke   = Ke + (N_ig'*(ax*Nx+ay*Ny) + nu*(Nx'*Nx+Ny'*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,example);
fe   = fe + (N_ig+tau*(ax*Nx+ay*Ny+sigma*N_ig))'*(f_ig*dvolu);
```

Results

As expected, the solutions of both SUPG (Figure 6)and GLS (Figure 7)are free of spurious oscillations. despite the difference in σ the solutions are very close to each other. The maximum difference between the solutions is 0.03 and it occurs at node 78 which has coordinates (0,1.4) . SUPG solution = 0.23, GLS solution = 0.20. The GLS solution is lower than that of SUPG at this node.

It is an expected result, since in case of Galerkin method, the solution at this node is -0.44. Comparing both SUPG and GLS, for linear elements (and a constant positive reaction), GLS performs exactly like

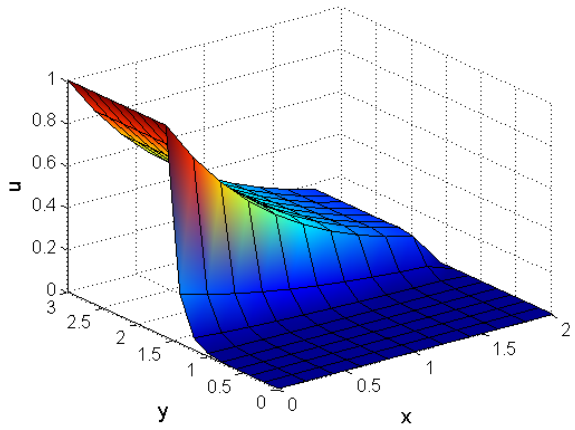

Figure 6: Case 4, SUPG Solution, h=0.2

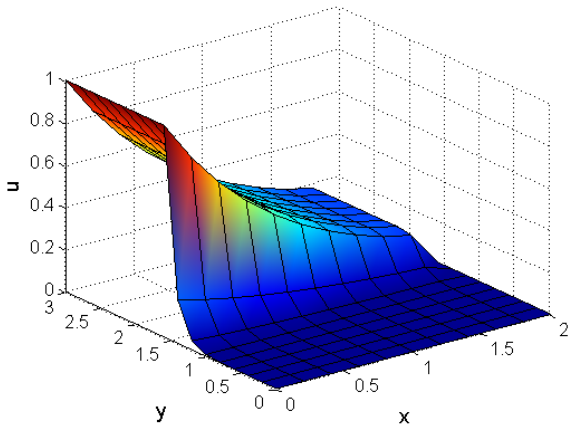

Figure 7: Case 4, GLS Solution, h=0.2

SUPG with the Galerkin term weighted $(1 + \sigma\tau)$ times more. This implies that the oscillations due to Galerkin are a little more amplified in GLS compared with SUPG. Therfore, GLS solution has to lie between SUPG and Galerkin which is exactly what has been observed.

Exercise 1D

Case 4 (a=1, ν=1e-3, σ=1, s=0, h=0.2) was solved using SUPG, but with new BCs. The new BCs were applied as follows,

```
nodesDir2 = gamma2;
nodesDir1 = gamma4;
% Boundary condition matrix
C = [nodesDir2, 2*ones(length(nodesDir2),1);
     nodesDir1, 1*ones(length(nodesDir1),1)];
```

The solution of Dirichlet problem was obtained and the flux was found at the post-processing stage using following equation, where K is global stiffness matrix (including all nodes), $Temp$ contains Dirichlet solution, f is the source term matrix for all nodes.

```
flux = K*Temp-f;
```

The value of flux at Γ_4 was stored to be used fro Neumann problem. The BCs for Neumann problem were applied as follows,

```
nodesDir2 = gamma2;
% Boundary condition matrix
C = [nodesDir2, 2*ones(length(nodesDir2),1)];
```

Since, Γ_4 is a Neumann BC now, it has been excluded from the boundary condition matrix. The fluxes at Γ_4 were applied to the system by adding them to f using following equation,

```
f(gamma4) = f(gamma4) + flux;
```

The solution of Dirichlet (Figure 8) and Neumann (Figure 9) was found to match. The maximum absolute difference between solutions was found to be 0.0006 which occurs at node 176 which has coordinates (2,3).

A similar procedure was followed for GLS method. The solution of Dirichlet (Figure 10) and Neumann (Figure 11) was found to match. The maximum absolute difference between solutions was found to be 0.0005 which occurs at node 176 which has coordinates (2,3).

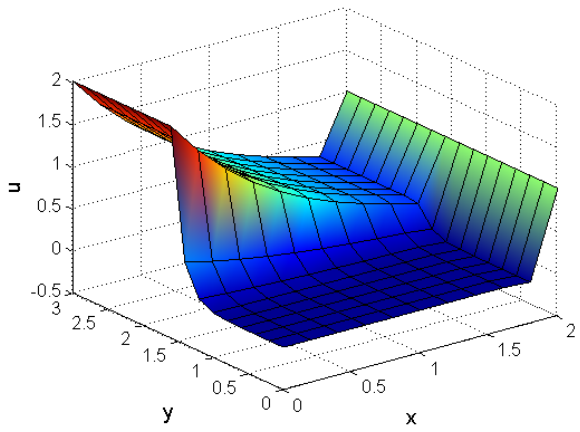

Figure 8: Case 4, SUPG Solution, Dirichlet

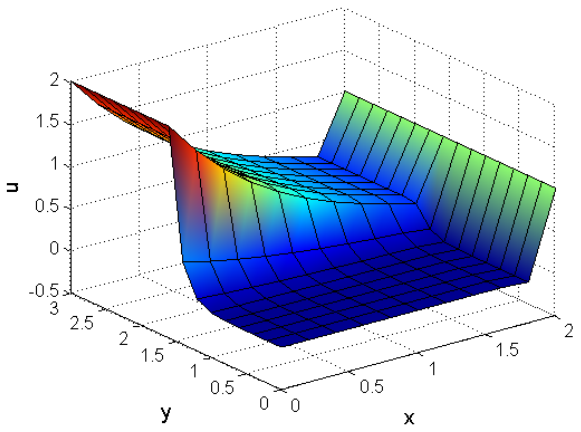

Figure 9: Case 4, SUPG Solution, Neumann

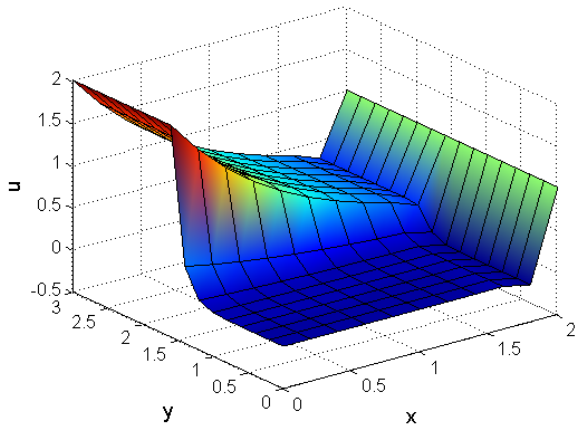

Figure 10: Case 4, GLS Solution, Dirichlet

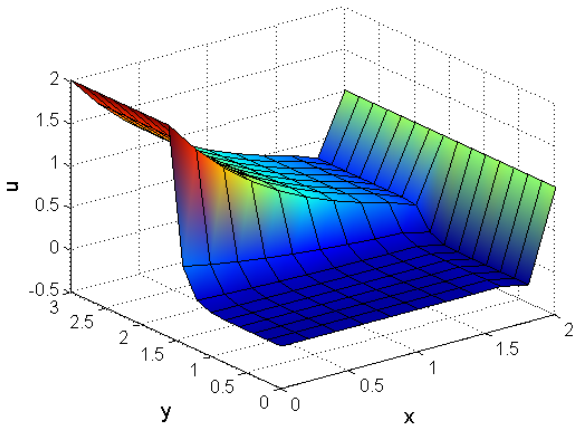

Figure 11: Case 4, GLS Solution, Neumann

Exercise 2A

Case 4 of Exercise 1 is continued further (all parameters are same except convective velocity). It is important to note here that convective velocity is not divergence free. Therefore, a general form of Convection-Diffusion-Reaction (CDR) equation needs to be used for deriving the discretization. SUPG formulation has been used for space discretization. All parameters except convective velocity are assumed to be constant in the domain and taken out of integration whenever necessary.

SUPG-Crank Nicolson

The strong form of CDR equation (2) and time marching algorithm for θ methods (3) are as follows,

$$u_t = -\boldsymbol{\nabla} \cdot (\boldsymbol{a}u) + \nu \nabla^2 u - \sigma u \tag{2}$$

$$\frac{P}{\Delta t} = \theta P_t + u_t^n \tag{3}$$

The stabilization term would be added, once the weak form is obtained. P is defined as $u^{n+1} - u^n$ where the superscript indicates time step. Multiplying (3) by weight function w, and converting in integral form, (Integrals signs are not written unless they are on boundaries)

$$w\frac{P}{\Delta t} = w\theta P_t + w u_t^n$$

The strong form of CDR can be written in difference form as follows,

$$P_t = -\boldsymbol{\nabla} \cdot (\boldsymbol{a}P) + \nu \nabla^2 P - \sigma P$$

Substituting P_t and u_t^n in equation,

$$w\frac{P}{\Delta t} = w\theta(-\boldsymbol{\nabla} \cdot (\boldsymbol{a}P) + \nu \nabla^2 P - \sigma P) + w(-\boldsymbol{\nabla} \cdot (\boldsymbol{a}u^n) + \nu \nabla^2 u^n - \sigma u^n)$$

$$\left[w\frac{P}{\Delta t}\right] = -\left[\theta w \boldsymbol{\nabla} \cdot (\boldsymbol{a}P)\right] + \left[\theta w \nu \nabla^2 P\right] - \left[\theta w \sigma P\right] - \left[w \boldsymbol{\nabla} \cdot (\boldsymbol{a}u^n)\right] + \left[w \nu \nabla^2 u^n\right] - \left[w \sigma u^n\right] \tag{4}$$

The terms in square brackets would be addressed one by one. Before proceeding it is necessary to notice the structure of various matrices defined in MATLAB program. These will be used to simplify and represent the individual terms in (4). The boundary was divided into two parts, outflow (defined by $\boldsymbol{a} \cdot \boldsymbol{n} > 0$ and 'inflow + no flow' defined as $\boldsymbol{a} \cdot \boldsymbol{n} \leq 0$. The definitions of matrices are as follows.

```
M     = Ni*Nj
K     = a grad Ni * a grad Nj
C     = Ni * grad Nj
R1    = SUPG correction Term 1 (coeff of solution variable)
R2    = SUPG correction Term 2 (coeff of u^n)
KS    = grad Ni * grad Nj
Mout  = (a.n) Ni Nj (on Outflow)
Cout  = (a.n) Ni (a.grad Nj) (on outflow)
Min   = (a.n) Ni Nj (on Inflow or zero flow)
Cin   = (a.n) Ni (a.grad Nj) (on Inflow or zero flow)
CoutN = Ni (n. grad Nj) (on Outflow)
CinN  = Ni (n. grad Nj) (on inflow or no flow)
```

Going back to equation, The terms are solved as follows, terms P and u^n on RHS indicate the vector

of nodal values of respective quantities

$$
\begin{aligned}
1 : w\frac{P}{\Delta t} &= \frac{M}{dt}P \\
2 : -\theta w \boldsymbol{\nabla}\cdot(\boldsymbol{a}P) &= -\theta \boldsymbol{\nabla}\cdot(w\boldsymbol{a}P) + \theta(\boldsymbol{\nabla}w\cdot\boldsymbol{a})P \\
&= -\theta \int_\Gamma (\boldsymbol{a}\cdot\boldsymbol{n})wP + \theta(\boldsymbol{\nabla}w\cdot\boldsymbol{a})P = -\theta(M_{out} - M_{in} - C^T)P \\
3 : \theta w\nu\nabla^2 P &= \theta\nu \int_\Gamma w(\boldsymbol{n}\cdot\boldsymbol{\nabla}P) - \theta\nu(\boldsymbol{\nabla}w\cdot\boldsymbol{\nabla}P) = \theta\nu(C_{out} - C_{in} - KS)P \\
4 : -\theta\sigma wP &= -\sigma\theta MP \\
5 : -w\boldsymbol{\nabla}\cdot(\boldsymbol{a}u^n) &= -\boldsymbol{\nabla}\cdot(w\boldsymbol{a}u^n) + (\boldsymbol{\nabla}w\cdot\boldsymbol{a})u^n \\
&= -\int_\Gamma (\boldsymbol{a}\cdot\boldsymbol{n})wu^n + (\boldsymbol{\nabla}w\cdot\boldsymbol{a})u^n = -(M_{out} - M_{in} - C^T)u^n \\
6 : w\nu\nabla^2 u^n &= \nu \int_\Gamma w(\boldsymbol{n}\cdot\boldsymbol{\nabla}u^n) - \nu(\boldsymbol{\nabla}w\cdot\boldsymbol{\nabla}u^n) = \nu(C_{out} - C_{in} - KS)u^n \\
7 : -\sigma w u^n &= -\sigma M u^n
\end{aligned}
$$

Combining all the terms together,

$$
\begin{aligned}
P\left(\frac{M}{dt} + \theta(M_{out} - M_{in}) - \theta C^T - \theta\nu(C_{out} - C_{in}) + \theta\nu KS + \theta\sigma M\right) = \\
u^n\left(-(M_{out} - M_{in}) + C^T + \nu(C_{out} - C_{in}) - \nu KS - \sigma M\right)
\end{aligned} \tag{5}
$$

This can be written as follows,

$$AP = Bu^n$$

Since, u^n is known and so are A and B. Thus P can be obtained by solving the above system of linear equations. Thus u^{n+1} can be obtained using the relation, $u^{n+1} = u^n + P$. This formulation does not include the stabilization term yet, which will be added now. In the strong form, the stabilization term would appear as follows,

$$u_t = -\boldsymbol{\nabla}\cdot(\boldsymbol{a}u) + \nu\nabla^2 u - \sigma u + S_{strong}$$

Using the stabilization term form for SUPG and using the above definition of residual,

$$
\begin{aligned}
Stable &= \int \boldsymbol{\nabla}\cdot(\boldsymbol{a}w)\tau(R^{n+1} - R^n) \\
R^{n+1} &= u_t^{n+1} + \boldsymbol{\nabla}\cdot(\boldsymbol{a}u^{n+1}) - \nu\nabla^2 u^{n+1} + \sigma u^{n+1} \\
R^n &= u_t^n + \boldsymbol{\nabla}\cdot(\boldsymbol{a}u^n) - \nu\nabla^2 u^n + \sigma u^n
\end{aligned}
$$

The residual can be written as,(integration sign not written)

$$Stable = \boldsymbol{\nabla}\cdot(\boldsymbol{a}w)\tau(P_t + \boldsymbol{\nabla}\cdot(\boldsymbol{a}P) - \nu\nabla^2 P + \sigma P)$$

Using the algorithm for time marching of Crank Nicolson,

$$P_t = \frac{P}{\theta\Delta t} - \frac{u_t^n}{\theta}$$

Substituting back and expanding u_t^n, R_1 is stabilization term to be added to A matrix and R_2 is stabilization term to be subtracted from B matrix. Also, the stabilization terms can be directly integrated without converting into weak form.

$$R_1 = \boldsymbol{\nabla} \cdot (\boldsymbol{a}w)\tau \left(\frac{P}{\theta \Delta t} - \nu \nabla^2 P + \boldsymbol{\nabla} \cdot (\boldsymbol{a}P) + \sigma P \right)$$
$$R_2 = \boldsymbol{\nabla} \cdot (\boldsymbol{a}w)\frac{\tau}{\theta} \left(-\nu \nabla^2 u^n + \boldsymbol{\nabla} \cdot (\boldsymbol{a}u^n) + \sigma u^n \right)$$

The stabilization parameter defined for θ methods is given as ,

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

It is important to note that since a changes within the domain, τ has to be calculated on elemental level. Assuming linear elements, thus the diffusive terms would vanish. Also, noting that $\boldsymbol{\nabla} \cdot (\boldsymbol{a}w) = (\boldsymbol{\nabla} \cdot \boldsymbol{a})(w) + \boldsymbol{\nabla} w \cdot \boldsymbol{a}$, and for the given field in the problem, $\boldsymbol{\nabla} \cdot \boldsymbol{a} = -2$. Using these relations,

$$R_1 = (-2N + \boldsymbol{a} \cdot \boldsymbol{\nabla} N)\tau \left(\frac{N}{\theta \Delta t} + \boldsymbol{a} \cdot \boldsymbol{\nabla} N - 2N + N\sigma \right)$$
$$R_2 = (-2N + \boldsymbol{a} \cdot \boldsymbol{\nabla} N)\frac{\tau}{\theta} \left(\boldsymbol{a} \cdot \boldsymbol{\nabla} N - 2N + N\sigma \right)$$

Therefore, the new equations become,

$$(A + R1)P = (B - R2)$$

Exercise 2B-2C

The following subroutines calculate the quantities mentioned in previous section. (Crank Nicolson)

```
%operations on convection
a      = N_ig*Conve; ax = a(1); ay = a(2);
aGradN = ax*Nx + ay*Ny;
GradN  = [Nx;Ny];
speed  = sqrt(a(1)^2+a(2)^2);
%standard martices calculation
KSe = Ke + GradN'*GradN*dvolu;
Ce  = Ce + N_ig'*aGradN*dvolu;
Ke  = Ke + aGradN'*aGradN*dvolu;
Me  = Me + N_ig'*N_ig*dvolu;
%stabilization terms
tau  = (1/theta/dt+2*speed/h+4*nu/h/h+sigma)^-1;
P    = (-2*N_ig+aGradN);
R1e  = R1e + P'*tau*(N_ig/theta/dt -2*N_ig + aGradN + sigma*N_ig)*dvolu;
R2e  = R2e + P'*tau/theta*(-2*N_ig + aGradN + sigma*N_ig)*dvolu;
```

The solver has been implemented as follows which solves the system indicated in the previous section.

```
A = M/dt+theta*(Mout-Min)-theta*C'+sigma*theta*M-nu*theta*(CoutN-CinN)+nu*theta*KS+R1;
B = -(Mout-Min)+C'-sigma*M+nu*(CoutN-CinN)-nu*KS-R2;
for n=1:1:nStep
    Ktot     = [A,ADir.';ADir, zeros(nDir,nDir)];
    Ftot     = [B*u(:,n);bDir];
    Soltot   = Ktot\Ftot;
    u(:,n+1) = u(:,n) + Soltot(1:size(A,1));
end
```

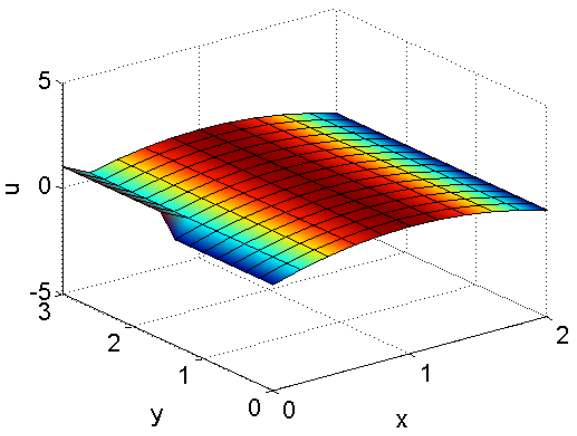

Figure 12: Initial Condition

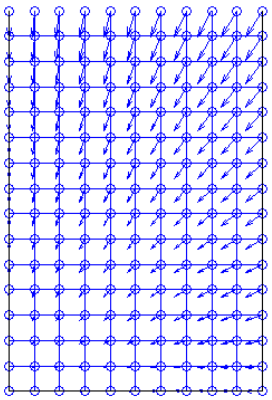

Figure 13: Velocity Plot, Mesh Plot

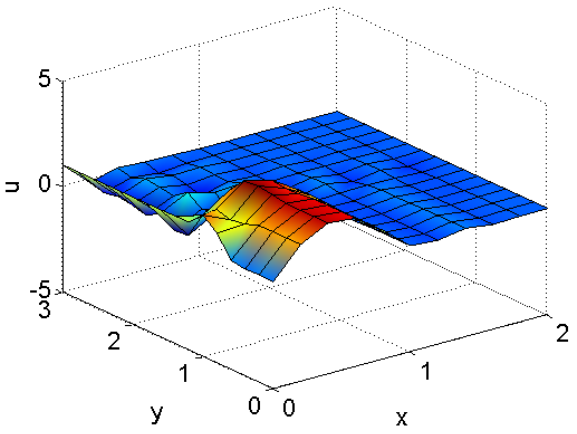

Figure 14: Solution at tEnd, No Stabilization

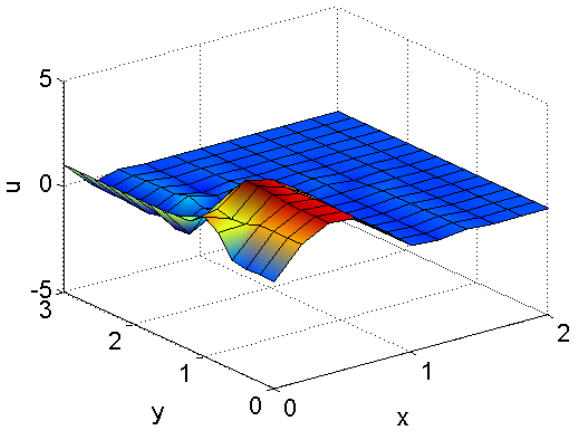

Figure 15: Solution at tEnd, SUPG

The given initial and boundary conditions have been implemented (Figure 12), $h = 0.2$ was used for mesh (Figure 13) and convection velocity vectors are plotted (Figure 13).

Exercise 2D

With the parameters ($h = 0.2, tEnd = 0.75, nStep = 110, \nu = 1e-3, \sigma = 1, \theta = 0.5, method = CN$) simulation was run, both with and without stabilization to observe difference. The solution plots at end time are shown in Figure 14 and Figure 15.

Solutions indicate the fact that stabilization technique are performing as per expectation. The minimum value of u throughout the simulation was -0.7644 without stabilization and it became -0.6425 after stabilization. Also, the maximum value of u remain unaffected when compared with the change in minimum (2.338 with stabilization and 0.3433 without it). This indicates that the stabilization technique has reduced the noise (e.g. negative u values) without affecting the maxima.

The following plots show SUPG solution at different times to notice its transient behavior. The behavior makes sense, since the convective velocity points towards the origin. So everything should move towards origin. Since, there is not outflow, there will be accumulation of u near origin which is observed too. Since, this is a convection-reaction dominated problem, motion of peaks can ob observed from the plots.

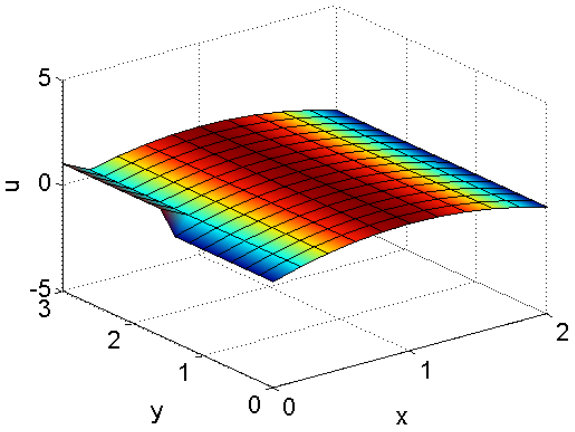

Figure 16: SUPG-CN, Time = 0

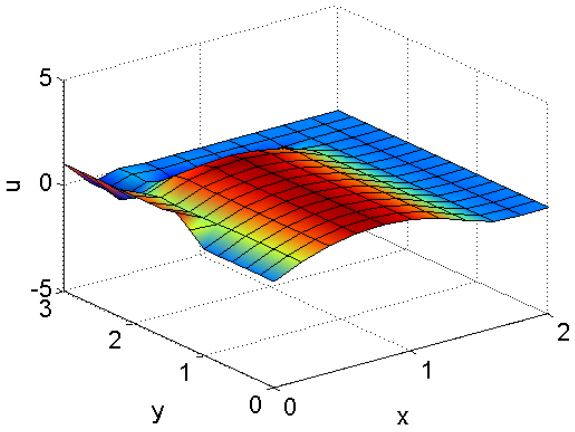

Figure 17: SUPG-CN, Time = 0.25

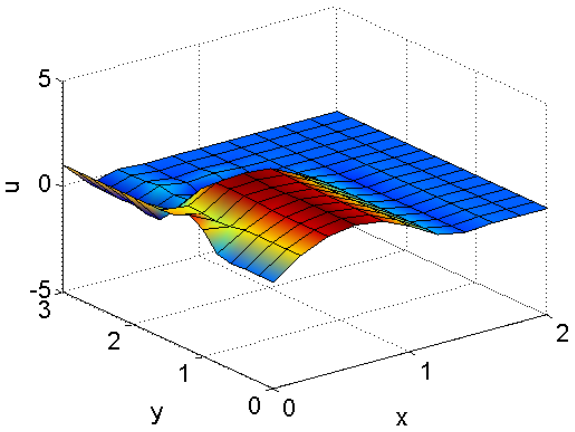

Figure 18: SUPG-CN, Time = 0.5

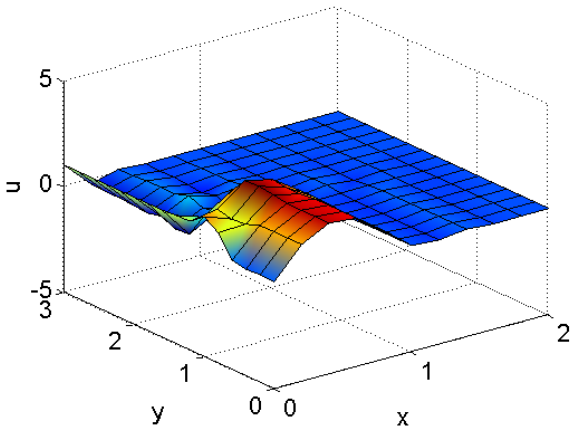

Figure 19: SUPG-CN, Time = 0.75

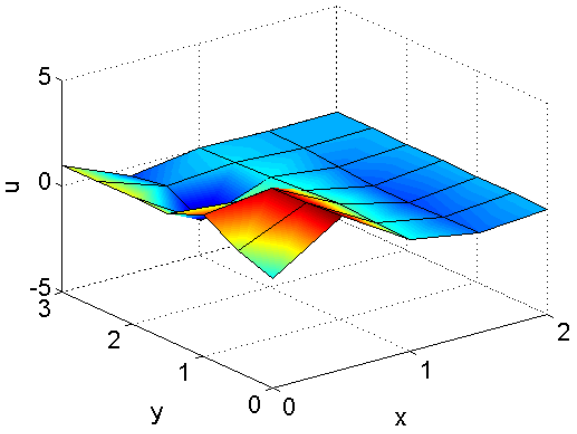

Figure 20: SUPG-CN, Time = 0.75, h = 0.5

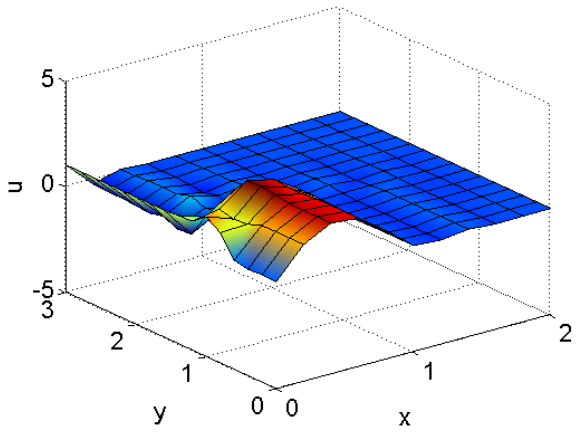

Figure 21: SUPG-CN, Time = 0.75, h = 0.2

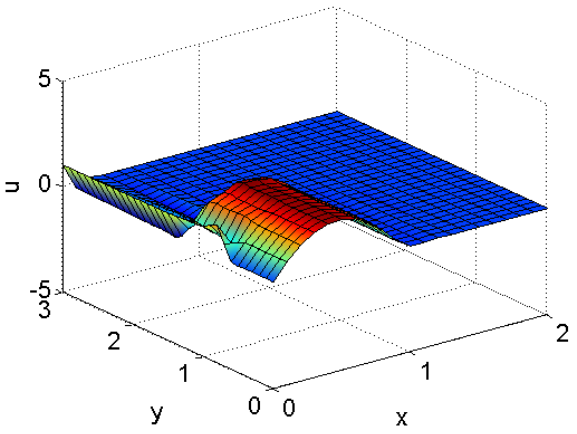

Figure 22: SUPG-CN, Time = 0.75, h = 0.1

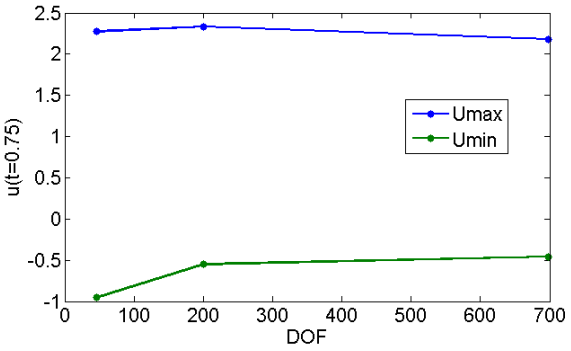

Figure 23: SUPG-CN, Mesh Sensitivity

Mesh Size Sensitivity

The case was simulated for 3 different mesh sizes, (h=0.5, 0.2 and 0.1). Figure 20 to Figure 22 show the solution at end time. Maximum and minimum values of u were found to be very insensitive towards mesh size. Figure 23 plots the maximum and minimum value of u as a function of size of matrix to be inverted (DOF+Dirichlet). Even though minimum value of u was found to increase (absolute value) substantially when mesh size was changed from h=0.5 to h=0.2, this could be to due to fact, that h=0.5 mesh is very stiff compared to h=0.2 thus it resists more, resulting in less value of u. Overall, insensitivity towards mesh indicates that the stabilization techniques are working well in space.

Time Step Sensitivity

The case was simulated for 3 different time steps, (nStep=55, 110 and 220). A constant mesh size of h=0.2 was used. Figure 24 to Figure 26 show the solution at end time. Maximum and minimum values of u were found to be very insensitive towards mesh size. Figure 23 plots the maximum and minimum value of u as a function of time steps. Overall, insensitivity towards time steps indicates that the stabilization techniques and time marching algorithm are working well.

Computational Time Comparison

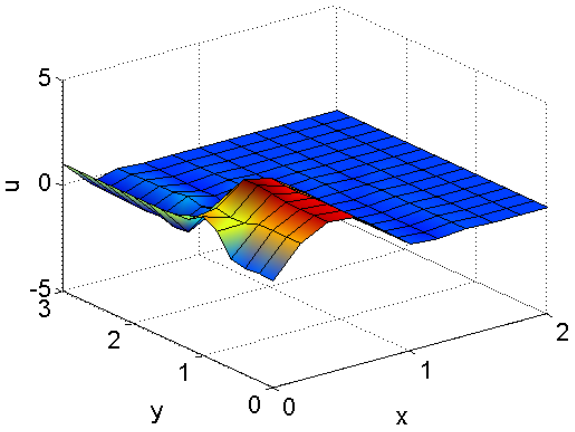

Figure 24: SUPG-CN, h = 0.2, steps = 55

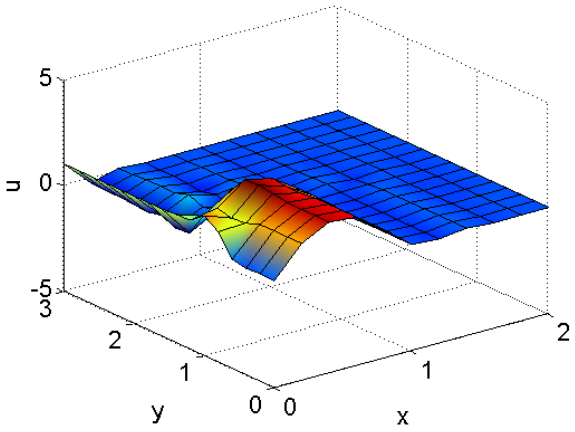

Figure 25: SUPG-CN, h = 0.2, steps = 110

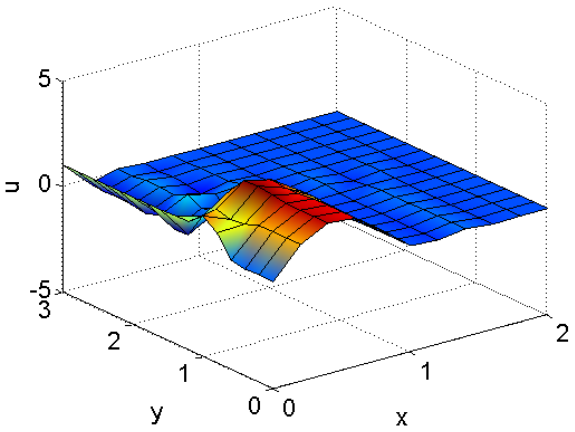

Figure 26: SUPG-CN, h = 0.2, steps = 220

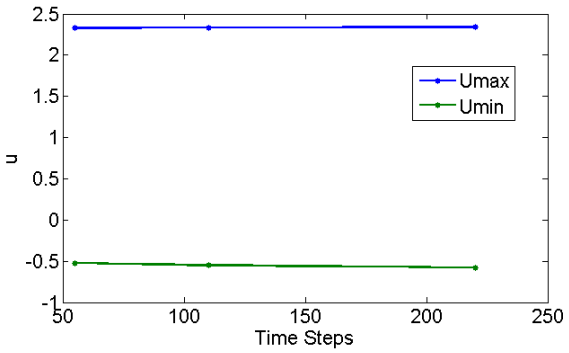

Figure 27: SUPG-CN,Time Step Sensitivity

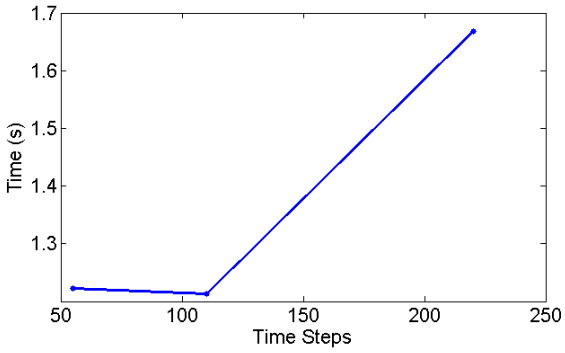

Figure 28: SUPG-CN,Time vs nSteps

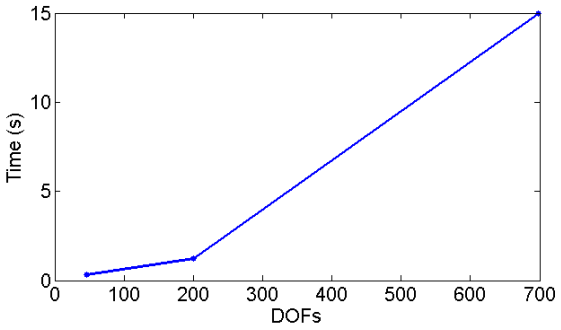

Figure 29: SUPG-CN,Time vs DOFs

Calculation time was recorded for all the above cases. It has been observed that at same DOFs, increasing the timesteps does not add much to the calculation time.(This included only the solution time). But increasing DOFs at constant timestep adds a significant amount of time to the overall computation. This implies that maximum time needed in the algorithm is used in performing matrix calculation. Thus, a spatial discretization should be as course as possible (without compromising the quality of solution), on the other hand, timesteps do not add much to the computational time, but if matrices are stored at all time points, it would generate a significant amount of data.

Summary of Results

Following tables summarizes the recorded observations. First set of values were generated for a constant timesteps = 110, and mesh size was varied. The second set was obtained at a constant mesh size of h=0.2, but with variable timesteps.

Mesh Size (h)	DOFs	u_{max}	u_{min}	time (s)
0.5	46	2.2729	-0.952	0.326432
0.2	200	2.3338	-0.55	1.212877
0.1	698	2.1797	-0.46	14.965806

Timesteps	DOFs	u_{max}	u_{min}	time (s)
55	200	2.3289	-0.5217	1.222386
110	200	2.3338	-0.55	1.212877
220	200	72.3378	-0.5763	1.667751

Galerkin-TG3-2S

A convection dominated problem can be written as follows,

$$u_t = -\nabla \cdot (au)$$

The TG3-2S algorithm can be written as follows,

$$u^p = u^n + \frac{1}{3}\Delta t u_t^n + \alpha(\Delta t)^2 u_{tt}^n$$
$$u^{n+1} = u^n + \Delta t u_t^n + 0.5(\Delta t)^2 u_{tt}^p$$

All boundary fluxes are zero, since on dirichlet inlet, $w = 0$, $\Gamma_N = 0$, $(a \cdot n = 0)$ on outflow boundaries. Therefore, to reduce the terms in the final expression, boundary integrals are directly put equal to 0

and are not included in the calculation. Also, index notation is used to expand expressions.

$$u_{tt} = a_{j,j}a_{i,i}u + 2a_j a_{i,i}u_{,j} + a_j a_i u_{,ij}$$

Weak form of the predictor step,

$$Mu^p = Mu^u + C^T u^n + wu^n_{tt}$$

The last term,

$$\begin{aligned} wu^n_{tt} &= a_{j,j}a_{i,i}uw + 2a_j a_{i,i}u_{,j} + a_j a_i u_{,ij} \\ &= 4Mu^n + 4C^T u^n - 8Mu^n + (4M - 2C^T - K)u^n \end{aligned}$$

The above final step has been written after quite a bit vector and tensor calculations in index notations. After combining all the terms, the predictor step can rewritten as follows,

$$\begin{aligned} Mu^p = Mu^n + \frac{1}{3}\Delta t C^T u^n + \alpha(\Delta t)^2(2C^T - K)u^n \\ A_1 u^p = B_1 u^n + B_2 u^n + B_3 u^n \end{aligned}$$

Similarly, the corrector step can be written as,

$$\begin{aligned} Mu^{n+1} = Mu^n + \Delta t C^T u^n + 0.5(\Delta t)^2(2C^T - K)u^p \\ A_1 u^p = B_1 u^n + B_2 u^n + B_3 u^p \end{aligned}$$

Thus, the system can be solved. This has been implemented as follows,

```
alpha = 1/9;
A1 = M;
B1 = M;
B2 = (C');
B3 = (2*C'-K);
for n=1:1:nStep
    %predictor step
    B       = B1*u(:,n)+B2*dt/3*u(:,n)+B3*alpha*dt*dt*u(:,n);
    A       = A1;
    Ktot    = [A,ADir.';ADir, zeros(nDir,nDir)];
    Ftot    = [B;bDir];
    Tempsol = Ktot\Ftot;
    up      = Tempsol(1:size(A,1));
    %corrector step
    B       = B1*u(:,n) + B2*dt*u(:,n) + B3*dt*dt/2*up;
    Ktot    = [A,ADir.';ADir, zeros(nDir,nDir)];
    Ftot    = [B;bDir];
    Tempsol = Ktot\Ftot;
    u(:,n+1)= Tempsol(1:size(A,1));
end
```

However, a care must be taken while imposing the BC for this problem, since the solver is not in the incremental form, the BC for Γ_2 should be 1 instead of 0 (0 is case of incremental solver since u = const $\implies \Delta u = 0$). Note the implementation of this method currently only supports Galerkin formulation and the written algorithm can only solve pure convection problems.

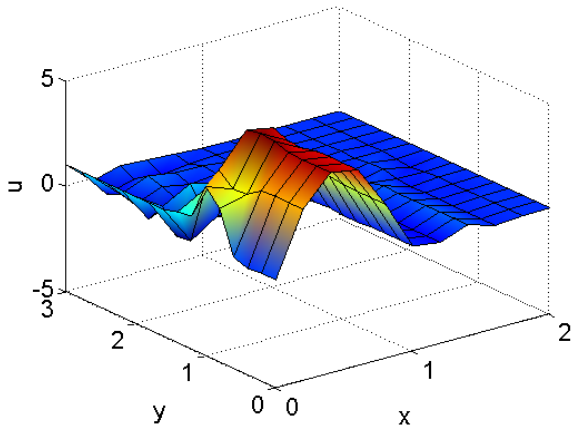

Figure 30: TG3-2S Solution

Results - Galerkin - Pure Convection - TG3-2S

To validate the correctness of the algorithm written so far (even though it is far for being complete), it is tested for a pure convection problem. THe parameters used are ($h = 0.2, tEnd = 0.75, nStep = 110, method = TG3 - 2S, stabilization = none$) The solution is shown here,

As expected, due to lack of stabilization in space, the method is generating some spurious oscillations. But the transport phenomenon is captured quite correctly.

Exercise 3A

The boundary conditions were modified in order to impose the given Dirichlet BC. Also, the corner nodes were added to the flow boundary, so that a leaky cavity problem is obtained, thus the pressure singularities at those corner points would come into picture. For the stokes problem, element type Q2Q1 was chosen due to the following reasons,

1. Q1P0 and Q1Q1 do not satisfy the LBB stability condition.
2. Mini element does satisfy the LBB condition, but the convergence is linear, and an element with a quadratic convergence behavior exists. For example, Q2Q1 has a quadratic convergence.
3. Crouzeix-Raviart element is almost as good as Q2Q1, but due to a pressure singularity, the most accurate description of pressure behavior is needed. Q2Q1 has a bilinear continuous pressure variation whereas Crouzeix-Raviart element has a discontinuous pressure variation. So, Q2Q1 would be able to capture the pressure variation effects more correctly, due to the continuous pressure variation.

A typical absolute viscosity of olive oil is 0.081 Pa.s and density is 800 Kg/m3. Therefore, the kinematic viscosity of 1.0125E-4 was used.

The weak form of the stationary stokes problem is as follows subject to boundary conditions,

$$\int_{\Omega} \boldsymbol{\nabla w} : \nu \boldsymbol{\nabla v} \mathrm{d}\Omega - \int_{\Omega} p \boldsymbol{\nabla} \cdot \boldsymbol{w} \mathrm{d}\Omega = \int_{\Omega} \boldsymbol{w} \cdot \boldsymbol{b} \mathrm{d}\Omega$$

$$\int_{\Omega} q \boldsymbol{\nabla} \cdot \boldsymbol{v} \mathrm{d}\Omega = 0$$

The first equation is obtained from the momentum conservation equation and the second is from mass conservation. The first term in the equation is the viscous bilinear term. Second term is pressure term

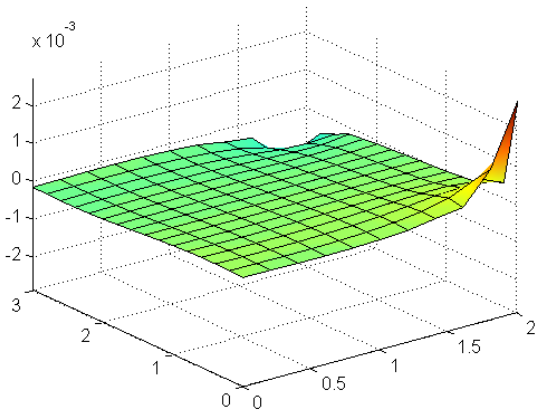

Figure 31: Pressure Plot, Q2Q1, nx=10, ny=15

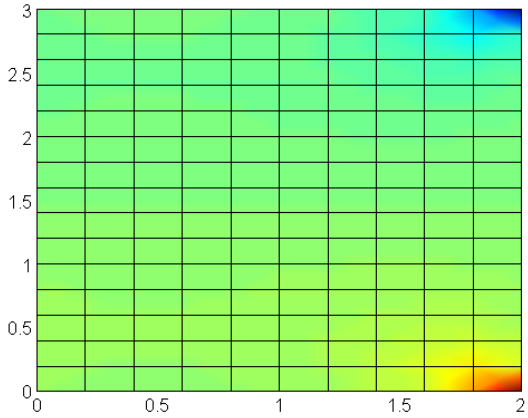

Figure 32: Pressure Plot, Top View, Q2Q1, nx=10, ny=15

and the term on RHS arises due to body forces (volumetric forces). Also, since w is arbitrary, it is chosen to be zero on Dirichlet BC.

The finite element discretization is as shown below,

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

Where, the individual matrices are given as follows,

$$\begin{aligned} K &= \int_\Omega \boldsymbol{\nabla w} : \nu \boldsymbol{\nabla v} \mathrm{d}\Omega \\ G &= -\int_\Omega p \boldsymbol{\nabla} \cdot \boldsymbol{w} \mathrm{d}\Omega \\ G^T &= \int_\Omega p \boldsymbol{\nabla} \cdot \boldsymbol{u} \mathrm{d}\Omega \\ f &= (\boldsymbol{w}, \boldsymbol{b}) + (\boldsymbol{w}, \boldsymbol{t})_{\Gamma_N} - a(\boldsymbol{w}, \boldsymbol{v}_D) \\ h &= -b(\boldsymbol{v}_D, q) \end{aligned}$$

Mesh Size

After a trial run with perfect square element with size h=0.2, pressure plot are shown in Figure 32 and Figure 31. Two things are note from the solution plot. First, the different between maximum and minimum pressure was 0.0056. Second, the pressure gradient in Y direction is almost same as that in X direction. Therefore, to capture the pressure gradient correctly, the element shape should be as square as possible.

After meshing with sizes h=0.15,0.12 and 0.1, the difference between the maximum and minimum pressure was found to be 0.0079, 0.0097 and 0.0113 were observed. This indicates that refinement is assisting in reaching the singularity. Also, apart from the corners, pressure is almost non-varying. Therefore, an option of adaptive refined mesh was considered.

To compare effect of adaptive meshing, same h=0.1 was used for both non-adapted mesh and adapted mesh. The pressure difference in adaptive mesh was 0.2191 and that in non adapted mesh was 0.0113.

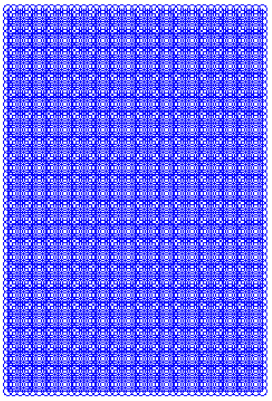

Figure 33: Mesh Plot, Q2Q1, Standard Mesh

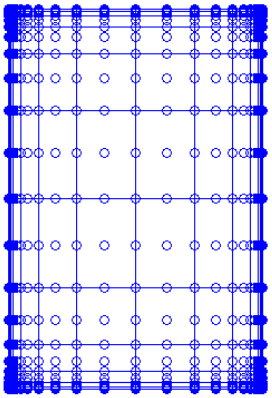

Figure 34: Mesh Plot, Q2Q1, Adaptive Mesh

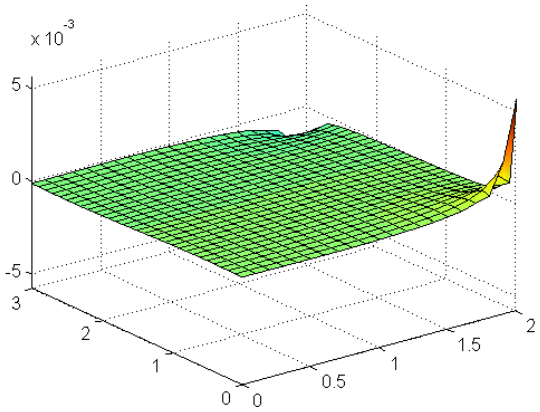

Figure 35: Pressure Plot, Q2Q1, Standard Mesh

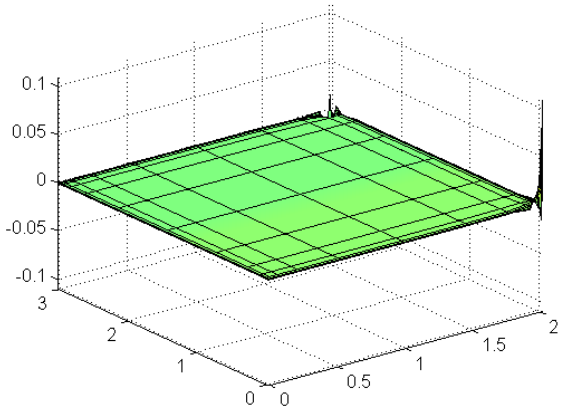

Figure 36: Pressure Plot, Q2Q1, Adaptive Mesh

So adaptive mesh gave better results in terms of pressure but in terms of velocity, the plots are far from accurate. Therefore, adaptive meshing is useful in this case, only when pressure needs to be calculated, otherwise standard meshing is giving better results. Total DOFs = 6056.

Results

The comments are based on the solution plots shown above for standard meshing since, this approach was able to all variations quite correctly. The following observations/comments are made.

1. Since, the fluid at the boundary x=2 is flowing in -Y direction, the fluid in the cavity would circulate in the clockwise direction when looked from above.
2. Due to direction of circulation and boundary flow direction, the pressure at point (2,0) is high and pressure at (2,3) is low.
3. The flow predicted in point 1 is confirmed in velocity plot (X direction). Also, due to symmetry about central axis, the velocity distributions (magnitude wise) are symmetric.
4. Due to no-slip condition at boundaries, the velocity both in X and Y direction is zero at the stationary boundaries. It matches the velocity of boundary along x=2 again due to no-slip
5. Since, pressure gradient is proportional to the viscosity, increasing viscosity results in the increases the difference between the maximum and minimum pressure values.(0.0557 at given viscosity, 0.5569 at 10 times the viscosity)

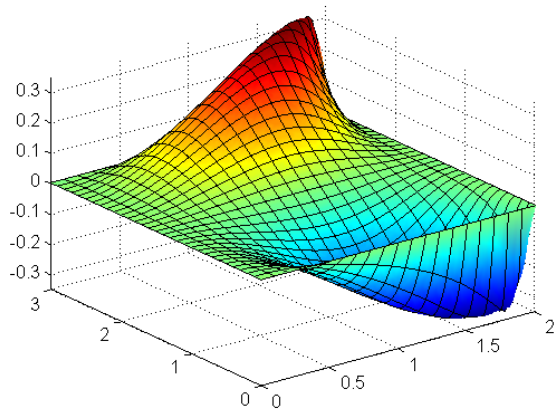

Figure 37: velocity X Plot, Q2Q1, Standard Mesh

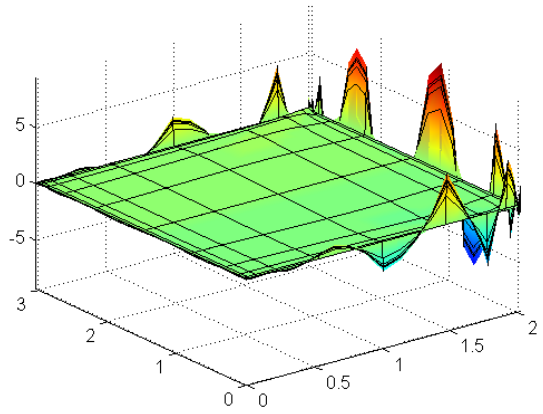

Figure 38: Velocity X Plot, Q2Q1, Adaptive Mesh

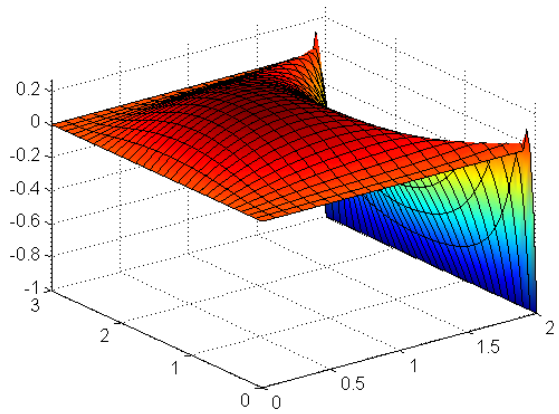

Figure 39: velocity Y Plot, Q2Q1, Standard Mesh

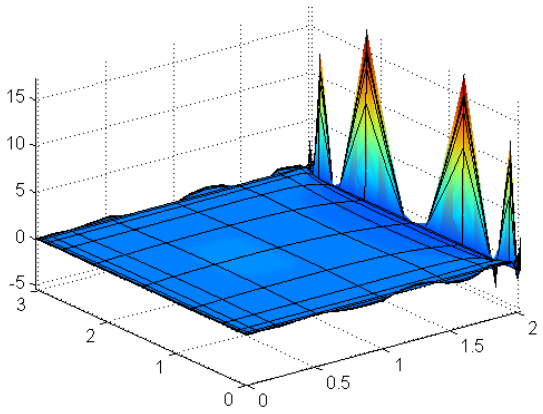

Figure 40: Velocity Y Plot, Q2Q1, Adaptive Mesh

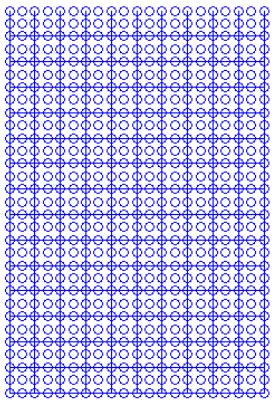

Figure 41: Mesh Plot, Q2Q1, Navier Stokes

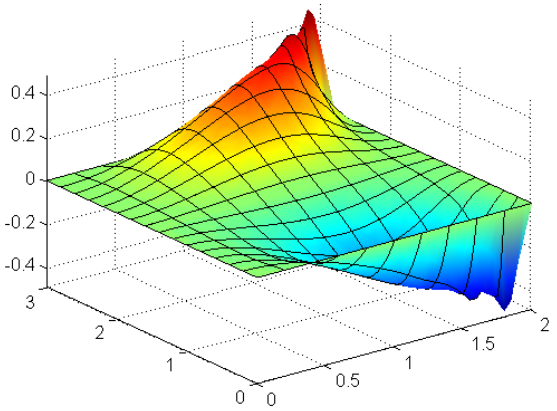

Figure 42: Velocity X Plot, Q2Q1, Navier Stokes

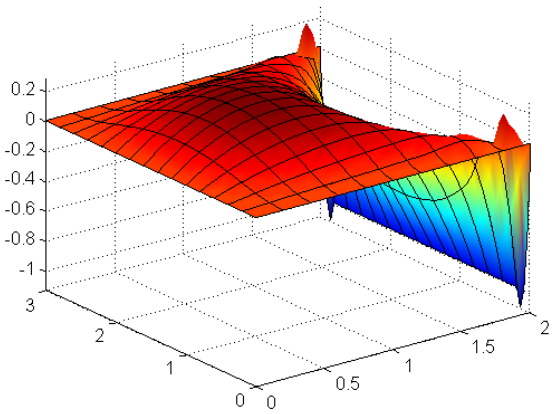

Figure 43: Velocity Y Plot, Q2Q1, Navier Stokes

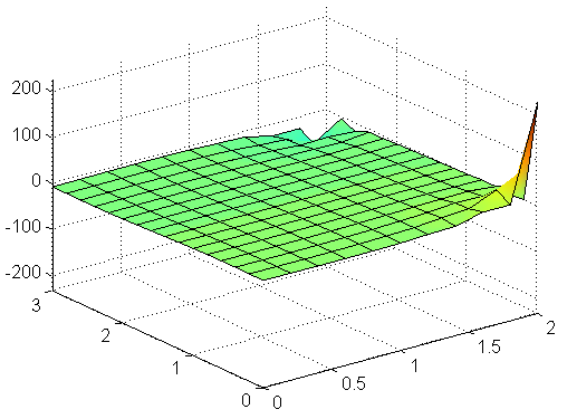

Figure 44: Pressure Plot, Q2Q1, Navier Stokes

Exercise 3B

Since, velocity and Re is given and cannot be changed, ν has to be changed. It was observed that Navier Stokes fails to converge, for Re = 1000 and Re = 2000. This could be true since around this value of Re, the flow does not remain laminar and inertial terms start to dominate over the viscous terms. So stokes solution cannot be reproduced with high Re.

The solution for Re=1, is shown as follows. The pressure difference was found to be 458.6746. To compare this with Stokes solution, $nu = 1$ was substituted in stokes equation to check. For same mesh size (h=0.2), the pressure difference in stokes was 229.3360 which is considerable less compared to Navier Stokes.
The Navier stokes solution at Re = 30, is shown below,

Iterations Comparison

The mesh size of h=0.2 was used with Q2Q1 element. Following table indicates the variation of number of iterations needed vs Re.

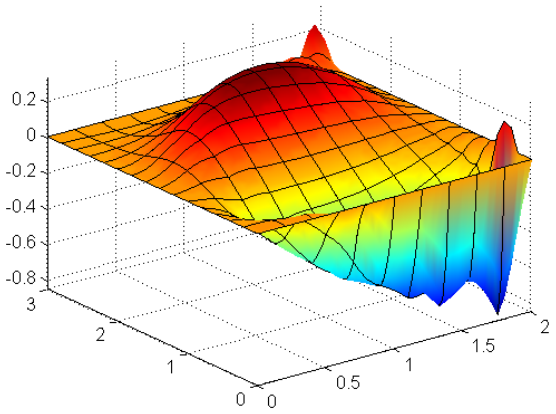

Figure 45: Velocity X Plot, Q2Q1, Navier Stokes

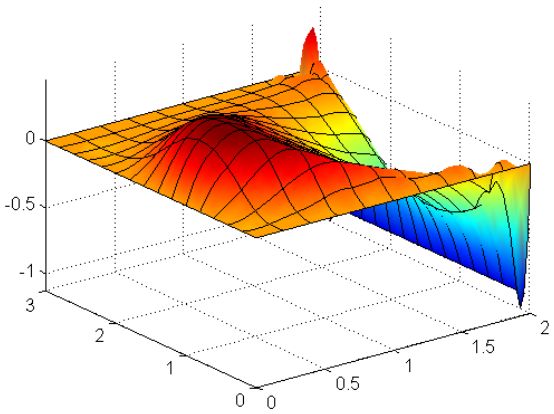

Figure 46: Velocity Y Plot, Q2Q1, Navier Stokes

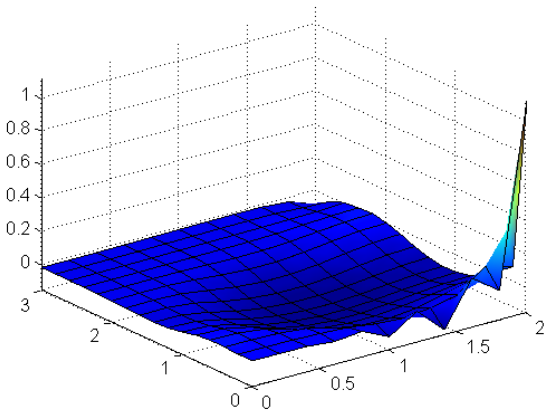

Figure 47: Pressure Plot, Q2Q1, Navier Stokes

Re	Iterations
1	2
10	7
20	11
30	47

Pressure Evolution

Following table indicates the variation of maximum pressure and minimum pressure value. But the location of maximum pressure was found to remain same. Form the values it is evident that as Re increases, the problem no longer remains symmetric. Also, the value of the pressure decreases significantly as Re increases.

Re	Pmax	Pmin
1	226.229	-232.4456
10	2.8846	-1.9494
20	1.3811	-0.3466
30	1.1114	-0.1545

Strength of Velocity

The plots for different magnitude of velocity are shown. It is observed that as Re increases, the circular current that traverse in the domain, tend to take a shorter route, thus the position in the domain where the speed is high moves closer and closer to corner at (2,0). THis generates a low pressure at (0,3) and also since, the connection moves away from the corner at (2,3), pmin increases.

Another interesting phenomenon can be seen in the streamlines plot. some fluid moves in the other direction due to lesser pressure region generation at corner (0,3).

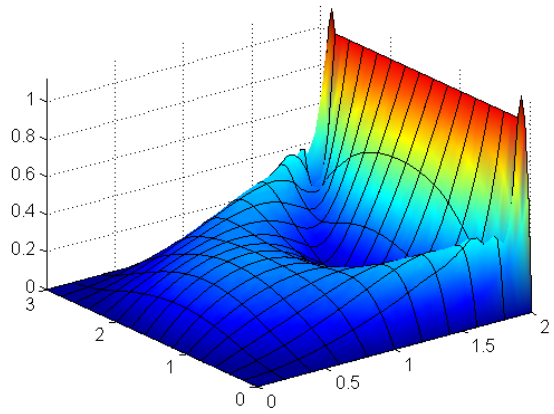

Figure 48: $|a|$, Re=1

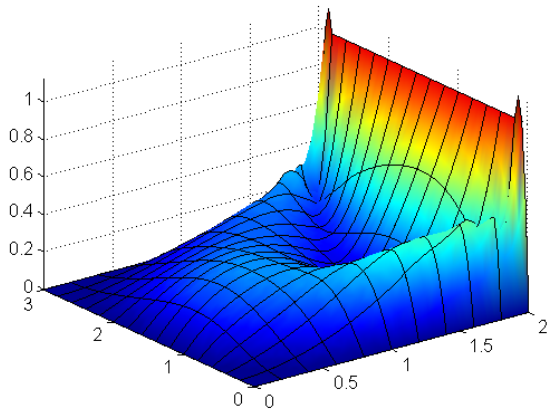

Figure 49: $|a|$, Re=10

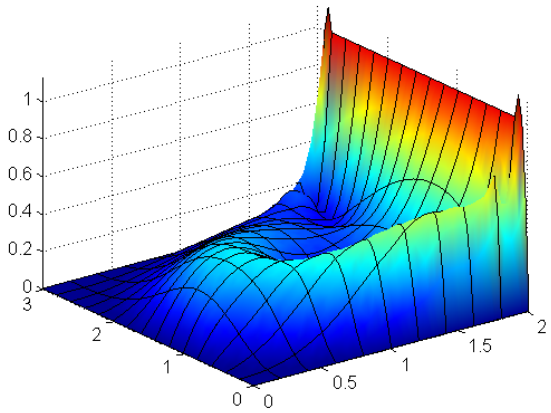

Figure 50: $|a|$, Re=20

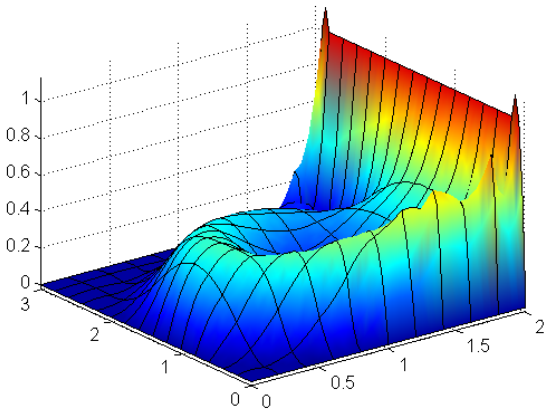

Figure 51: $|a|$, Re=30

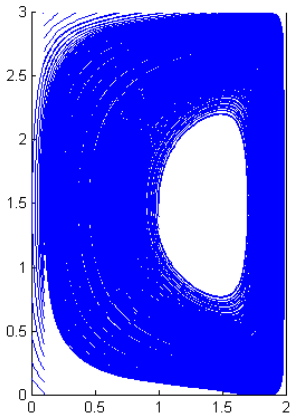

Figure 52: Streamlines, Re=1

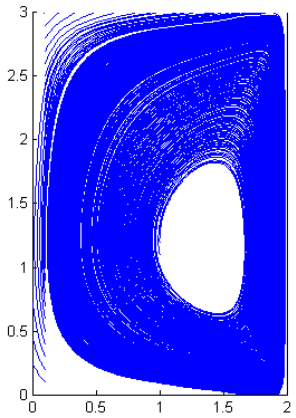

Figure 53: Streamlines, Re=10

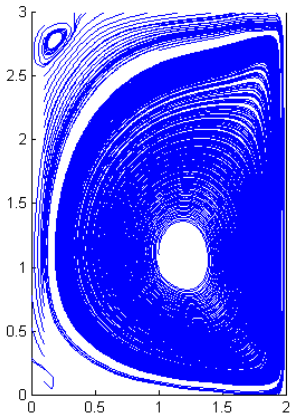

Figure 54: Streamlines, Re=20

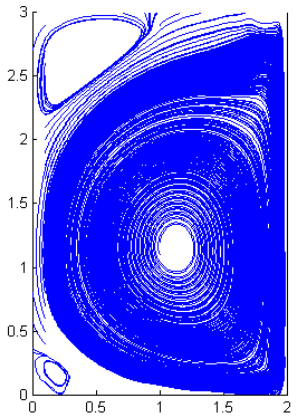

Figure 55: Streamlines, Re=30

THE END