Finite Elements in Fluids: Course Assignment

Corbella Coll, Xavier
xcorbellacoll@gmail.com

May 23, 2016

Exercise 1: Steady state

Consider a steady convection-diffusion-reaction problem with the unknown "ρ", convective term "a", reaction term "σ" and source term "s".

a)

Use weighted residuals to derive the weak form for convection-diffusion-reaction problem. Write down the system you obtain after discretizing this weak form using Galerkin's method and an approximation of the solution:

$$\boldsymbol{\rho}^h(\boldsymbol{x}) = \sum_j \boldsymbol{\rho_j} \boldsymbol{N_j}(\boldsymbol{x})$$

The steady convection-diffusion-reaction equation is:

$$\boldsymbol{a} \cdot \boldsymbol{\nabla}\rho - \boldsymbol{\nabla} \cdot (\nu \boldsymbol{\nabla}\rho) + \sigma\rho = s$$

The first step is to pre-multiplicate it by a test function w:

$$\int_\Omega w\boldsymbol{a} \cdot \boldsymbol{\nabla}\rho d\Omega - \int_\Omega w\boldsymbol{\nabla} \cdot (\nu \boldsymbol{\nabla}\rho)\, d\Omega + \int_\Omega w\sigma\rho d\Omega = \int_\Omega ws d\Omega$$

If we integrate by parts the diffusion term

$$\int_\Omega w\boldsymbol{\nabla} \cdot (\nu \boldsymbol{\nabla}\rho)\, d\Omega = -\int_\Omega \boldsymbol{\nabla} w \cdot (\nu \boldsymbol{\nabla}\rho)\, d\Omega + \int_\Omega \boldsymbol{\nabla}\,(w\nu \boldsymbol{\nabla}\rho)\, d\Omega =$$
$$= -\int_\Omega \boldsymbol{\nabla} w \cdot (\nu \boldsymbol{\nabla}\rho)\, d\Omega + \int_\Gamma w\boldsymbol{n} \cdot (\nu \boldsymbol{\nabla}\rho)\, d\Gamma = -\int_\Omega \boldsymbol{\nabla} w \cdot (\nu \boldsymbol{\nabla}\rho)\, d\Omega + \int_\Gamma w h_N d\Gamma$$

then we get:

$$\int_\Omega w\boldsymbol{a} \cdot \boldsymbol{\nabla}\rho\, d\Omega + \int_\Omega \boldsymbol{\nabla} w \cdot (\nu \boldsymbol{\nabla}\rho)\, d\Omega + \int_\Omega w\sigma\rho\, d\Omega = \int_\Omega ws\, d\Omega + \int_\Gamma w h_N d\Gamma$$

Discretizing ($\rho \approx \rho^h$) and using Galerkin's method ($w = N$):

$$\int_\Omega N_i \boldsymbol{a} \cdot \boldsymbol{\nabla} N_j \rho_j \, d\Omega + \int_\Omega \boldsymbol{\nabla} N_i \cdot (\nu \boldsymbol{\nabla} N_j \rho_j) \, d\Omega + \int_\Omega N_i \sigma N_j \rho_j \, d\Omega = \int_\Omega N_i s \, d\Omega + \int_\Gamma N_i h_N d\Gamma$$

This can be expressed as the following system of equations:

$$\boldsymbol{D\rho + A\rho + R\rho = f}$$

Where D is the diffusion matrix, A the convection matrix and R the reaction matrix, and f the vector of "forces".

b)

Modify the Matlab code to solve this problem using Galerkin formulation with linear elements, a spatial discretization $h = 0.2$ and:

1. $\boldsymbol{a = 1, \quad \nu = 10^{-3}, \quad \sigma = 10^{-3}, \quad s = 0}$
2. $\boldsymbol{a = 10^{-3}, \quad \nu = 10^{-3}, \quad \sigma = 1, \quad s = 0}$
3. $\boldsymbol{a = 1, \quad \nu = 10^{-3}, \quad \sigma = 0, \quad s = 1}$
4. $\boldsymbol{a = 1, \quad \nu = 10^{-3}, \quad \sigma = 1, \quad s = 0}$

The boundary conditions are:

$$\boldsymbol{\rho = 1 \quad in \Gamma_2}$$

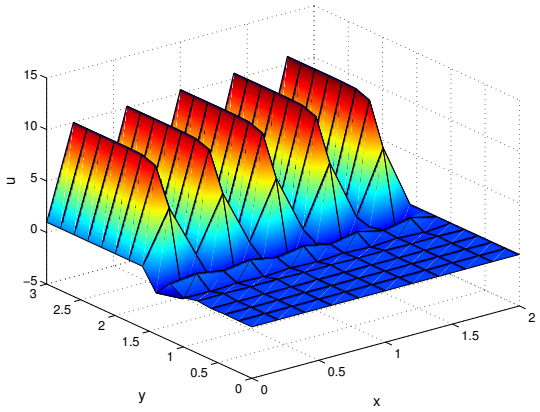
$$\boldsymbol{\rho = 0 \quad in \Gamma_4}$$

In case that the solution is not acceptable in any of the cases above propose how you would overcome the problem without modifying the Galerkin approach nor changing a, ν, σ and s.

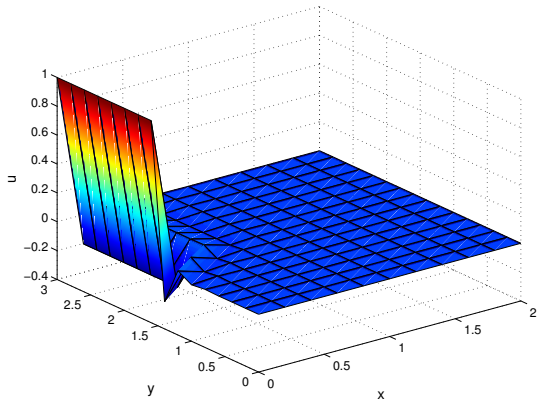
Since the code was already able to use Galerkin's method, it only had to be modified to add the reaction term, change the geometry ($\Omega = (0,2)x(0,3) \in \mathbb{R}^2$) and use the parameters and boundary conditions stated above. The result obtained for the different combinations of coefficients are depicted in figure 1. In combinations 1,3 and 4 we can observe instabilities and oscillations which appear because it is a convection dominated problem. The Peclet number

$$Pe = \frac{|a|h}{2\nu}$$

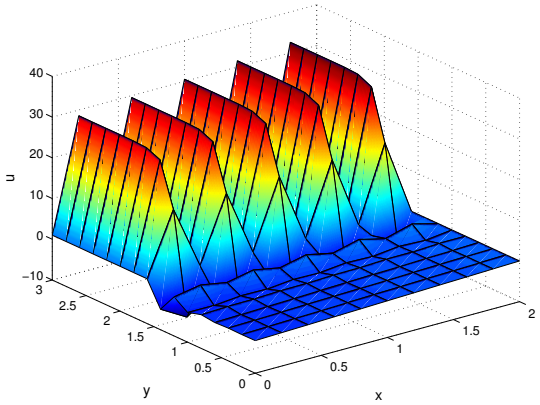
for combinations 1,3 and 4 is $Pe = 100 > 1$. We can improve the results obtained if working with $Pe < 1$. However, doing so would require to use an element size $h < 2 \cdot 10^{-3}$ which would require a huge amount of computational resources. In the case of set 2, $Pe = 0.2$ and the solution does not exhibit instabilities due to convection. However, the problem is dominated by the reaction term and the result obtained in the boundary layer is quite bad and could be improved refining the mesh there. For example, the results obtained for $h = 0.05$ are depicted in figure 2.



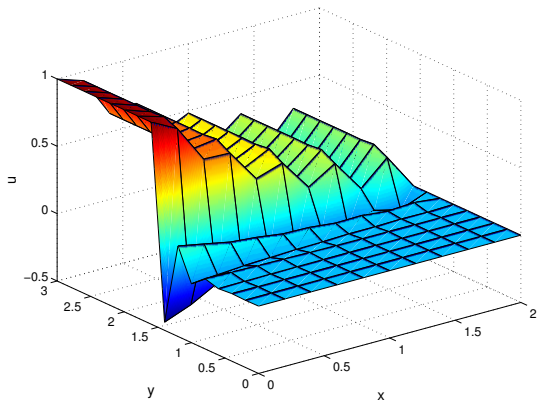
(a) Combination of coefficients 1



(b) Combination of coefficients 2



(c) Combination of coefficients 3



(d) Combination of coefficients 4

Figure 1: Results obtained using Galerkin's method and h=0.2

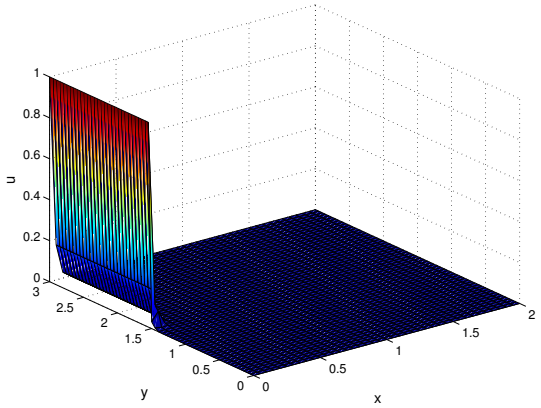


Figure 2: Results for set 2 using $h = 0.05$.

c)

Choose one of the set of parameters for which using SUPG and GLS formulations would have sense and explain why. Write the modifications down and how each method is obtained step by step. Explain the modifications introduced in the Matlab code and compare the results.

As commented before, for sets 1,3 and 4 we have $Pe = 100 > 1$. Thus, in this case we cannot obtain an stable solution via Galerkin's method when using the mesh proposed in the statement of the problem. For this mesh, Galerkin's method is not able to solve for the convection and spurious oscillations appear. These oscillations can be avoided either refining the mesh (as said in section b this would be too expensive) or using stabilization techniques such as SUPG and GLS formulations. These formulations introduce a stabilization term in the lhs of the weak form:

$$\sum_e \int_{\Omega_e} \mathcal{P}(w)\, \tau \mathcal{R}(\rho)\, d\Omega$$

Where $R(\rho)$ is the residual of the convection-diffusion-reaction

$$\mathcal{R}(\rho) = \boldsymbol{a} \cdot \boldsymbol{\nabla}\rho - \boldsymbol{\nabla} \cdot (\nu \boldsymbol{\nabla}\rho) + \sigma\rho - s = \mathcal{L}(\rho) - s$$

τ is a stabilization parameter and $\mathcal{P}(w)$ depends on the method used:

$$\begin{aligned} SUPG \quad &\rightarrow \quad \mathcal{P}(w) = \boldsymbol{a} \cdot \boldsymbol{\nabla} w \\ GLS \quad &\rightarrow \quad \mathcal{P}(w) = \mathcal{L}(w) = \boldsymbol{a} \cdot \boldsymbol{\nabla} w - \boldsymbol{\nabla} \cdot (\nu \boldsymbol{\nabla} w) + \sigma w \end{aligned}$$

The parameter τ used will be :

$$\tau = \frac{h}{2a}\left(1 + \frac{9}{Pe^2} + \left(\frac{h}{2a}\sigma\right)^2\right)^{-\frac{1}{2}}$$

The changes done in the matlab code are the calculation of τ in function 'FEM_system'

```
sif method == 0
    % Galerkin
    tau = 0;
elseif method == 1
    % SUPG
    Pe = a*h/(2*nu);
    tau_p = h/(2*a)*(1 + 9/Pe^2+(h/(2*a)*sigma)^2)^(-1/2);
    disp(strcat('Recommended stabilization parameter = ',num2str(tau_p)));
    tau = cinput('Stabilization parameter',tau_p);
    if isempty(tau)
        tau = tau_p;
    end
elseif method == 2
    % GLS
    Pe = a*h/(2*nu);
    tau_p = h/(2*a)*(1 + 9/Pe^2+(h/(2*a)*sigma)^2)^(-1/2);
    disp(strcat('Recommended stabilization parameter = ',num2str(tau_p)));
    tau = cinput('Stabilization parameter',tau_p);
    if isempty(tau)
```

```
        tau = tau_p;
    end
else
    error ('Unavailable method')
end
```

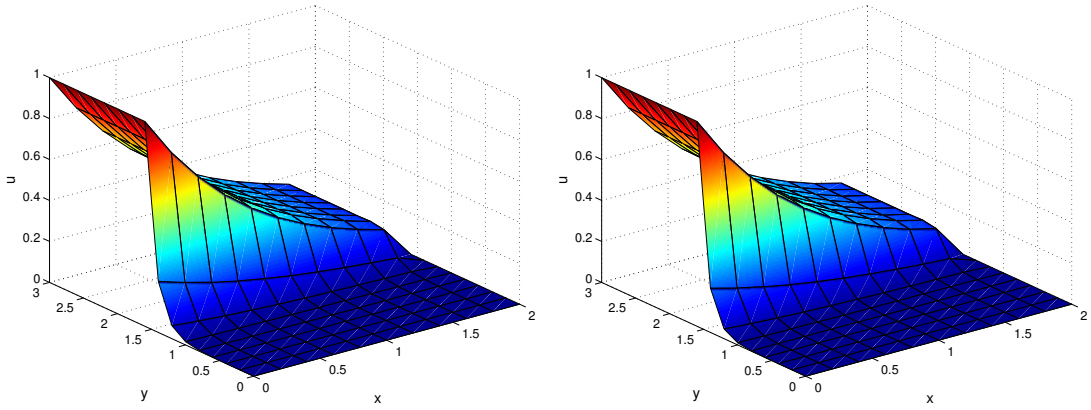
and the definition of FEM matrices for the stabilization methods in function 'eleMat':

```
% Loop on Gauss points (computation of integrals on the current element)
for ig = 1:ngaus
    N_ig = N(ig,:);
    Nxi_ig = Nxi(ig,:);
    Neta_ig = Neta(ig,:);
    Jacob = [Nxi_ig*(Xe(:,1))    Nxi_ig*(Xe(:,2))
        Neta_ig*(Xe(:,1))    Neta_ig*(Xe(:,2))];
    dvolu = wgp(ig)*det(Jacob);
    res = Jacob\[Nxi_ig;Neta_ig];
    Nx = res(1,:);
    Ny = res(2,:);

    if method == 0
        % Galerkin
        Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny)+sigma*N_ig'*
    N_ig)*dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux);
        fe = fe + N_ig'*(f_ig*dvolu);
    elseif method == 1
        % SUPG
        Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + sigma*N_ig'*
    N_ig...
            +tau*(ax*Nx+ay*Ny)'*((ax*Nx+ay*Ny)+sigma*N_ig))*dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux);
        fe = fe + (N_ig+tau*(ax*Nx+ay*Ny))'*(f_ig*dvolu);
    else
        % GLS
        Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + sigma*N_ig'*
    N_ig...
            +tau*(ax*Nx+ay*Ny + sigma*N_ig)'*((ax*Nx+ay*Ny)+sigma*N_ig))*
    dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux);
        fe = fe + (N_ig+tau*(ax*Nx+ay*Ny))'*(f_ig*dvolu);
    end
end
```

The term $-\boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}w)$ has not been included since we are dealing with linear shape functions.

If we solve the fourth set of parameters ($a=1, \quad \nu=10^{-3}, \quad \sigma=1, \quad s=0$) using SUPG and GLS methods we obtain solutions which are much better than Galerkin's solution and do not show spurious oscillations (see figure 3).



(a) SUPG (b) GLS

Figure 3: Results obtained using SUPG and GLS formulations for set of parameters 4

d)

Solve the problem with $\rho = 2$ in Γ_2 and $\rho = 1$ in Γ_4. Modify the BC in Γ_4 to impose Neumann BC so that the same solution is obtained.

The results obtained for the fourth set of parameters and $\rho = 2$ in Γ_2 and $\rho = 1$ in Γ_4 are depicted in figure 4.

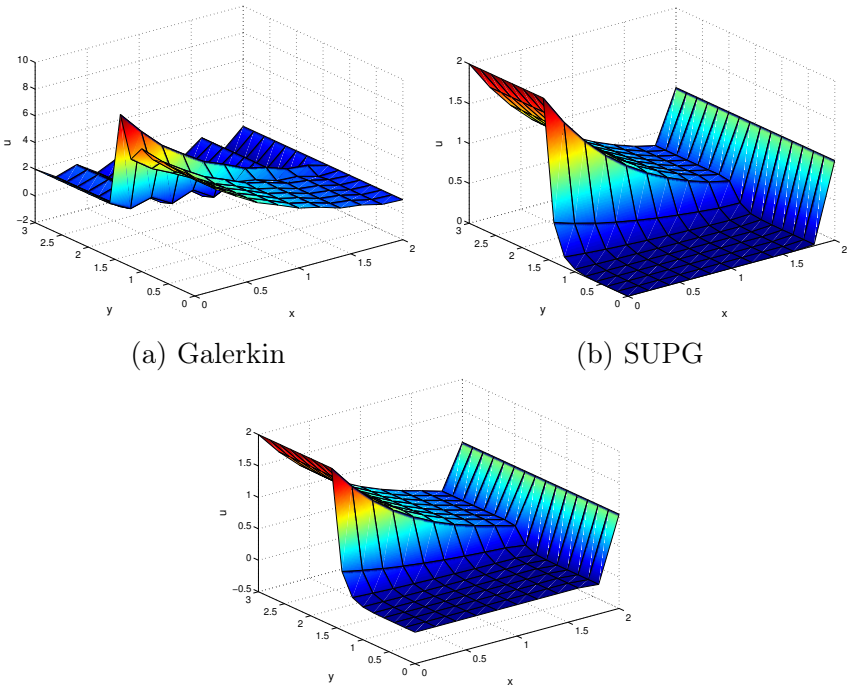
If we want to obtain the same results but imposing Neumann boundary conditions in Γ_4, we first must know which are the fluxes of u in Γ_4 that are obtained when imposing $\rho = 1$. These values of fluxes are the Lagrange multipliers obtained when solving the system with boundary conditions. Thus, we must store these fluxes when solving with $\rho = 1$ in Γ_4:

```
%Here we recover the neumann bc that shoulb be imposed in BC4 to
        %get the same results that for the boundary conditions imposed in
        %Ex1d using Dirichlet bc
        BC4 = size(nodesBC4,1);
        BC2 = size(nodesBC2,1);
        new_neumann = zeros(BC4,2);
        for i = 1:BC4
            aux = i+BC2;
            new_neumann(i,1)=nodesBC4(i);
            new_neumann(i,2)=multip(aux);
        end
        save('bc2_neumann','new_neumann')
```

And use these fluxes when solving with Neumann boundary conditions in Γ_4:

```
case 2
        % 1 in nodesBC2 and Neumann in nodesBC4
        % Boundary condition matrix
        C = [nodesBC2, 2*ones(length(nodesBC2),1)];
        load('bc2_neumann');
        for count=1:size(nodesBC4,1)
            f(nodesBC4(count)) = f(nodesBC4(count)) - new_neumann(count,2);
        end;
```

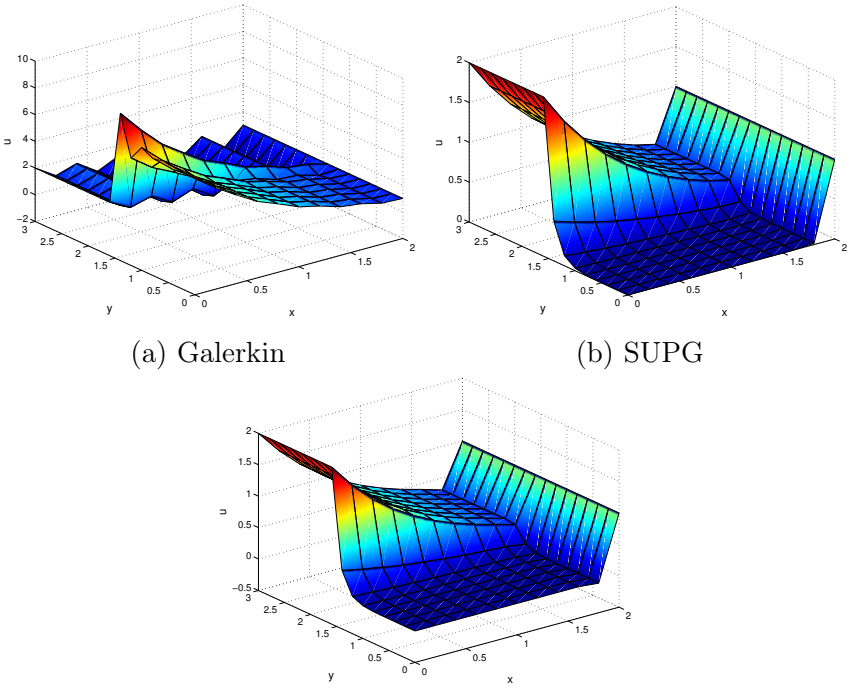
The results obtained are the same that when Dirichlet boundary conditions were applied (see figures 4 and 5).



(a) Galerkin (b) SUPG

(c) GLS

Figure 4: Results obtained for set of parameters 4 and $\rho = 2$ in Γ_2 and $\rho = 1$ in Γ_4.



(a) Galerkin (b) SUPG

(c) GLS

Figure 5: Results obtained for set of parameters 4 and $\rho = 2$ in Γ_2 and Neumann in Γ_4.

Exercise 2

Now we include the transient term and initial boundary conditions $\boldsymbol{\rho_0}(\boldsymbol{x},\boldsymbol{0}) = \frac{\boldsymbol{1}}{\boldsymbol{2}}(\boldsymbol{2}-\boldsymbol{x})$ in $\boldsymbol{\Omega}$. BC will be homogeneous Dirichlet BC $\boldsymbol{\rho = 1}$ in $\boldsymbol{\Gamma_2}$ and $\boldsymbol{\rho = 0}$ in $\boldsymbol{\Gamma_4}$. The vector $\boldsymbol{a}(\boldsymbol{x},\boldsymbol{y}) = (-\boldsymbol{x},-\boldsymbol{y})$.

a)

Discretize the problem above using the formulation for space that you consider more appropiate among the ones described in Exercise 1. For the time discretization use 2 of the following methods:

1. **Crank-Nicholson method**
2. **Two step third order TG method**
3. **Pade approximation of order R22**

State the system that has to be solved at each time step.

The time discretization schemes chosen are Crank-Nicholson (CN) method and Pade approximation of order R_{22}. Both these schemes can be obtained as Pade implicit schemes (CN is a Pade approximation of order R_{11}). Implicit Pade methods can be expressed as:

$$\frac{\Delta\boldsymbol{\rho}}{\Delta t} - \boldsymbol{W}\Delta\boldsymbol{\rho}_t = \boldsymbol{w}\rho_t^n \tag{1}$$

For Crank-Nicholson:

$$\Delta\boldsymbol{\rho} = \begin{bmatrix}\rho^{n+1} & \rho^n\end{bmatrix} \quad \boldsymbol{W} = \frac{1}{2} \quad \boldsymbol{w} = 1$$

For R_{22}:

$$\Delta\boldsymbol{\rho} = \begin{bmatrix}\rho^{n+\frac{1}{2}} & \rho^n \\ \rho^{n+1} & \rho^{n+\frac{1}{2}}\end{bmatrix} \quad \boldsymbol{W} = \frac{1}{24}\begin{bmatrix}7 & -1 \\ 13 & 5\end{bmatrix} \quad \boldsymbol{w} = \frac{1}{2}\begin{bmatrix}1 \\ 1\end{bmatrix}$$

The equation that we are dealing with is:

$$\rho_t + \boldsymbol{a}\cdot\boldsymbol{\nabla}\rho - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}\rho) + \sigma\rho = s$$

Thus:

$$\rho_t = -\boldsymbol{a}\cdot\boldsymbol{\nabla}\rho + \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}\rho) - \sigma\rho + s \tag{2}$$

Adding equation 2 into 1:

$$\frac{\Delta\boldsymbol{\rho}}{\Delta t} - \boldsymbol{W}\left[\Delta\boldsymbol{s} - \left[\boldsymbol{a}\cdot\boldsymbol{\nabla} - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}) + \sigma\right]\Delta\boldsymbol{\rho}\right] = \boldsymbol{w}\left[s^n - \left[\boldsymbol{a}\cdot\boldsymbol{\nabla} - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}) + \sigma\right]\rho^n\right]$$

Thus:

$$\frac{\Delta\boldsymbol{\rho}}{\Delta t} + \boldsymbol{W}\left[\boldsymbol{a}\cdot\boldsymbol{\nabla} - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}) + \sigma\right]\Delta\boldsymbol{\rho} = \boldsymbol{W}\Delta\boldsymbol{s} + \boldsymbol{w}\left[s^n - \left[\boldsymbol{a}\cdot\boldsymbol{\nabla} - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}) + \sigma\right]\rho^n\right] \tag{3}$$

Two different spatial discretizations have been used: Galerkin and SUPG. The Galerkin discretization is obtained pre-multiplicating equation 3 by a test function v and integrating by parts:

$$\int_\Omega v\frac{\Delta\boldsymbol{\rho}}{\Delta t}\,d\Omega + \int_\Omega v\boldsymbol{a}\cdot\boldsymbol{\nabla}\left(\boldsymbol{W}\Delta\boldsymbol{\rho}\right)\,d\Omega + \int_\Omega v\boldsymbol{\nabla}\cdot\left(\nu\boldsymbol{\nabla}\left(\boldsymbol{W}\Delta\boldsymbol{\rho}\right)\right)\,d\Omega + \int_\Omega v\sigma\boldsymbol{W}\Delta\boldsymbol{\rho}\,d\Omega =$$
$$= \int_\Omega v\left(\boldsymbol{W}\Delta\boldsymbol{s} + \boldsymbol{w}s^n\right)\,d\Omega - \int_\Omega v\boldsymbol{a}\cdot\boldsymbol{\nabla}\left(\boldsymbol{w}\rho^n\right)\,d\Omega + \int_\Omega v\boldsymbol{\nabla}\cdot\left(\nu\boldsymbol{\nabla}\left(\boldsymbol{w}\rho^n\right)\right)\,d\Omega -$$
$$- \int_\Omega v\sigma\boldsymbol{w}\rho^n\,d\Omega + \int_{\Gamma_N} v\left(\boldsymbol{W}\Delta\boldsymbol{h} + \boldsymbol{w}h^N\right)\,d\Gamma$$

The system of equations arising from this weak form is:

$$A\Delta\boldsymbol{\rho} = b$$

This system of equations must be solved at every time step and its size will be different for R_{22} and CN. This system must be subjected to the constraints added to the problem (Lagrange multipliers for Dirichlet BC and Neumann terms added to the fluxes). If BC or source terms (or material parameters) are time dependent, terms A and b will change every iteration. If A does not change, we can store its LU decomposition to reduce the time needed to solve the problem.

In order to add stabilization to the problem, we add the following term to the rhs of the integral form:

$$\sum_e \int_{\Omega_e} \tau\mathcal{P}\left(v\right)\mathcal{R}\left(\Delta\boldsymbol{\rho}\right)\,d\Omega$$

Where the residual is:

$$\mathcal{R}\left(\Delta\boldsymbol{\rho}\right) = \frac{\Delta\boldsymbol{\rho}}{\Delta t} - \boldsymbol{W}\boldsymbol{\rho}_t - \boldsymbol{w}\rho_t^n = \frac{\Delta\boldsymbol{\rho}}{\Delta t} + \boldsymbol{W}\mathcal{L}\left(\Delta\boldsymbol{\rho}\right) - \boldsymbol{W}\left[s^n - \mathcal{L}\left(\rho^n\right)\right]$$

The term $\mathcal{P}(v)$ is:

$$SUPG \quad \rightarrow \quad \mathcal{P}\left(v\right) = \boldsymbol{W}\left(\boldsymbol{a}\cdot\boldsymbol{\nabla}v\right)$$

The integral form for the SUPG problem will be:

$$\int_\Omega v\frac{\Delta\boldsymbol{\rho}}{\Delta t}\,d\Omega + \int_\Omega v\boldsymbol{a}\cdot\boldsymbol{\nabla}\left(\boldsymbol{W}\Delta\boldsymbol{\rho}\right)\,d\Omega + \int_\Omega v\boldsymbol{\nabla}\cdot\left(\nu\boldsymbol{\nabla}\left(\boldsymbol{W}\Delta\boldsymbol{\rho}\right)\right)\,d\Omega + \int_\Omega v\,\sigma\boldsymbol{W}\Delta\boldsymbol{\rho}\,d\Omega$$
$$+\sum_e \int_{\Omega_e} \tau\boldsymbol{W}\left(\boldsymbol{a}\cdot\boldsymbol{\nabla}v\right)\frac{\Delta\boldsymbol{\rho}}{\Delta t}\,d\Omega + \sum_e \int_{\Omega_e} \tau\boldsymbol{W}^T\boldsymbol{W}\left(\boldsymbol{a}\cdot\boldsymbol{\nabla}v\right)\left(\boldsymbol{a}\cdot\boldsymbol{\nabla} + \sigma - \boldsymbol{\nabla}\cdot\left(\nu\boldsymbol{\nabla}\right)\right)\Delta\boldsymbol{\rho}\,d\Omega =$$
$$= \int_\Omega v\left(\boldsymbol{W}\Delta\boldsymbol{s} + \boldsymbol{w}s^n\right)\,d\Omega - \int_\Omega v\boldsymbol{a}\cdot\boldsymbol{\nabla}\left(\boldsymbol{w}\rho^n\right)\,d\Omega + \int_\Omega v\boldsymbol{\nabla}\cdot\left(\nu\boldsymbol{\nabla}\left(\boldsymbol{w}\rho^n\right)\right)\,d\Omega -$$
$$- \int_\Omega v\sigma\boldsymbol{w}\rho^n\,d\Omega + \int_{\Gamma_N} v\left(\boldsymbol{W}\Delta\boldsymbol{h} + \boldsymbol{w}h^N\right)\,d\Gamma + \sum_e \int_{\Omega_e} \tau\boldsymbol{W}\left(\boldsymbol{a}\cdot\boldsymbol{\nabla}v\right)\boldsymbol{w}s^n\,d\Omega +$$
$$+\sum_e \int_{\Omega_e} \tau\boldsymbol{W}^T\boldsymbol{W}\left(\boldsymbol{a}\cdot\boldsymbol{\nabla}v\right)\Delta\boldsymbol{s}\,d\Omega - \sum_e \int_{\Omega_e} \tau\boldsymbol{W}\left(\boldsymbol{a}\cdot\boldsymbol{\nabla}v\right)\boldsymbol{w}\left(\boldsymbol{a}\cdot\boldsymbol{\nabla} + \sigma - \boldsymbol{\nabla}\cdot\left(\nu\boldsymbol{\nabla}\right)\right)\Delta\boldsymbol{\rho}\,d\Omega$$

The matrix τ used is (Soulaïmani and Fortin (1994), Codina (2000)):

$$\tau = \left[\frac{\boldsymbol{W}^{-1}}{\Delta t} + \left(\frac{2a}{h} + \frac{4\nu}{h^2} + \sigma\right)\boldsymbol{I}\right]^{-T}\boldsymbol{W}^{-1}$$

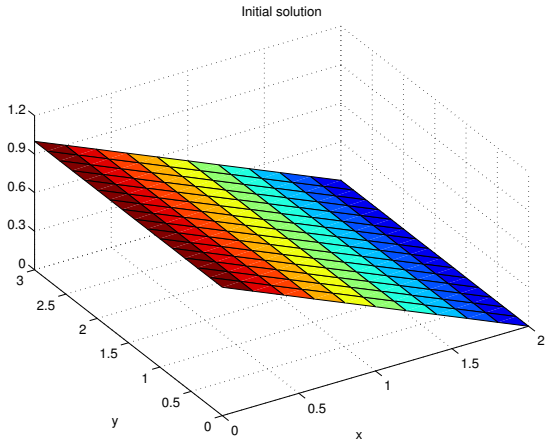


Figure 6: Initial conditions.

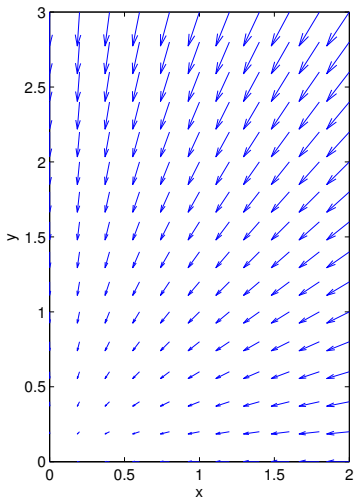


Figure 7: Velocity $\boldsymbol{a}$.

b)

Modify the code and use it to solve the problems described above. Write the procedure used to implement these methods.

The methods described above have been implemented for the unsteady convection-diffusion-reaction equation with steady source term and using linear 4-node and quadratic 9-node quadrilateral elements. First, it reads the input parameters (order of elements, mesh size h, ν,σ, time step Δt and end time), creates the mesh and set initial and boundary conditions ($\rho = 1$ **in** Γ_2 and $\rho = 0$ **in** Γ_4). The initial conditions used are $\rho_0(x,0) = \frac{1}{2}(2-x)$ (see figure 6). We also defined the convection velocity field $\boldsymbol{a}$ (see figure 7).

The user has the option to select which time and spatial discretization schemes use:

```
disp(' ')
disp('There are two integration schemes available ')
disp('      [0] = Crank-Nicolson');
disp('      [1] = R22');
disp('and three methods to perform spatial discretization')
```

```
disp('      [0] = Galerkin');
disp('      [1] = Galerkin/Least-Squares');
disp('      [2] = Streamline-Upwind Petrov-Galerkin (SUPG)');
disp(' ')
d_temp = cinput('Choose a method to perform time integration = ',0);
d_esp = cinput('and another one for the spatial discretization  = ',0);

% Matrices for the time integration
if d_temp == 0
    method = 'CN + ';
    W = 1/2;
    w = 1;
    beta = [0,1];
elseif d_temp == 1
    method = 'R22 + ';
    W = (1/24)*[7 -1; 13 5];
    w = [1/2; 1/2];
    beta = [0,1/2,1];
else
    error('Unavailable time integration scheme')
end
```

The spatial discretization is implemented in files Galerkin.m and SUPG.m which create the matrices of the system and solve it for every time step constrained to the boundary conditions. This files can be found at the Appendix.

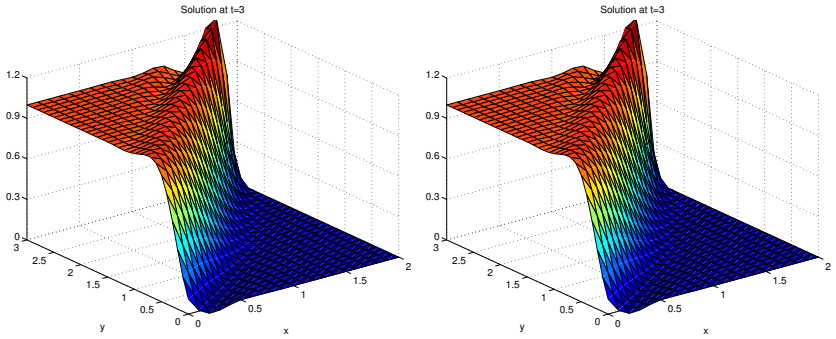
c)

Compute the solutions using linear elements in the time frame $t \in [0, t^n]$. Choose t^n with your own criterion to analyze the entire transient model. Do a sensitivity analysis of the mesh dependency using linear elements and comment on the performance. A initial discretization time of 110 is proposed. Analyze how the solution behaves when this time discretization is changed below and above the proposed value. Finally, comment on the computational cost of these approaches.

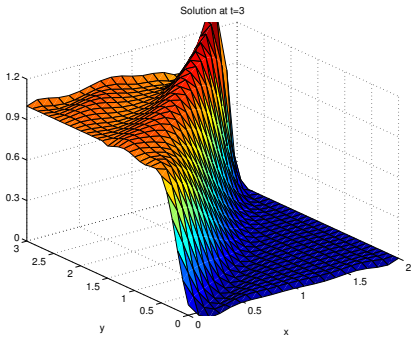
The convection velocity field used (figure 7) is larger than the ones used in Exercise one, leading to larger Péclet number. This makes harder to obtain stabilization in the spatial discretization of the problem.

If we wanted to work, for example, with $\nu = 10^{-3}$, $\sigma = 0$ and $s = 0$ (third set of coefficients but without source term), then when using a mesh size $h = 0.1$, which is a fine mesh that requires huge computational resources when working with personal computers, we obtain a maximum Peclet number $Pe_{max} = 18.03$ and the spatial discretization of the problem would introduce oscillations. Thus, for this set of coefficients, the spatial discretization of the problem cannot achive stable results whitout using a very fine mesh. In order to study a stable problem without having to use very large computational resources, the values of ν used will be larger.

First we will study the effects of h and dt for CN and Galerkin methods for $\nu = 0.3$, $\sigma = 0$ and no source term. The results obtained for $h = 0.1$ ($Pe_{max} = 0.6$) after 3 seconds when using 109 time steps ($C = 1$) and 27 time steps ($C = 4$) are depicted in figure 8. The



(a) 109 time steps ($C = 1$) (b) 27 time steps ($C = 4$)



(c) 7 time steps ($C = 15.45$)

Figure 8: Results obtained for $\nu = 0.3$, $\sigma = 0$ using CN and Galerkin's method.

temporal evolution of the solution at the lower left corner is depicted in figure 9. As can be seen in the results, for this Peclet number, the steady solution is obtained after 2.2 s and the combination of Galerkin and CN is able to fully solve the problem for any Courant number or number of time steps (The results obtained for $C = 15.45$ are not very accurate since we are only using 7 time steps). The same conclusions could be obtained for R_{22} and SUPG, and agree with the fact that CN and R_{22} methods are unconditionally stable.

However, when working with larger Peclet numbers, the spatial discretization can lead to instabilities due to the strong convection of the problem. For example, when working with $h = 0.2$ and $\nu = 0.1$ ($Pe = 3.6$), the problem cannot be solved for any number of time steps.

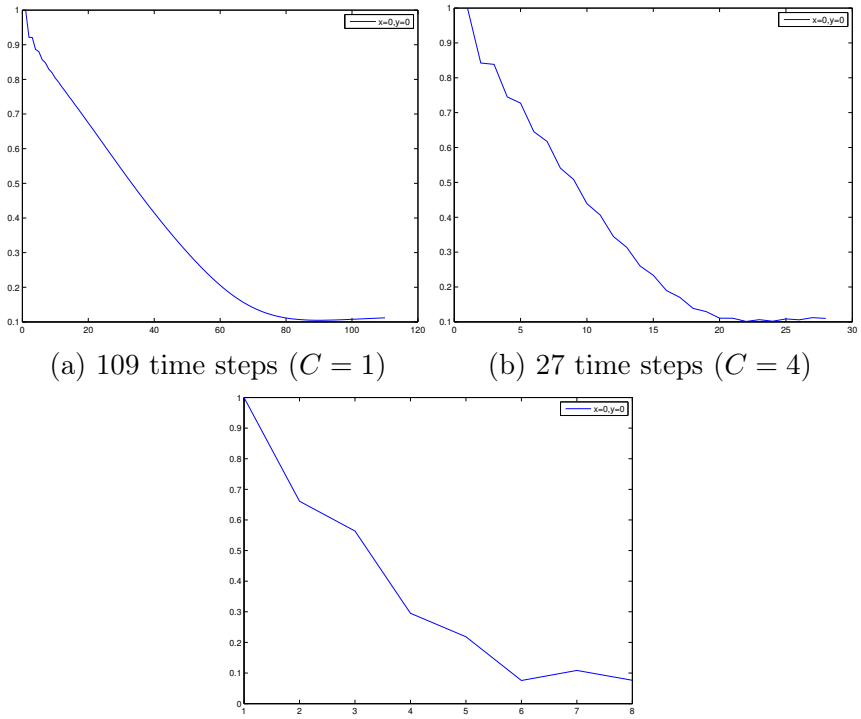
The results obtained for this problem could be improved using formulations that provide better stabilization, such as Least-Squares or Discountinous Galerkin methods.

d)

Include quadratic elements and comment on the differences in the solutions, if any, compared with the linear elements.

Quadratic elements have been included. To do so, they have been added to the function 'ShapeFunc.m' and the term $\nu\nabla\cdot\nabla$ has been added to the residual for SUPG formulation. The results obtained when using quadratic elements for $Pe > 1$ (see figure 10) are similar to those obtained for linear elements when using SUPG formulation, but smoother. The

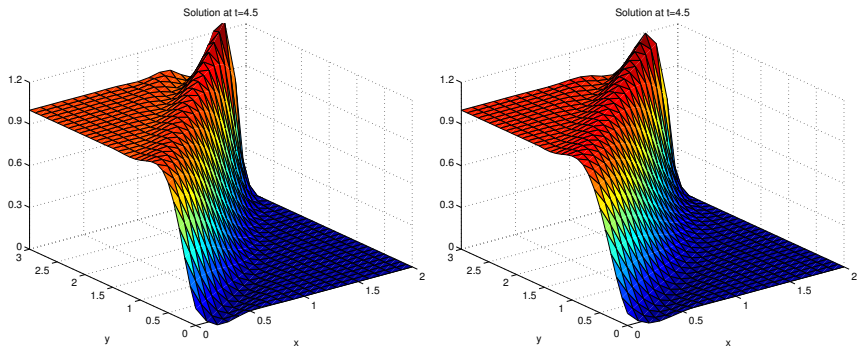
oscilations obtained for quadratic elements are smaller since we are including the diffusion term in the residual and the stabilization provided by SUPG formulation becomes consistent.



(a) 109 time steps ($C = 1$) (b) 27 time steps ($C = 4$)

(c) 7 time steps ($C = 15.45$)

Figure 9: Temporal evolution at lower left's corner for $\nu = 0.3$, $\sigma = 0$ using CN and Galerkin's method.



(a) Linear elements (b) Quadratic elements

Figure 10: Comparison of linear and quadratic elements for $Pe = 1.8$ and SUPG formulation.

Exercise 3

We aim to analyze the Stokes and Navier-Stokes equations in the domain of interest. Use the code available in the Virtual Learning center to compute the finite elements approximation of these problems and answer the questions below. Consider the following BC:

$$\boldsymbol{v = 0\ in\ \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_5}$$

$$\boldsymbol{v_y = -1\ in\ \Gamma_4}$$

Either use one element that based on your own criteria is going to be appropriate, discussing why, or show the results for the different elements presented in the Matlab code.

a)

Use the function Stokes.m to compute the solution of the Stokes problem. The fluid you are considering is a dense fluid, such an oil, select a viscosity that satisfies this assumption.

Describe the weak form and finite element discretization. The mesh size is an important aspect of the quality of the solution. Use your own criterion to explain the mesh you used. Explain why you did in one specific way and the criteria for such a choice. Comment on the results and describe the main properties of the velocity and pressure fields.

In order to work with a very viscous problem, we will use ν such that the Reynolds number is:

$$Re = \frac{v_{ref} L_{ref}}{\nu} << 1$$

Where $v_{ref} = 1$ is the velocity at boundary Γ_4 and $L_{ref} = 3$ is the length of the vertical side of the cavity. A problem in which $Re = 0.01$ can be considered as a very viscous problem:

$$\nu = \frac{v_{ref} L_{ref}}{Re} = \frac{3}{0.01} = 300$$

The pde form of the stationary Stokes problem is:

$$-\nu \boldsymbol{\nabla^2 v} + \boldsymbol{\nabla} p = \boldsymbol{b}$$
$$\boldsymbol{\nabla} v = 0$$

Premultiplicating every equation by a test function (w for velocity and q for pressure) and integrating:

$$-\int_\Omega w\nu \boldsymbol{\nabla^2 v}\, d\Omega + \int_\Omega w \boldsymbol{\nabla} p\, d\Omega = \int_\Omega w \boldsymbol{b}\, d\Omega$$
$$\int_\Omega q \boldsymbol{\nabla} v\, d\Omega = 0$$

Applying the divergence theorem to the viscous and pressure terms:

$$\int_\Omega \boldsymbol{\nabla} w : \nu \boldsymbol{\nabla} \boldsymbol{v} \, d\Omega - \int_\Omega p \boldsymbol{\nabla} \Delta w \, d\Omega = \int_\Omega w \boldsymbol{b} \, d\Omega + \int_{\Gamma^N} \boldsymbol{n} \Delta \boldsymbol{t} \, d\Gamma$$
$$\int_\Omega q \boldsymbol{\nabla} v \, d\Omega = 0$$

The term v can be decomposed into $\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{v_D}$ where $\boldsymbol{v_D}$ are the prescribed values of velocity:

$$\int_\Omega \boldsymbol{\nabla} w : \nu \boldsymbol{\nabla} \boldsymbol{u} \, d\Omega - \int_\Omega p \boldsymbol{\nabla} \Delta w \, d\Omega = \int_\Omega w \boldsymbol{b} \, d\Omega + \int_{\Gamma^N} \boldsymbol{n} \Delta \boldsymbol{t} \, d\Gamma - \int_\Omega \boldsymbol{\nabla} w : \nu \boldsymbol{\nabla} \boldsymbol{v_D} \, d\Omega$$
$$\int_\Omega q \boldsymbol{\nabla} \Delta u \, d\Omega = - \int_\Omega q \boldsymbol{\nabla} \Delta v_D \, d\Omega$$

This system of equations can be expressed as:

$$\begin{bmatrix} \boldsymbol{B} & \boldsymbol{G} \\ \boldsymbol{G} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{h} \end{bmatrix}$$

The problem has been solved using Taylor-Hood Q2/Q1 quadrilaterals, which uses piecewise continuous interpolations, linear for pressure and quadratic for velocity. This element has been chosen because it is stable, satisfies LBB compatibility condition and exhibits optimal quadratic convergence. A uniform mesh with mesh size of 0.1 will be used. A non-uniform mesh with a refinement in the boundaries/vertexes of the domain could improve the solution, but it was discarded due to the complication of its implementation. The mesh size was selected after studying the results obtained for different element sizes seeking convergence. The mesh used is depicted in figure 11, blue for velocity and red for pressure.

The boundary conditions imposed are $v = 0 \; in \; \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_5$, $v_y = -1 \; in \; \Gamma_4$ (the right lower and upper corners belong to Γ) and the pressure is defined up to a constant $p_0 = 0$ at the lower left corner. The results obtained for the velocity and pressure are depicted in figures 12 and 13. The main features that can be observed are the symmetry with respect to the horizontal center line and the pressure singularities at lower and upper right corners, which are characteristic of a leaky cavity problem. The velocity field shows a main vortex which has its center at the middle of the domain, and the pressure field is uniform, except for the singular corners.

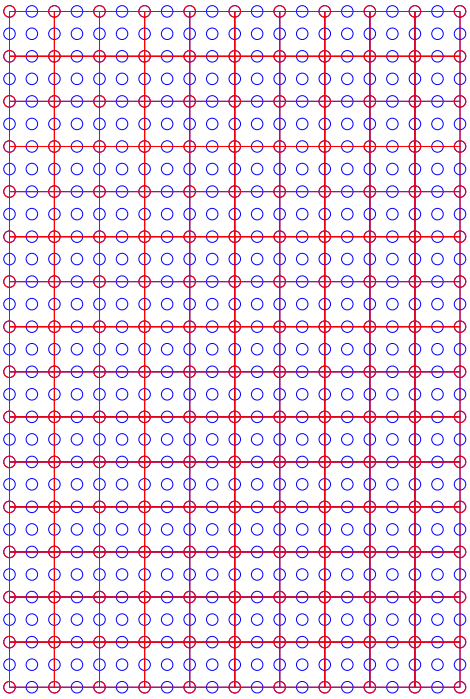
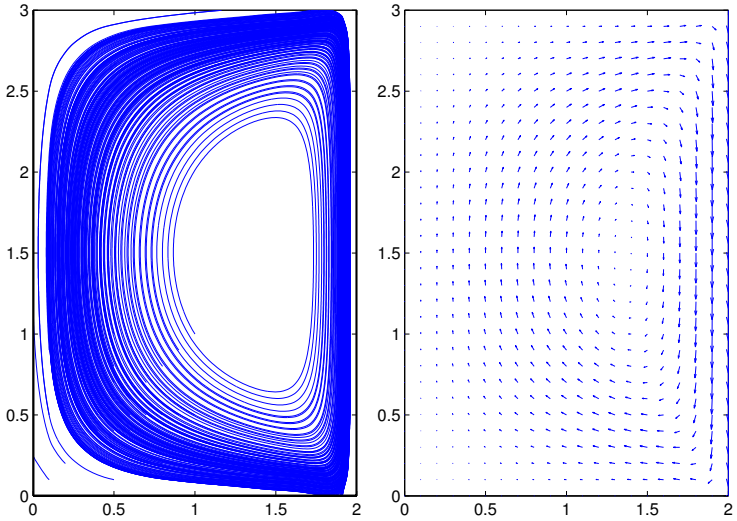


Figure 11: Mesh for Stokes problem.



(a) Streamlines (b) Velocity field

Figure 12: Velocity field obtained in Stokes problem.

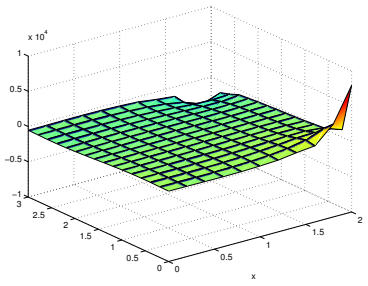


Figure 13: Pressure field obtained in Stokes problem.

b)

Based on your conclusion in the previous point, use the function Navier-Stokes.m to compute the solution of the Navier-Stokes equations using one specific spatial discretization and type of element. Consider a physical problem with Reynolds numbers $Re = 1, 100, 1000, 2000$. Take into account this feature and choose the remaining parameters to fulfil the requirement. Comment on the results.

For every Re studied we must compute ν:

$$\nu = \frac{v_{ref} L_{ref}}{Re} = \frac{3}{\nu}$$

The problem has been solved using a tolerance $tol = 10^{-4}$ and a zero velocity field for the initial solution. The results obtained for $Re = 1, 10, 100$ and 1000 are depicted in figure 15 to 18. The number of iterations needed for the different values of Re is depicted in 14.

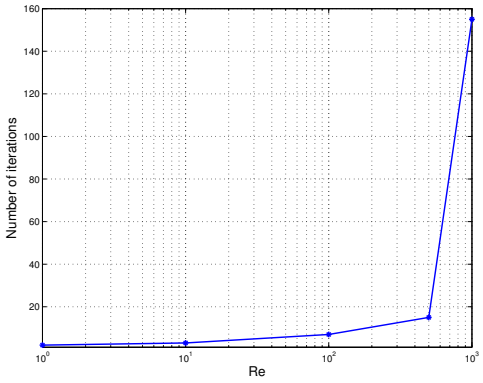


Figure 14: Number of iterations needed for $Re = 1, 10, 100, 500$ and 1000.

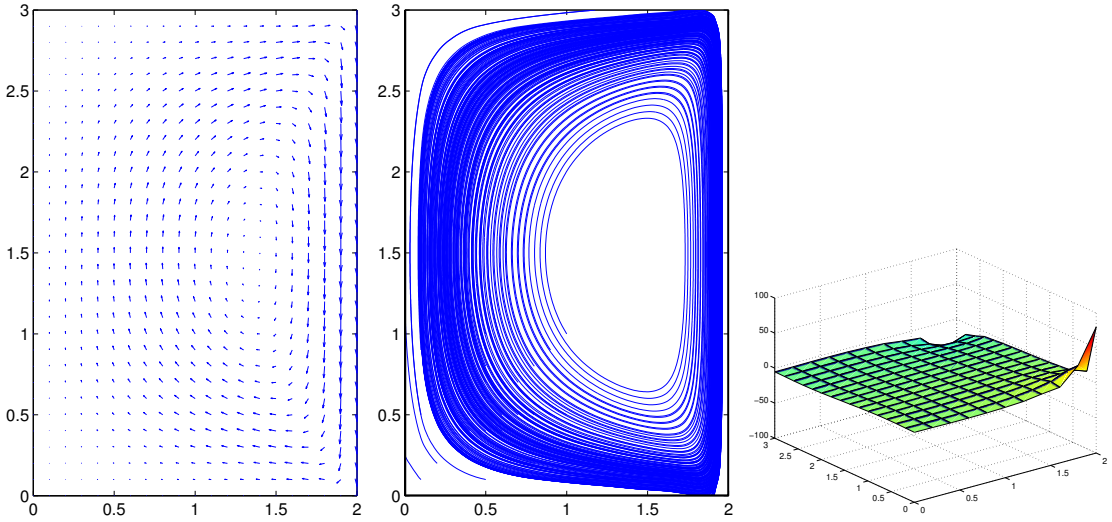
For $Re = 2000$, the problem does not converge using the methods and mesh used. This is due to the fact that convection becomes much more important and the problem begins to move towards turbulence; hence, it is much harder to get an steady solution for the problem and the mesh should be refined in order to capture the small scales of the flow. Other options to improve convergence would be to use a stabilization formulation or use a much better initial solution: For example, we could solve the problem for $Re = 1900$ and then use the solution obtained as the initial guess for $Re = 2000$.

However, since a refinement of the mesh would require much more computational resources and the utilisation of better initial guesses does not ensure convergence, I decided to solve the problem using a transient analysis with the same mesh as before and initial solution the velocity field obtained for $Re = 1000$. The transient analysis has been computed using an explicit scheme for the Chorin-Temam projection method which is a two-step method: we first include viscous and convective terms and then add pressure and incompressibility constraint for velocity. The results obtained for $Re = 2000$ are depicted in 19.

The results obtained for low values of Reynolds numbers (1 and 10) are very close to those obtained for Stokes flow. The shape of the velocity field is almost the same and so

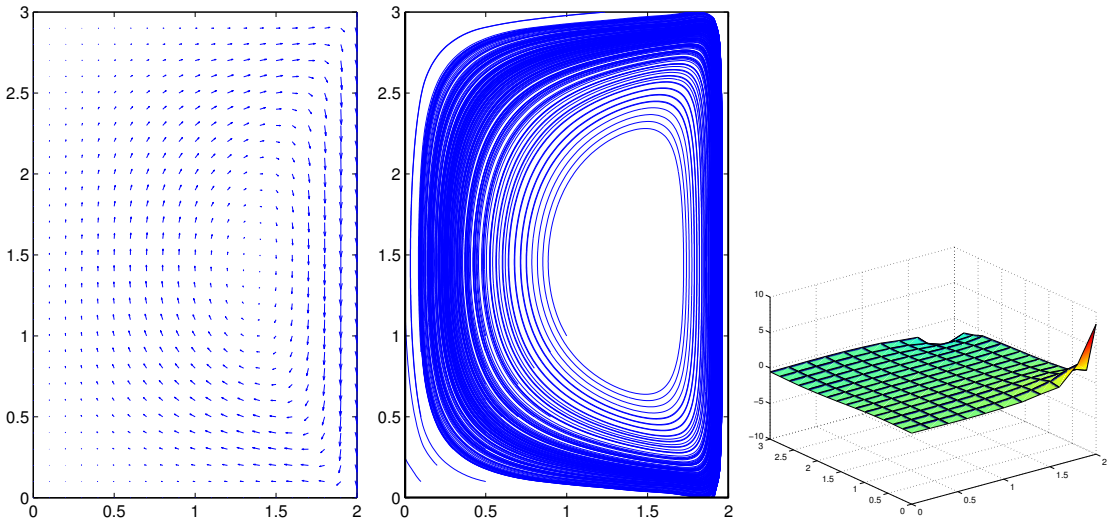
is the shape of the pressure field. However, the magnitude of the pressure decreases as ν or Re increase.

For $Re = 100$, the center of the main vortex tends to move downwards, but the pressure field is similar. For $Re = 1000$, the solution has lost its symmetry and a secondary vortex appears at the upper part of the domain. The pressure field shows some oscillations and it is not as uniform as for lower Re.



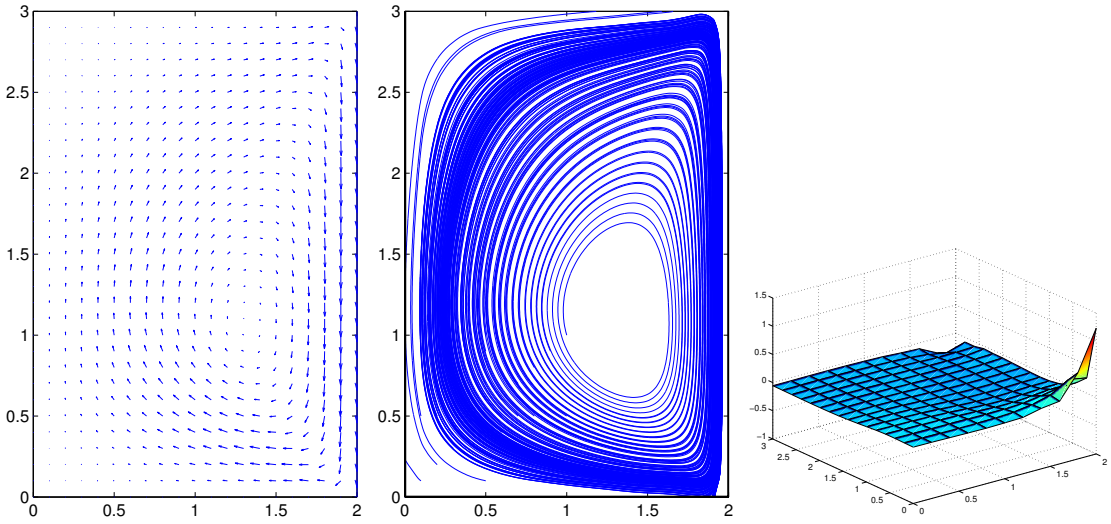
(a) Velocity field (b) Streamlines (c) Pressure

Figure 15: Results for $Re = 1$.



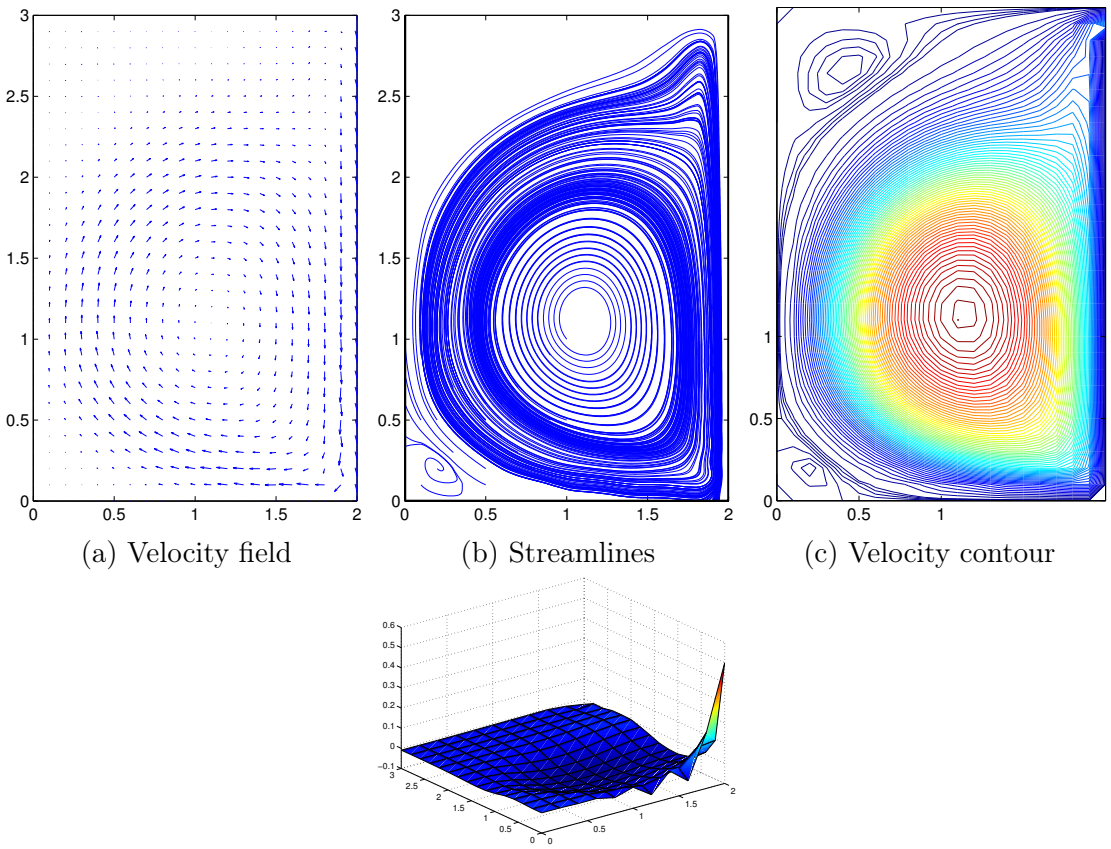
(a) Velocity field (b) Streamlines (c) Pressure

Figure 16: Results for $Re = 10$.



(a) Velocity field (b) Streamlines (c) Pressure

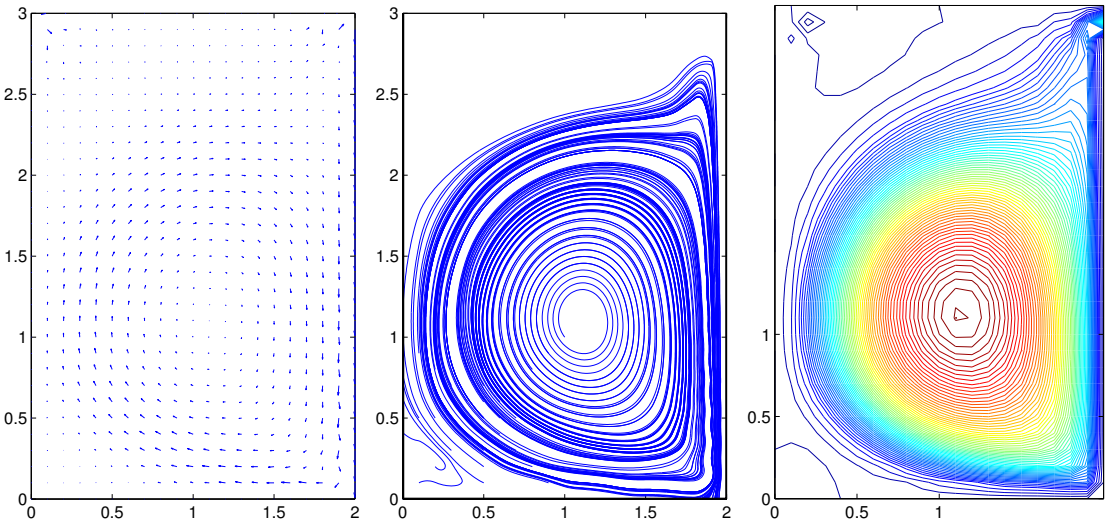
Figure 17: Results for $Re = 100$.



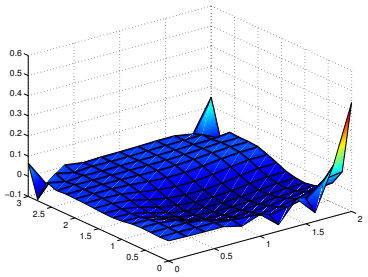
(a) Velocity field (b) Streamlines (c) Velocity contour

(d) Pressure

Figure 18: Results for $Re = 1000$.



(a) Velocity field (b) Streamlines (c) Velocity contour



(d) Pressure

Figure 19: Results for $Re = 2000$.

Appendix: Spatial discretization for Ex 2

Galerkin.m

```
function Sol = Galerkin(X,IEN,Conv,nu,sigma,f,c,Accd1,bccd1,T,s,beta,dt,
    nstep,Element)

% Number of points used in the discretization
npoin = size(X,1);

% COMPUTATION OF THE MATRICES
pospg = Element.GaussPoints;
wpg = Element.GaussWeights;
N = Element.N;
Nxi = Element.Nxi;
Neta = Element.Neta;
N2xi = Element.N2xi;
N2eta = Element.N2eta;

% Matrices obtained by discretizing the Galerkin weak-form
disp(' ')
disp ('Computation of mass matrix M, obtained by discretizing the term (w,u
    )')
M = CreMassMat(X,IEN,pospg,wpg,N,Nxi,Neta);
disp ('Computation of convection matrix C, obtained by discretizing the
    term (w,a grad(u))')
C = CreConvMat(X,IEN,Conv,pospg,wpg,N,Nxi,Neta);
disp ('Computation of stiffness matrix K, obtained by discretizing the term
     (grad(w),grad(u))')
K = CreStiffMat(X,IEN,pospg,wpg,N,Nxi,Neta);

% Integration matrix

[n,m] = size(T);
Id = eye(n,m);

% Computation of the matrix necessary to obtain solution at each time-step:
     A du = F
disp(' ')
disp('Computation of total matrices for the time-integration scheme')
Kt  = C + nu*K + sigma*M;

A = [];
for i = 1:n
    row = [];
    for j = 1:m
        row = [row, Id(i,j)*M  + dt*T(i,j)*Kt];
    end
    A = [A; row];
end

nccd = size(Accd1,1);
Accd = []; bccd = [];
for i = 1:n
    row = [];
    for j = 1:m
        row = [row, Id(i,j)*Accd1];
```

```
    end
    Accd = [Accd; row];
    bccd = [bccd; bccd1];
end

nccd = n*nccd;
Atot = [A Accd'; Accd zeros(nccd)];
Atot = sparse(Atot);

% Factorization of matrix Atot
disp(' ')
disp('Factorization of the matrices for the time-integration scheme')
[L,U] = lu(Atot);
L = sparse(L);
U = sparse(U);

% Initial condition
Sol = c;
% Loop to compute the transient solution

disp(' ')
disp('Transient analysis: computation of the solution at each time step')
for i=1:nstep
    aux = dt*(-Kt*c);
    F = [];
    for i =1:n
        F = [F; s(i)*aux];
    end
    F = [F;bccd*0];
    dc  = U\(L\F);
    dc = reshape(dc(1:n*npoin),npoin,n);
    c = c + sum(dc,2);
    Sol = [Sol c];
end
```

SUPG.m

```
function Sol = SUPG(X,IEN,Conv,nu,sigma,f,c,Accd1,bccd1,T,s,beta,dt,nstep,
    tau,Element)

% Number of points used in the discretization
npoin = size(X,1);

% COMPUTATION OF THE MATRICES
pospg = Element.GaussPoints;
wpg = Element.GaussWeights;
N = Element.N;
Nxi = Element.Nxi;
Neta = Element.Neta;
N2xi = Element.N2xi;
N2eta = Element.N2eta;
h = Element.meshsize;

% Matrices obtained by discretizing the Galerkin weak-form
disp(' ')
disp ('Computation of mass matrix M, obtained by discretizing the term (w,u
    )')
```

```
M = CreMassMat(X,IEN,pospg,wpg,N,Nxi,Neta);
disp('Computation of convection matrix C, obtained by discretizing the term (w,agrad(u))')
C = CreConvMat(X,IEN,Conv,pospg,wpg,N,Nxi,Neta);
disp('Computation of stiffness matrix K, obtained by discretizing the term (grad(w),grad(u))')
K = CreStiffMat(X,IEN,pospg,wpg,N,Nxi,Neta);
disp('Computation of matrix K2, obtained by discretizing the term (agrad(w),agrad(u)+sigma-nu gradgrad(u))')
K2 = CreK2Mat(X,IEN,Conv,pospg,wpg,N,Nxi,Neta,N2xi,N2eta,nu,sigma,h);


% Integration matrix
[n,m] = size(T);
Id = eye(n,m);
tauT = tau*T;
tauT_T = tauT'*T;
tauT_s = tauT'*s;


if size(tau) == [1,1]
    tau = tau*ones(size(T));
end

% Computation of the matrix necessary to obtain solution at each time-step: A du = F
disp(' ')
disp('Computation of total matrices for the time-integration scheme')
Kt  = C + nu*K + sigma*M;

A = [];
for i = 1:n
    row = [];
    for j = 1:m
        row = [ row, Id(i,j)*M  + dt*T(i,j)*Kt + ...
                        tauT(j,i)*C' + dt*tauT_T(i,j)*K2 ];
         end
     A = [A; row];
end

nccd = size(Accd1,1);
cero = zeros(nccd,npoin);
Accd = []; bccd = [];
for i = 1:n
    row = [];
    for j = 1:m
        row = [row, Id(i,j)*Accd1];
    end
    Accd = [Accd; row];
    bccd = [bccd; bccd1];
end

nccd = n*nccd;
Atot = [A Accd'; Accd zeros(nccd)];
Atot = sparse(Atot);

% Factorization of matrix Atot
disp(' ')
```

```
disp('Factorization of the matrices for the time-integration scheme')
[L,U] = lu(Atot);
L = sparse(L);
U = sparse(U);

% Initial condition
Sol = c;
% Loop to compute the transient solution
disp(' ')
disp('Transient analysis: computation of the solution at each time step')
for i = 1:nstep
    aux1 = -dt*Kt*c;
    aux2 = -dt*K2*c;
    F = [];
    for i =1:n
        F = [F; s(i)*aux1 + tauT_s(i)*aux2];
    end
    F = [F;bccd*0];
    dc  = U\(L\F);
    dc = reshape(dc(1:n*npoin),npoin,n);
    c = c + sum(dc,2);
    Sol = [Sol c];
end
```