Master of Science in Computational Mechanics 2016

Finite Element Methods In Fluids

Mohanakrishnan guruguhanathan
Assignment
Universitat Politècnica de Catalunya – BarcelonaTech ,
Jordi Girona 1, E-08034 Barcelona, Spain

May 23, 2016

Exercise 1

a) Neglecting the transient term, 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 Galerkins method.and an approximation of the solution

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

solution:- The weak form of the convection-diffusion-reaction problem can be derived as the following,

$$u^h(x) = \sum_j u_j N_j(x) \tag{1}$$

is same as

$$\int W u^{"} dx = \int W f dx \tag{2}$$

integration by parts

$$\int W u dx - [Wu] = \int W f dx \tag{3}$$

rewriting the equation

$$u^h = \sum_j^{n-1} N_j u_j + 0N_0 + \alpha N_n \int \frac{dW_j}{dx} \sum_j^n \frac{dN_i}{dx} u_i dx = N_n u|_{x=1} + N_0 u|x = 0 + \int W_i f_i dx \tag{4}$$

by weighted residuals method,

$$W_i = N_i \sum \int_{l^e} \frac{dN_i}{dx} \sum_{j=0} n \frac{dN_i}{dx} u_j dx = N_n u|_{x=1} + N_0 u|_{x=0} + \sum_i^n \int N_i f_i dx \tag{5}$$

take

$$q_0 = N_0 u|_{x=0} \tag{6}$$

and

$$q_1 = -N_n u|_{x=1} q_0 + q_1 = q \tag{7}$$

$$\sum_0^{n^{elem}} \frac{dN_i}{dx} \sum_{j=0}^n \frac{dN_i}{dx} u_j dx = q + \sum_0^{n^{elem}} N_i f_i dx \tag{8}$$

Thus here we can see the general convection-diffusion-reaction is

$$\sum_0^{n^{elem}} \frac{dN_i}{dx} \sum_{j=0}^n \frac{dN_i}{dx} u_j dx = \sum_0^{n^{elem}} N_i f_i dx + \sigma N_j - q$$

b) Use the Matlab code that solves a 2D convection-diffusion equation during this course which was downloaded from the Virtual Learning Center. Modify this code, if you didnt before, to solve this problem with linear elements, a spatial discretization h=0.2 and, convective velocity, diffusion parameter, reaction and source term as:

- $a = 1, \nu = 10^{-3}, \sigma = 10^{-3}, s = 0$
- $a = 10^{-3}, \nu = 10^{-3}, \sigma = 1, s = 0$
- $a = 1, \nu = 10^{-3}, \sigma = 10^{-3}, s = 1$
- $a = 1, \nu = 10^{-3}, \sigma = 1, s = 0$

The BC are

$$\rho = \mathbf{1} \textbf{ in } \Gamma_2$$
$$\rho = \mathbf{0} \textbf{ in } \Gamma_4$$

solution:- The aim of this section of the exercise is to understand the numerical methods used for solving the steady state convection-diffusion equations. Since the Galerkin methods inherently have a problem of lacking diffusion and results in oscillations, so we use techniques which utilize to stabilize the oscillations in the solutions. Some of the techniques we use are Streamline Upwind Petrov-Galerkin (SUPG), Galerkin, Galerkin Least-Squares(GLS).

In the program the boundary is taken as the given as $\rho = 1$ & 0 in Γ_2 & Γ_4.we also add σ & source term to the program. In this exercise we vary as given in the cases a, ν , σ & s, to see their effect on the solution. in the program i have created a cases selection called case.m to run the program and the direction of the flow is taken y= 0

Since here we want to see initial conditions for galerkin method lets see the working of the method for all 4 cases. now we can see all the 4 graph with cases.

If we see that in the above figure the boundary is applied well and the we can see that the solution in

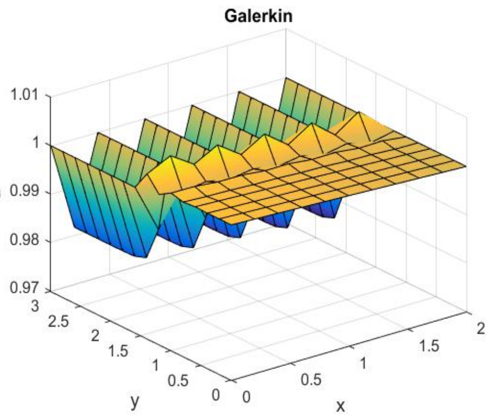


Figure 1: a = 1, $\nu = 10^{-3}$, $\sigma = 10^{-3}$, s = 0

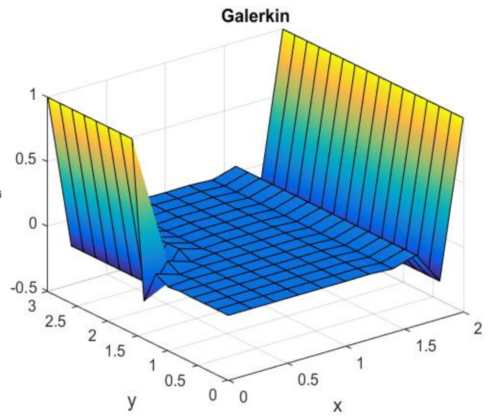


Figure 2: $a = 10^{-3}$,$\nu = 10^{-3}$,$\sigma = 1$,s = 0

using the method has given a very bad result so we have to make this better using the other few stabilization methods like SUPG and GLS IN THE next section of the assignment. But an important point here to be noted is that change due to the diffusion and reaction term is very less so the adding the term or not is not gone to change the solution of the problem.

Here to improve the solution we can by increasing the time step and artificial diffusion term Note:- Diffusion and Reaction values are negligible or close to zero. Codes have been add to the appendix Exercise 1 of the Report.

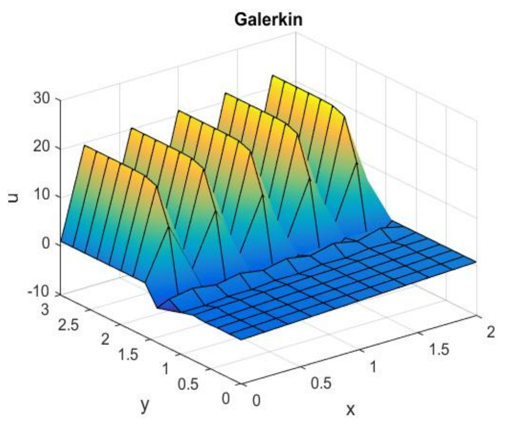


Figure 3: a = 1, $\nu = 10^{-3}$, $\sigma = 10^{-3}$, s = 1

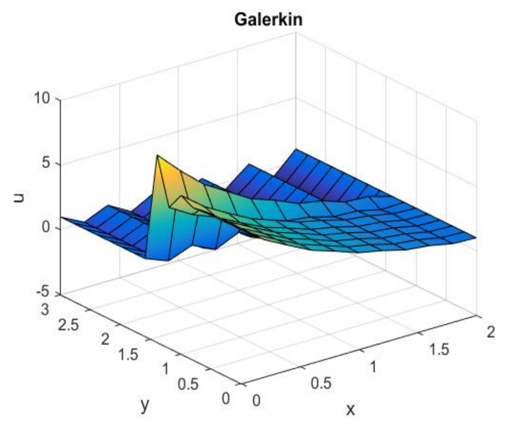


Figure 4: a = 1,$\nu = 10^{-3}$,$\sigma = 1$,s = 0

c) Choose one of the set of parameters for which using 2 and 3 would have sense. Explain why you choose it. Write the modifications down in a clear mathematical fashion as well as how each method is obtained step by step. Explain the modifications you have introduced in the Matlab source code for the two cases and compare the results. Comment on the obtained results.

solution:- The weak foam of the general form of the convection-diffusion-reaction term,

$$\int_\Omega w(a.\nabla u)d\Omega - \int_\Omega \nabla w.(\nu\nabla u)d\Omega = \int_\Omega wsd\Omega + \int_{\Gamma_N} whd\Gamma + \int_\Omega \sigma wd\Omega \tag{9}$$

for all w $\in \nu$
Now we can use the following addition term to the general we can write this equation in the following way,

$$\mathbf{a}(w,u) = \int_\Omega \nabla w.(\nu\nabla u)d\Omega, \qquad (w,s) = \int_\Omega wsd\Omega, \qquad C(\mathbf{a};w,u) = \int_\Omega w(\mathbf{a.}\ \nabla u)d\Omega, \qquad (w,h)_{\Gamma_N} whd\Gamma. \qquad (w,\sigma \tag{10}$$

$$a(w,u) + C(\mathbf{a};w,u) + (w,\sigma,u) + \sum_e \int_{\Omega^e} \mathcal{P}(w)\tau\mathcal{R}(u)d\Omega = (w,s) + (w,h)_{\Gamma_N} + \sum_e \int_{\Omega_e} [\mathbf{a}.\nabla w^h - \nabla(\nu\nabla w^h) + \sigma w^h] \tag{11}$$

In the equation number 11, $\sum_e \int_{\Omega^e} \mathcal{P}(w)\tau\mathcal{R}(u)d\Omega$. This term is called the stabilization term.

SUPG Method

According to the SUPG method $\mathcal{P}(w) = \mathbf{a}.\ \nabla$ w,

$$[h]\mathbf{a}(w^h,u^h) + \mathbf{C}(\mathbf{a};w^h,u^h) + (w^h,\sigma u^h) + \sum_e \int_{\Omega^e} (\mathbf{a}.\nabla w^h)\tau[a.\nabla u^h - \nabla.(\nu\nabla u^h) + \sigma u^h - s]d\Omega \tag{12}$$

$$= (w^h,s) + (w^h,h)_{\Gamma_N} + \sum_e \int_{\Omega_e} [\mathbf{a}.\nabla w^h - \nabla(\nu\nabla w^h) + \sigma w^h]$$

so according this method we can see that the solution obtained is much better. But the solution is thus obtain is good for now. So the program is thus run after which we got the following results, Here we take

The reason we take case 3 is because at zero σ condition the method acts like the GLS.So it easy for us to verify the program for both the SUPG & GLS. But if take the result for the case 4 for the same program the following result can be obtain, so now you can see in the result that the boundary of the problem is applied well and we can also see that the convective velocity (a) = 1 and the diffusion parameter = 10^3 and reaction of the problem is 1. So,we can see that zreo at all the boundary and $\Gamma_2 = 1$. this show the correctness of the results.

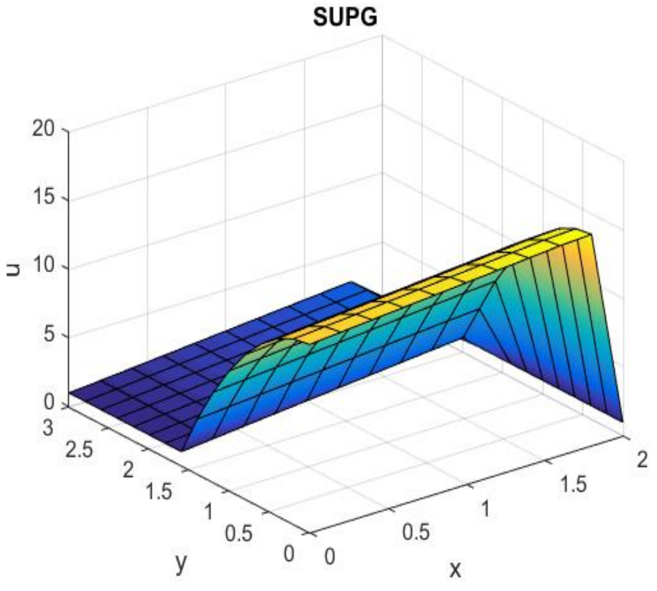


Figure 5: a = 1,$\nu = 10^{-3}$,$\sigma = 0$,s = 1

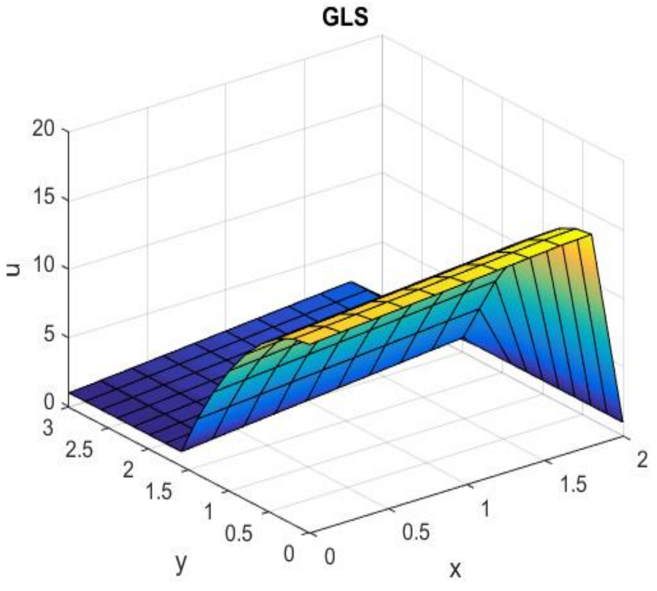


Figure 6: a = 1,$\nu = 10^{-3}$,$\sigma = 0$,s = 1

GLS Method

In this section we can see that the mathematical steps to achieve GLS method, So now we go back to equation number 11 $\sum_e \int_{\Omega^e} \mathcal{P}(w)\tau\mathcal{R}(u)d\Omega$.
Now we can see how stabilization terms can be used to GLS method from the general Galerkin method, $\mathcal{P}(w) = \mathcal{L}(w) = \mathbf{a}.\nabla w - \nabla.(\nu\nabla w) + \sigma w$. So if we replace the the following term in equation 11 we get the mathematical equation of Galerkin Least-squares Method.

$$[h]\mathbf{a}(w^h, u^h) + \mathbf{C}(\mathbf{a}; w^h, u^h) + (w^h, \sigma u^h) + \sum_e \int_{\Omega^e} (\mathbf{a}.\nabla w - \nabla.(\nu\nabla w) + \sigma w - s]d\Omega \tag{13}$$

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

This can be rewriten in the following way

$$[h]\mathbf{a}(w^h, u^h) + \mathbf{C}(\mathbf{a}; w^h, u^h) + (w^h, \sigma u^h) + \sum_e \int_{\Omega^e} (\mathcal{L}(w^h)\tau[\mathcal{L}(u^h) - s]d\Omega \tag{14}$$

$$= (w^h, s) + (w^h, h)_{\Gamma_N} + \sum_e \int_{\Omega_e} [\mathbf{a}.\nabla w^h - \nabla(\nu\nabla w^h) + \sigma w^h]$$

So now we can show that the result goes according to the theoretical values. So we can see that the method is correct and as said before we can both SUPG and GLS works same way when the value of the reaction term is 0.the modification in the codes have been show in the appendix.

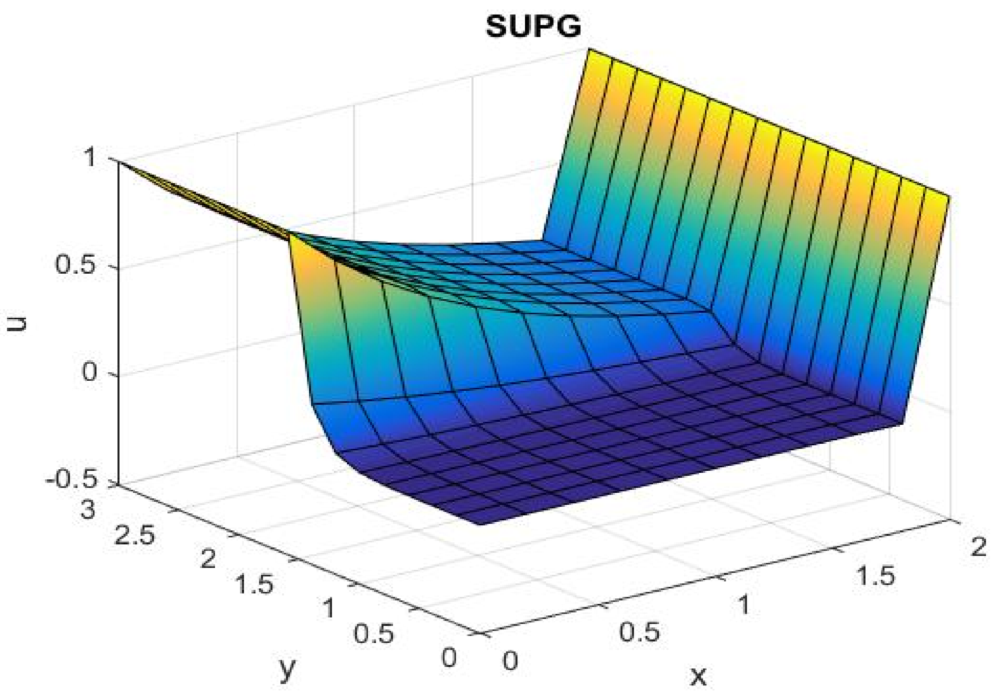


Figure 7: a = 1,$\nu = 10^{-3}$,$\sigma = 1$,s = 0

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.

solution:- To solve this problem we have to use the following method to find the solution we use the following steps initially we find flux for which we impose the dirichlet boundary conditions $\rho = 2$ in Γ_2 and $\rho = 1$ in Γ_4. so according to the FEM system $ku = f + q$.
Using the above we find the values of the flux on a point. once we find that we can find the normal of the this point and trace back the it in the Neumann bc. but before this step we should impose the the values of this flux in Γ_4. if we see in this process we can get the same result. whih is show below. here we can see that the

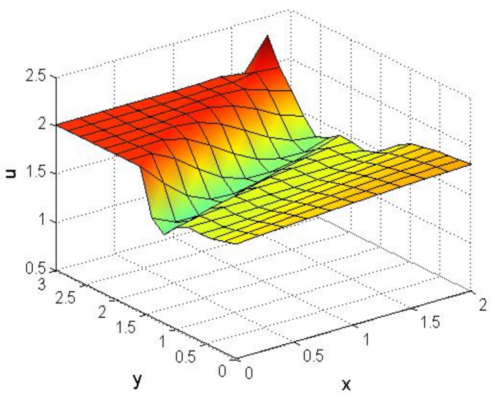


Figure 8: a = 1,$\nu = 10^{-3}$,$\sigma = 1$,s = 0

result are same.

Exercise 2

In this exercise we will include the transient term. We need to impose the initial condition for the unknown variable ρ. This is given by $x_0(x;0) = x(2-x)$ in Ω. The boundary condition will be given by with homogeneous Dirichlet boundary conditions $\rho = 1$ in Γ_2 and $\rho = 0$ in Γ_4 The vector $a(x,y) = (-x;-y)$.

a) & b) Discretize the problem above using the formulation for space that you consider more appropriate among the ones described in Exercise 1. For the time discretization use: 1. Crank-Nicolson method 2. Two step third order TG method.

solution:-In this problem we are going to solve the pure convection to a convection diffusion equation using crank nicolson method for time and galerkin formulation for space. for this we have to identify the convection matrix method using the following formula here we also used diffusion-transient term.

$$C_{ab} = \int_{\Omega} N_a(a.\nabla N_b)d\Omega$$

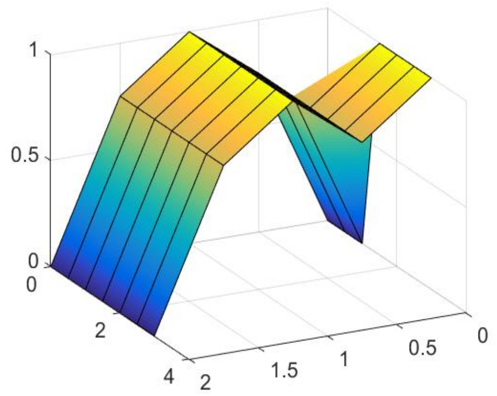
So now considering nu as 10^{-3} and since there is no reaction as such we implement the code and then now we set the given problem.

In this section we have to use the model $x_0 = x(2-x)$ has to be evaluated by Crank-Nicolson method & 2-Step 3^{rd} TG method. so lets see the mathematical equation for the two methods first,

Space discretization

Here we take Galerkin method for the comfortability of programming in the space. so that the method go side by side. now lets see the mathematical derivation for crank-Nicolson method followingly TG3.

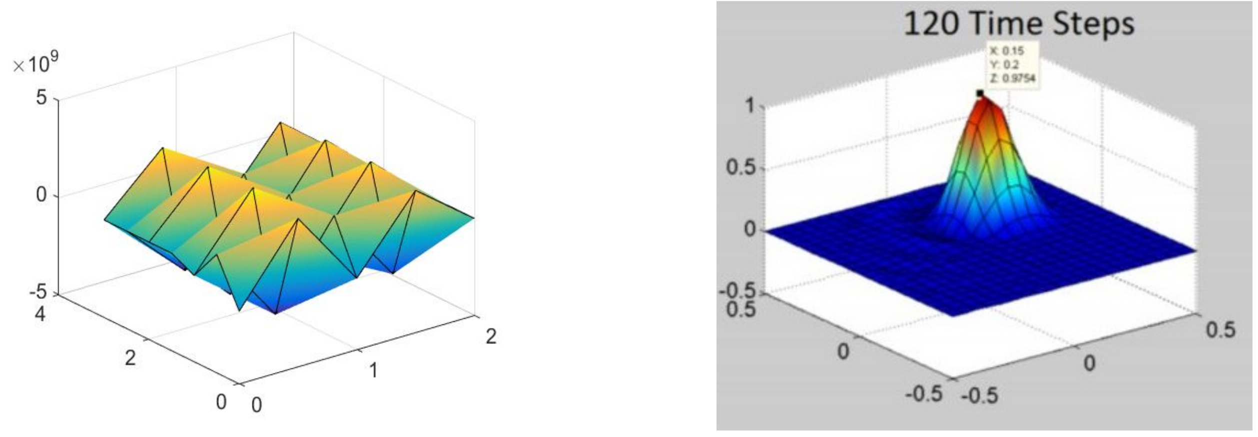
Crank-Nicolson



$$case3Crank-Nicolson+Galerkin$$
$$A = M - 1/2 * dt * C + 1/2 * dt * Mout;$$
$$B = dt * C - dt * Mout;$$

this is the step we include it the program and compute for the given above is the mathematical expression in terms of matlab code. here the stabilization term is the same galerkin method. if see the solution

Figure 9: CRANK-NICOLSON for time steps 120



When we see the results with the traveling cone problem and our problem. This problem can be used for the verification of the result from traveling cone to that of our results are shown in the above graph. we can see that the results are good. so further stabilization techniques can be used in the problem for much refine and better results. for understanding the working. Note:- The modified program can be seen in the appendix section.

TG3 2-STEP

the mathematical equation for TG3-2step method can be written in the following way,

$$\tilde{u}^n = u^n + \frac{1}{3}\Delta t u_t^n + \alpha \Delta t^2 u_{tt}^n, \tag{15}$$

$$u^{n+1} = u^n + \Delta t u_t^n + \frac{1}{2}t^2 \tilde{u}_{tt}^n,$$

$$(1+\frac{1}{6}\delta^2)(\tilde{u}^n - u^n) = \frac{1}{6}C\delta u^n + \alpha c^2\delta^2 u^n, \tag{16}$$

$$(1+\frac{1}{6}\delta^2)(u^{n+1} - u^n) = -\frac{1}{2}C\delta u^n + \frac{1}{2}C^2\delta^2\tilde{u}^n,$$

this can be written in other words this scheme can be re-written in the following,

$$(N_i, N_j)(\tilde{u}^n - u^n) = \frac{1}{3}\Delta t(N_i, \mathbf{a.}\nabla N_j)u^n + \alpha\Delta t^2(\mathbf{a.}\ \nabla N_i, \mathbf{a.}\ \nabla N_j)u^n \tag{17}$$

$$(N_i, N_j)(u^{n+1} - u^n) = \Delta t(a.\nabla N_i, N_j)u^n + \frac{1}{2}\Delta t^2(a.\nabla N_i.a.N_j)u^n$$

the following algorithem has been add to the program for its functions,

$$case4TG3$$
$$alpha = 1/9;$$
$$B = cell(3,1);$$
$$A = M;$$
$$B1 = dt/3 * (C - Mout) + alpha * dt^2 * (-K + Cout);$$
$$B2 = dt * (C' - Mout);$$
$$B3 = 0.5 * dt^2 * (-K + Cout);$$

now we can see the working of the TG3 2-step in our problem and its results, Here we

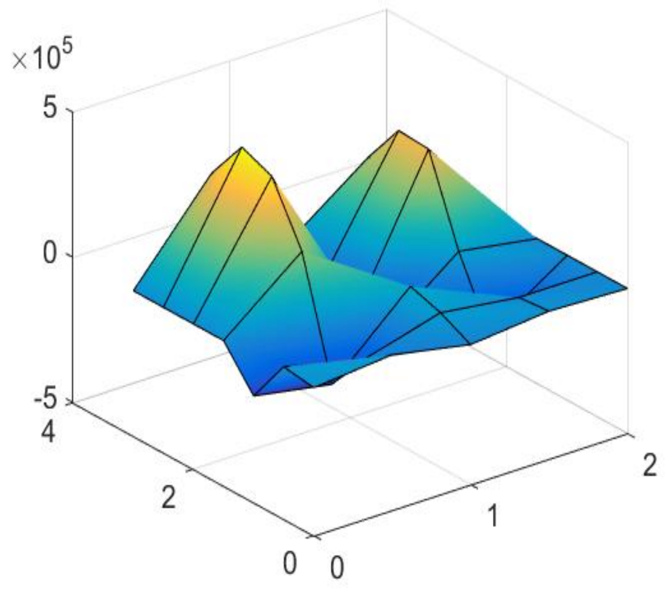


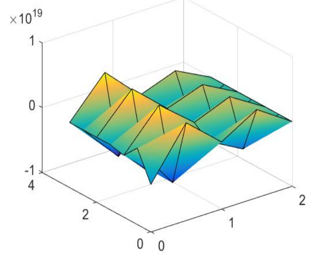
Figure 10: TG3 2-step

can see that the dirichlet bc as 0 in all the bc except in Γ_2 as 1. so we can see that BC is well applied and the vector a(x,y)=(-x,-y) so which is seen in the graph figure 10. this methods complete codes can be seen in the appendix under ex 2.

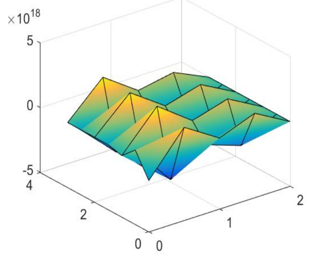
c) Compute the solutions in the time frame t ∈ (0; t^n). You should choose t^n with your on criterion, making sure that the entire transient model is analyzed. For the spatial discretization,using linear elements, do a sensitivity analysis of the mesh dependency and comment on the performance. A initial 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.

solution:- As we know that time change can give us difference in result and computational cost of the solution can be determined accordingly, but lets see if the there is any change in the solution due to time frame. the proposed time frame was 110. And I have taken 60 and 150 as the time lower time frame and higher time frame. so lets discuss the result accordingly.

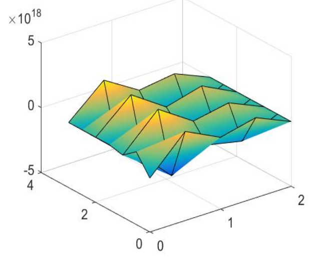
Crank-Nicolson method with time frame



60

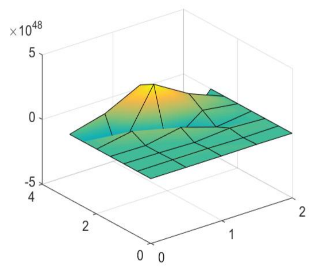


110

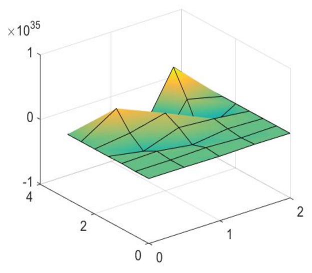


150

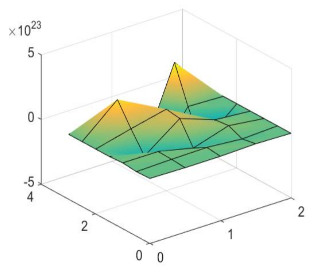
TG3 2-Step method with time frame



60



110



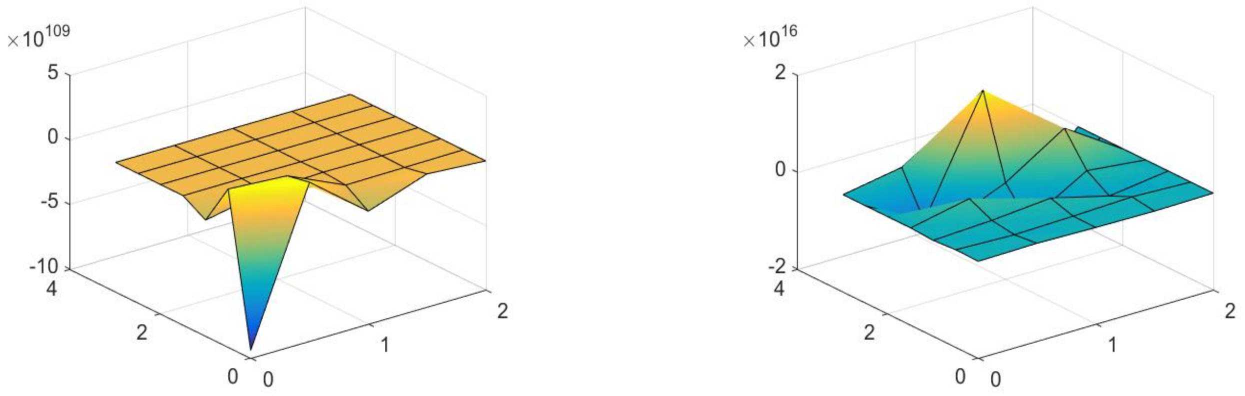
150

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

solution:-if we see the solution after the change of the element are shown below
Here we see the change in the element according to shape function and elemental size.
Here we can see the solution is become much better then other means that we can see that solution is getting much refined when we use linear element. The quadratic element solution is better and accurate in many times. if we refine the mesh size the solution is much better. note:- all the result above are from quadratic element.

Figure 19: quadratic element and linear element

Figure 20: quadratic element and linear element



Exercise 3

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. Choose the viscosity in this direction. Describe the weak form and fintie 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 choice.Comment on the results. Describe the main properties of the velocity and pressure fields.

solution:- Applying the boundary condition,

$$v = 0 \qquad in \qquad \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_5$$
$$v_y \qquad in \qquad \Gamma_4.$$

and since velocity of the problem is now set in the Y-direction and the mesh uniform. the plot show the mesh
This problem I use the Q2Q1 element as the other element in model doesn't satisfy the LLB condition. this element is made of continuos biquadratic velocity,continuos pressure elements in the mesh we seen the blue element are the velocity elements and the red elements are the pressure element as the element is interpolation is done piece-wise and continuous. this help the element to satisfy the LLB condition, $dim\mathcal{Q}^h \leqslant dim\nu^h$
Here the velocity is located in the nodes & midpoint of the boundary and center, which the sum upto 9 nodes. The pressure is located in the nodes edges and sums upto 4. which we can see from the velocity and pressure graphs.
Here you can see that the velocity field is moving more in the y-direction and pressure is tending from -5 to 5. here if the mesh is refined then the solution much prefect and nicer.

Figure 21: Uniform Meshing

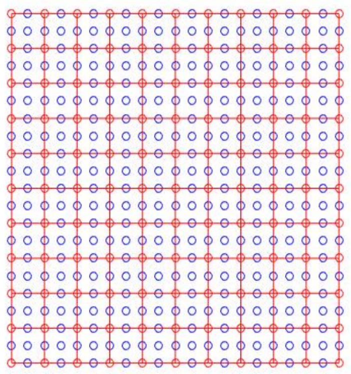
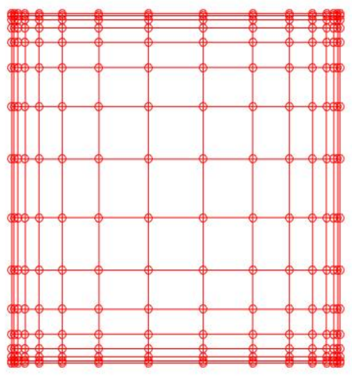


Figure 22: Uniform Meshing



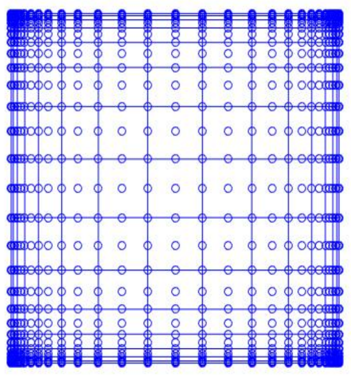
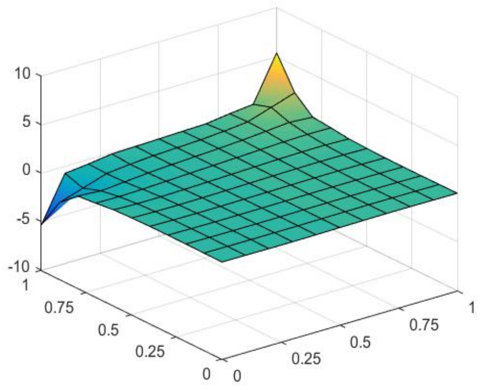
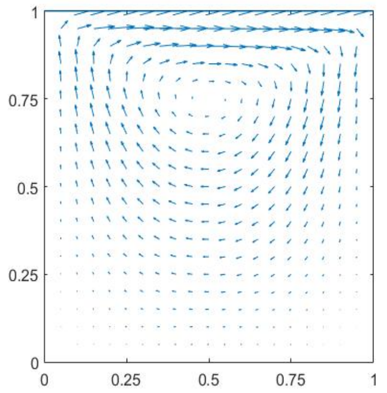


Figure 23: Uniform Meshing





b) Based on your conclusion in the previous point, use the function NavierStokes.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. In particular, discuss the number of iterations needed to achieve convergence, the evolution of the pressure field, the position and the strength of the velocity.

solution:- As this part of Exercise we have to change Re and see the working of the velocity and pressure in the model using uniform mesh and difference of mesh size.
The following graphs can so the working so the pressure and velocity in trems of the mesh and number iterations for convergence.

Here as we see if the mesh is size is changed then results better which we can see from the following graphs,

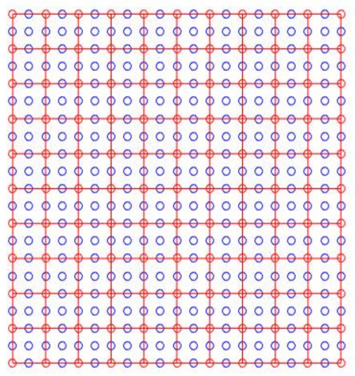
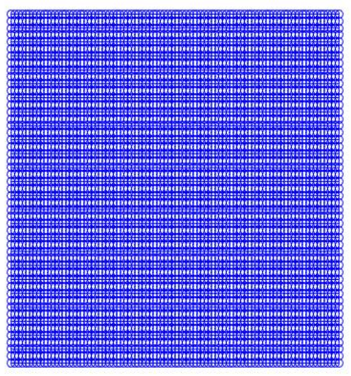



Figure 24: Uniform Meshing

but before that lets see the working of the function of the streamline in the model with element Q2Q1, As we

Figure 25: Streamlines

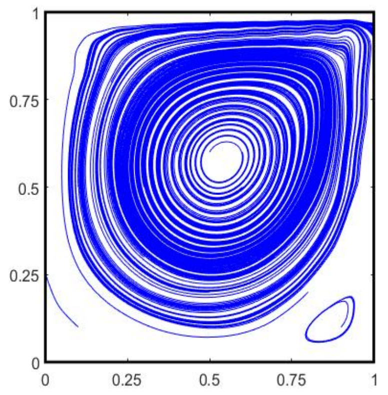


Figure 26: Re=1

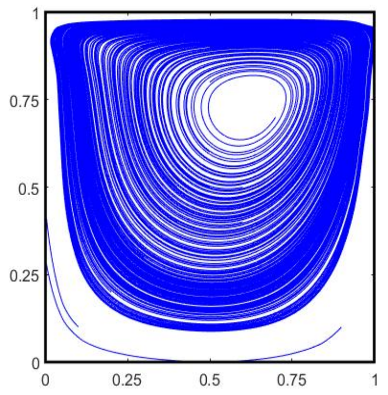


Figure 27: Re=100

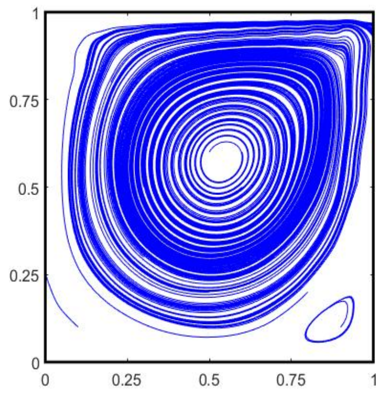


Figure 28: Re=1000

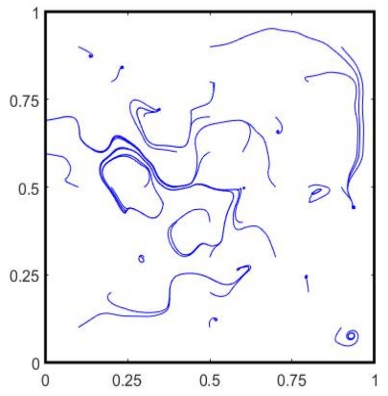


Figure 29: Re=2000

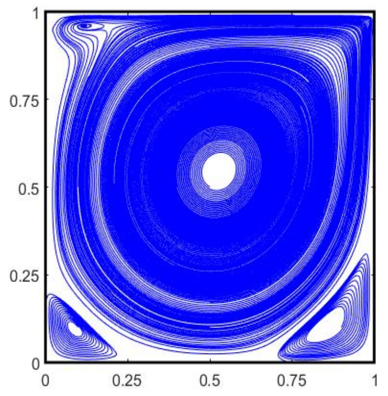


Figure 30: Re=2000 with Denser mesh

increases the Re the streamline, velocity field and pressure field get more dissolute and nosier this is because the LLB condition of the model gets filed slow with increased of the Reynolds number.
If we take care of the convergence the problem are in the table:-

Figure 31: pressure field

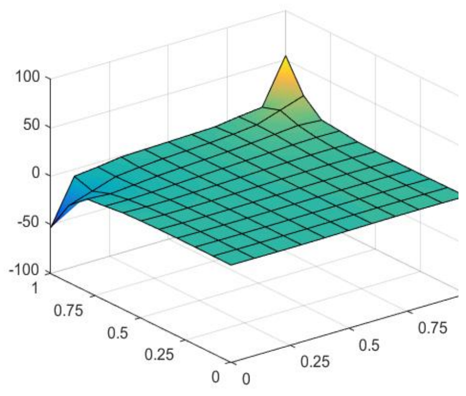


Figure 32: Re=1

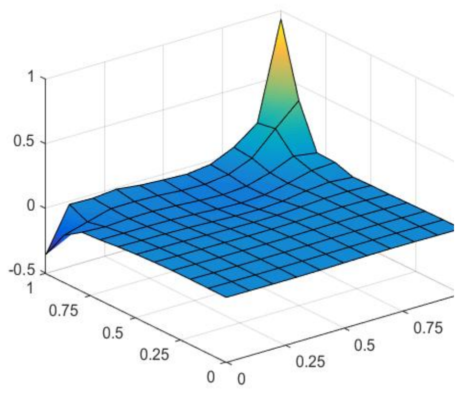


Figure 33: Re=100

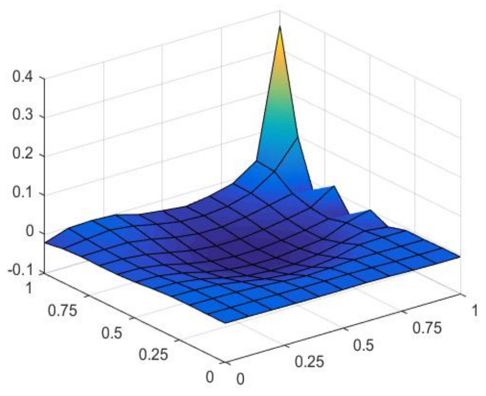


Figure 34: Re=1000

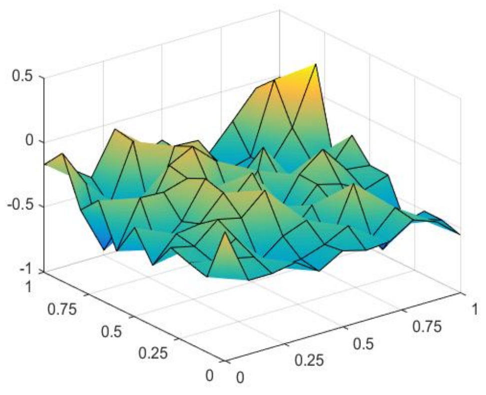


Figure 35: Re=2000

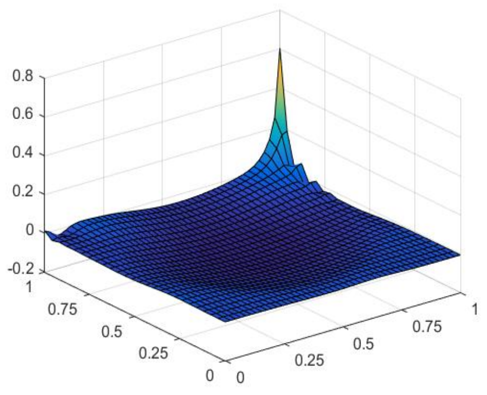


Figure 36: Re=2000 with Denser mesh

Reynolds Number	number iterations for convergence.
1	2
100	7
1000	99
2000	99

Figure 37: Velocity field

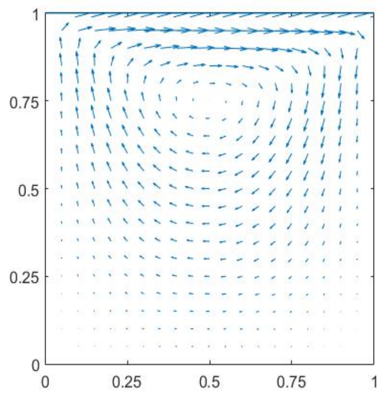


Figure 38: Re=1

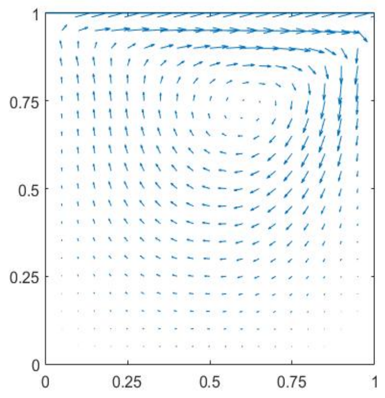


Figure 39: Re=100

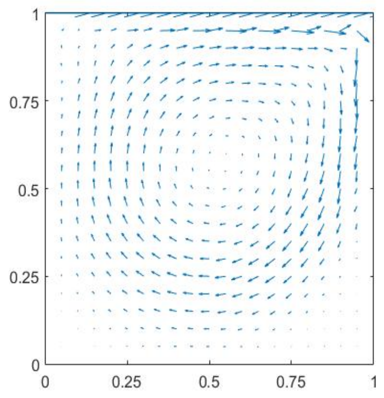


Figure 40: Re=1000

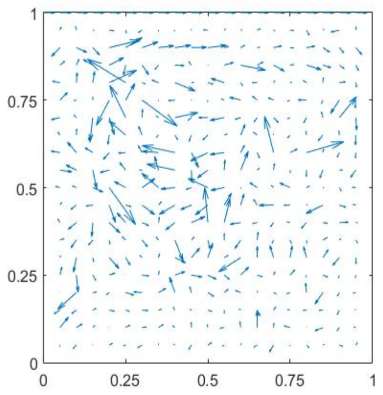


Figure 41: Re=2000

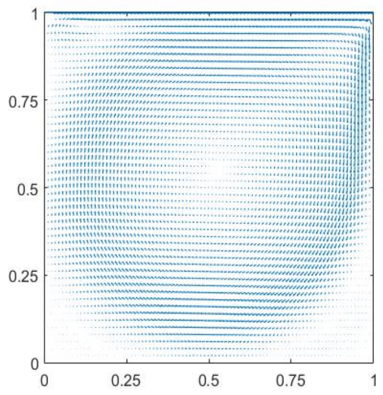
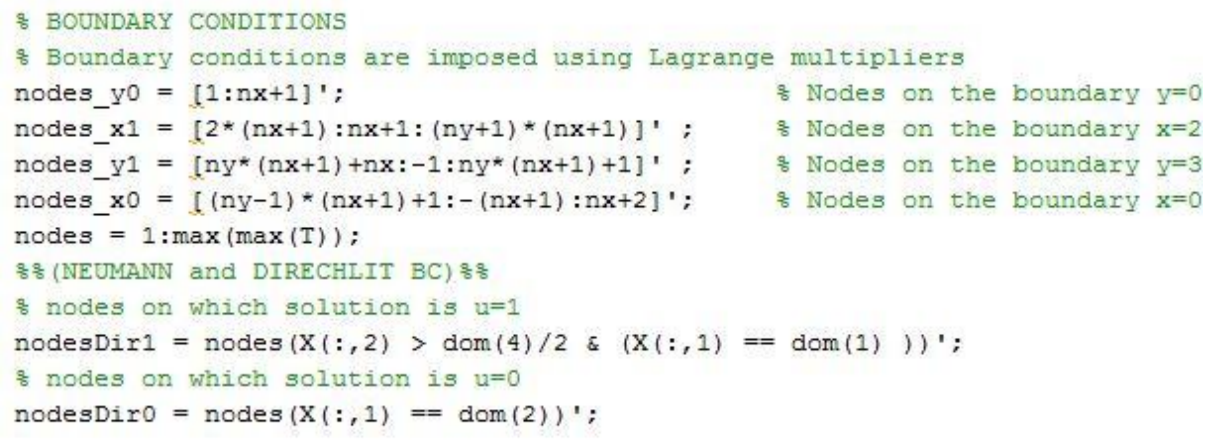


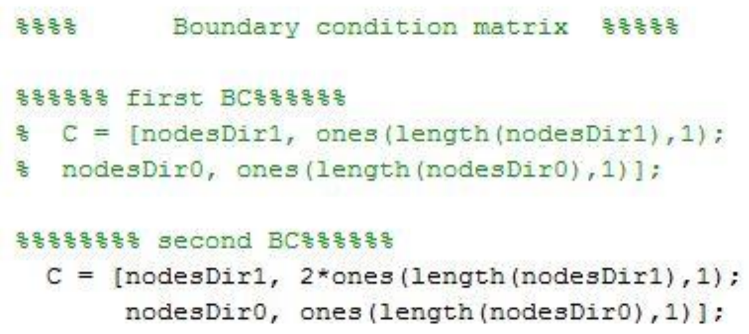
Figure 42: Re=2000 with Denser mesh

APPENDIX

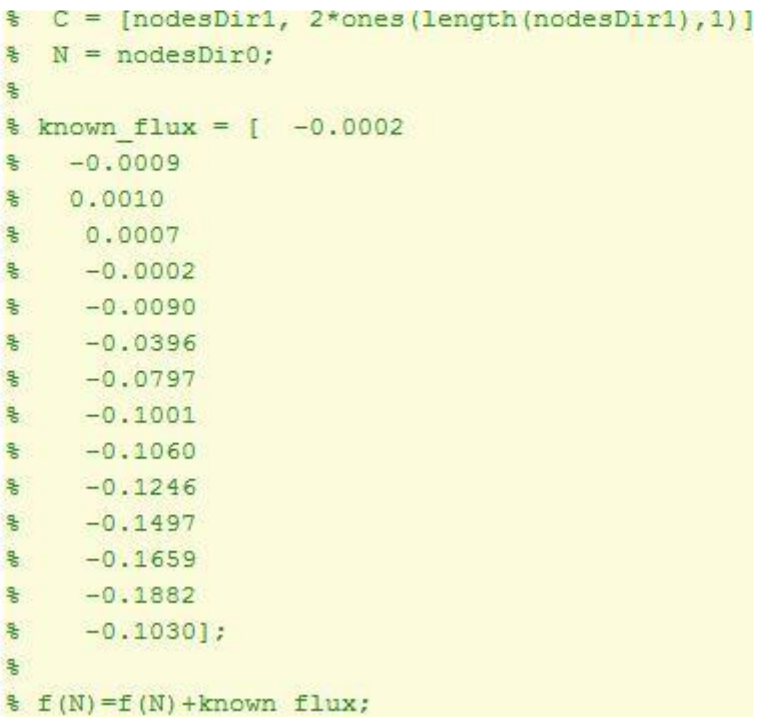
Exercise 1



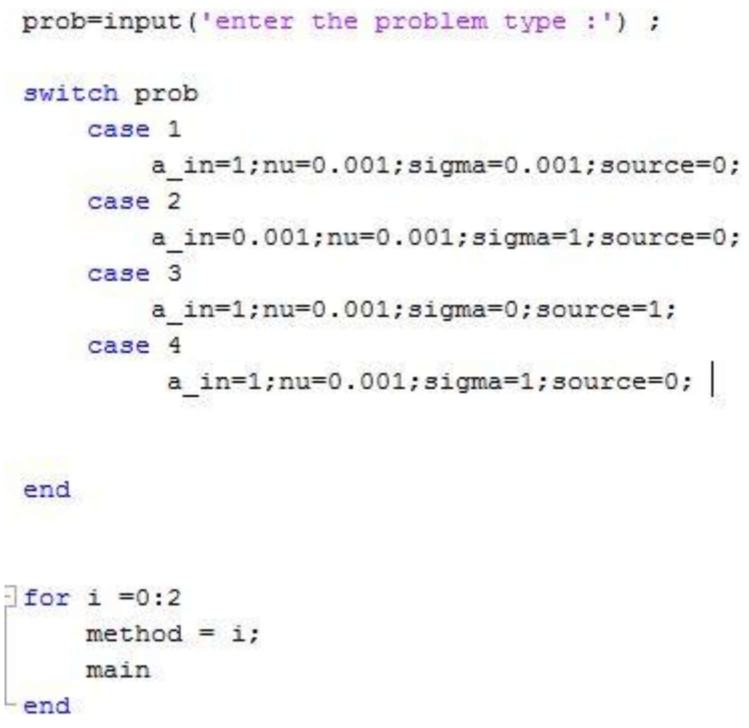
```
% BOUNDARY CONDITIONS
% Boundary conditions are imposed using Lagrange multipliers
nodes_y0 = [1:nx+1]';                              % Nodes on the boundary y=0
nodes_x1 = [2*(nx+1):nx+1:(ny+1)*(nx+1)]' ;        % Nodes on the boundary x=2
nodes_y1 = [ny*(nx+1)+nx:-1:ny*(nx+1)+1]' ;        % Nodes on the boundary y=3
nodes_x0 = [(ny-1)*(nx+1)+1:-(nx+1):nx+2]';        % Nodes on the boundary x=0
nodes = 1:max(max(T));
%%(NEUMANN and DIRECHLIT BC)%%
% nodes on which solution is u=1
nodesDir1 = nodes(X(:,2) > dom(4)/2 & (X(:,1) == dom(1) ))';
% nodes on which solution is u=0
nodesDir0 = nodes(X(:,1) == dom(2))';
```



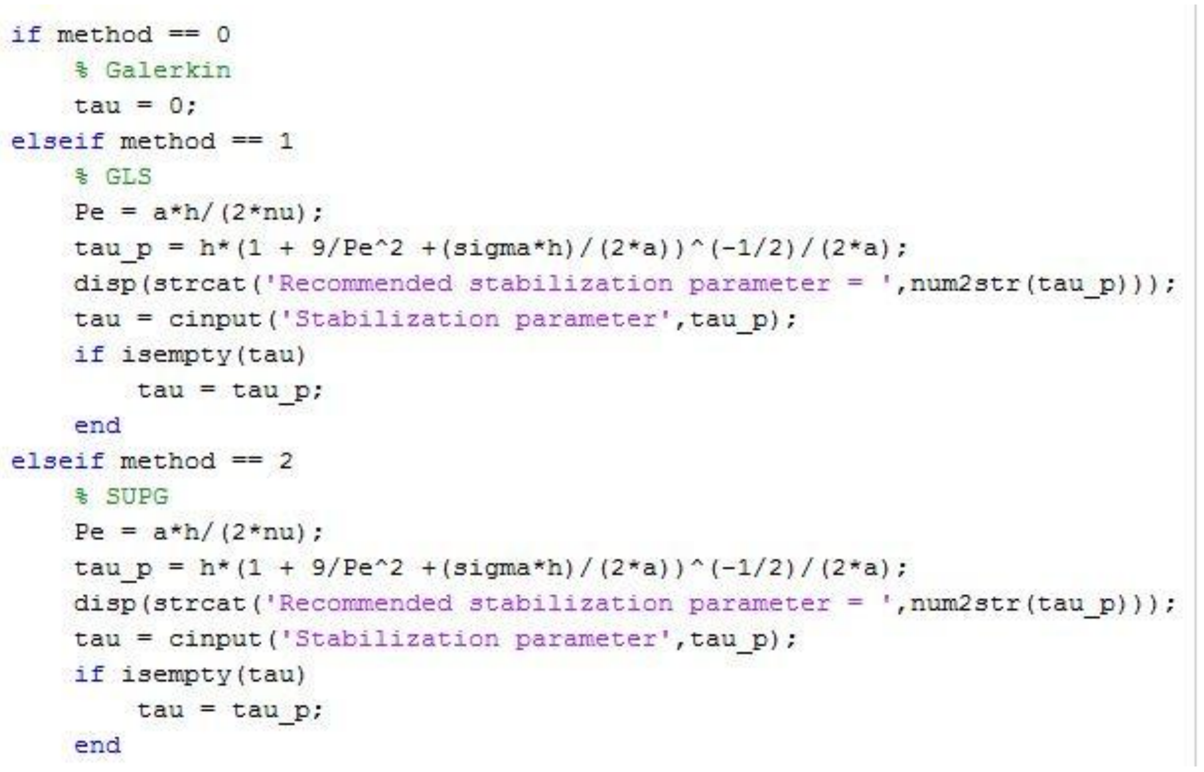
```
%%%%      Boundary condition matrix  %%%%%

%%%%%% first BC%%%%%%
%  C = [nodesDir1, ones(length(nodesDir1),1);
%  nodesDir0, ones(length(nodesDir0),1)];

%%%%%%%% second BC%%%%%%
  C = [nodesDir1, 2*ones(length(nodesDir1),1);
       nodesDir0, ones(length(nodesDir0),1)];
```



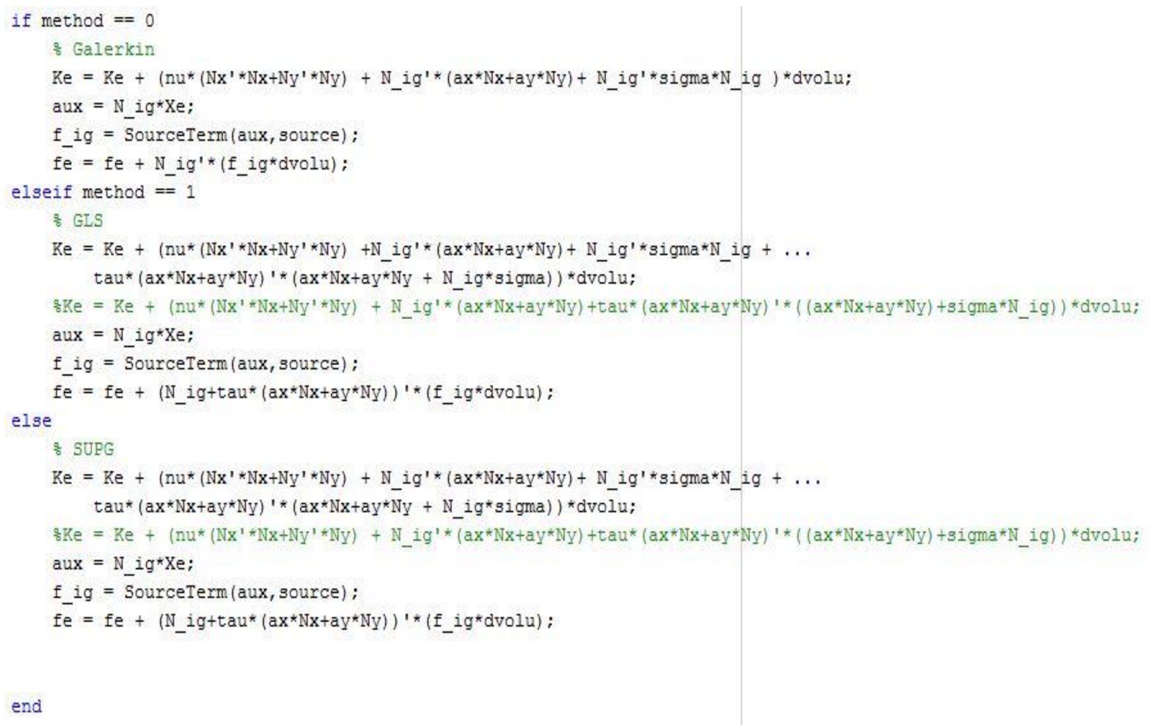
```
%  C = [nodesDir1, 2*ones(length(nodesDir1),1)]
%  N = nodesDir0;
%
% known_flux = [  -0.0002
%   -0.0009
%   0.0010
%    0.0007
%    -0.0002
%    -0.0090
%    -0.0396
%    -0.0797
%    -0.1001
%    -0.1060
%    -0.1246
%    -0.1497
%    -0.1659
%    -0.1882
%    -0.1030];
%
% f(N)=f(N)+known_flux;
```



```
prob=input('enter the problem type :') ;

switch prob
    case 1
        a_in=1;nu=0.001;sigma=0.001;source=0;
    case 2
        a_in=0.001;nu=0.001;sigma=1;source=0;
    case 3
        a_in=1;nu=0.001;sigma=0;source=1;
    case 4
         a_in=1;nu=0.001;sigma=1;source=0;


end


for i =0:2
    method = i;
    main
end
```

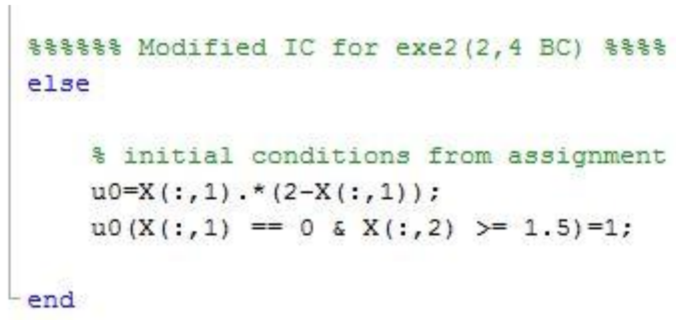


```
if method == 0
    % Galerkin
    tau = 0;
elseif method == 1
    % GLS
    Pe = a*h/(2*nu);
    tau_p = h*(1 + 9/Pe^2 +(sigma*h)/(2*a))^(-1/2)/(2*a);
    disp(strcat('Recommended stabilization parameter = ',num2str(tau_p)));
    tau = cinput('Stabilization parameter',tau_p);
    if isempty(tau)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

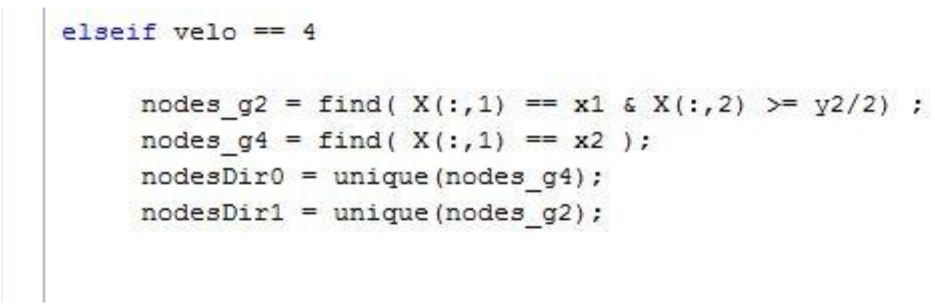


```
if method == 0
    % Galerkin
    Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny)+ N_ig'*sigma*N_ig )*dvolu;
    aux = N_ig*Xe;
    f_ig = SourceTerm(aux,source);
    fe = fe + N_ig'*(f_ig*dvolu);
elseif method == 1
    % GLS
    Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) +N_ig'*(ax*Nx+ay*Ny)+ N_ig'*sigma*N_ig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


end
```

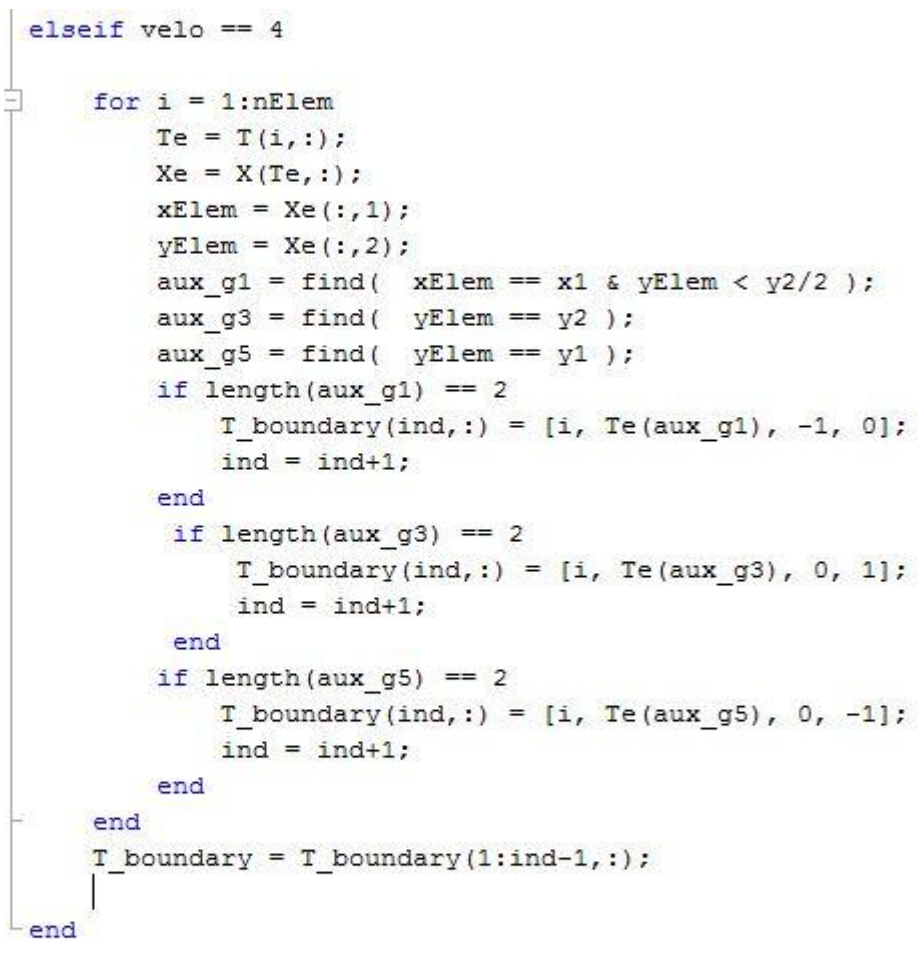
Exercise 2



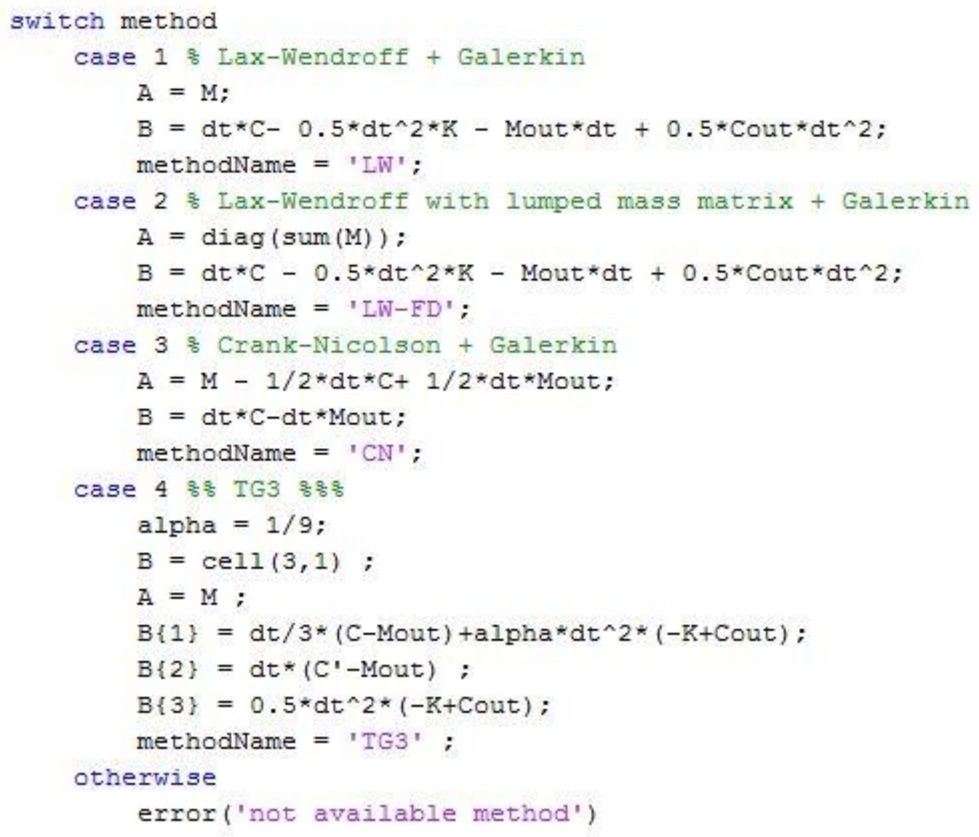
```
%%%%%% Modified IC for exe2(2,4 BC) %%%%
else

    % initial conditions from assignment
    u0=X(:,1).*(2-X(:,1));
    u0(X(:,1) == 0 & X(:,2) >= 1.5)=1;

end
```



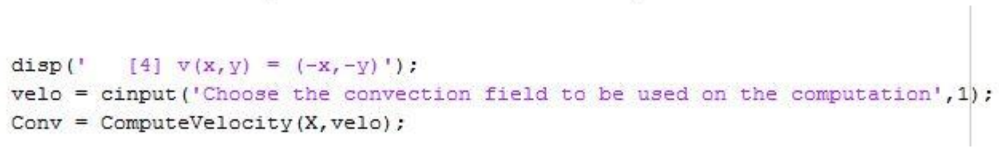
```
elseif velo == 4

    nodes_g2 = find( X(:,1) == x1 & X(:,2) >= y2/2) ;
    nodes_g4 = find( X(:,1) == x2 );
    nodesDir0 = unique(nodes_g4);
    nodesDir1 = unique(nodes_g2);
```



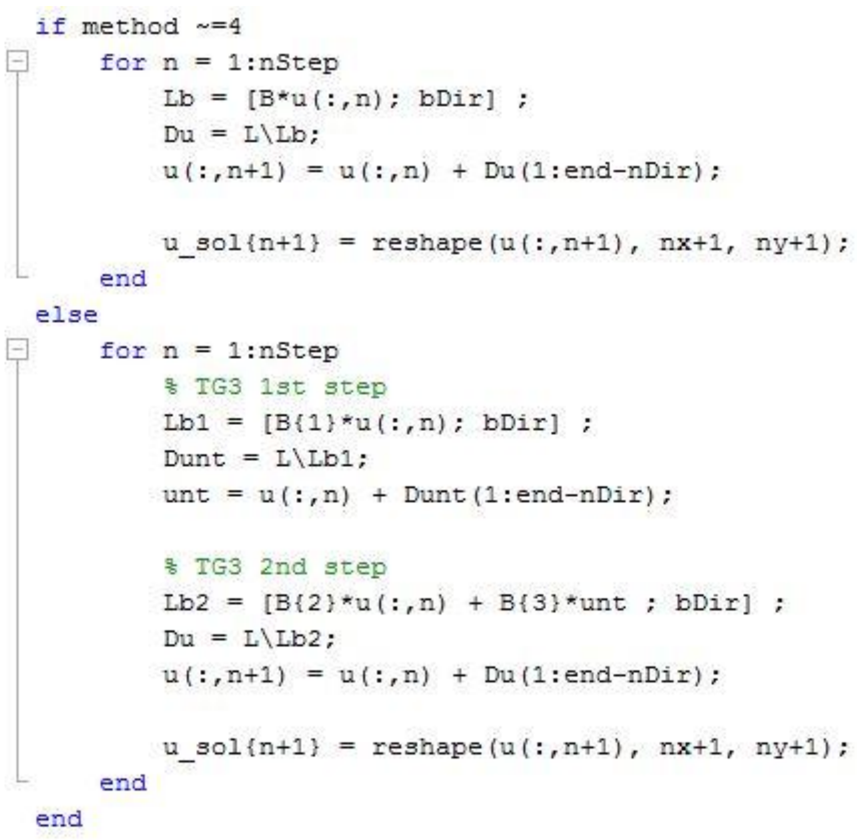
```
elseif velo == 4

    for i = 1:nElem
        Te = T(i,:);
        Xe = X(Te,:);
        xElem = Xe(:,1);
        yElem = Xe(:,2);
        aux_g1 = find(  xElem == x1 & yElem < y2/2 );
        aux_g3 = find(  yElem == y2 );
        aux_g5 = find(  yElem == y1 );
        if length(aux_g1) == 2
            T_boundary(ind,:) = [i, Te(aux_g1), -1, 0];
            ind = ind+1;
        end
         if length(aux_g3) == 2
             T_boundary(ind,:) = [i, Te(aux_g3), 0, 1];
             ind = ind+1;
         end
        if length(aux_g5) == 2
            T_boundary(ind,:) = [i, Te(aux_g5), 0, -1];
            ind = ind+1;
        end
    end
    T_boundary = T_boundary(1:ind-1,:);

end
```



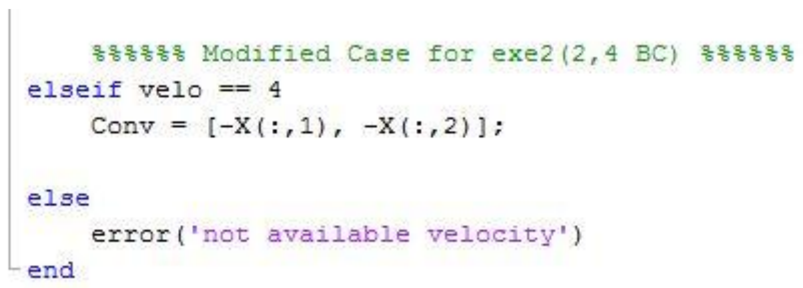
```
switch method
    case 1 % Lax-Wendroff + Galerkin
        A = M;
        B = dt*C- 0.5*dt^2*K - Mout*dt + 0.5*Cout*dt^2;
        methodName = 'LW';
    case 2 % Lax-Wendroff with lumped mass matrix + Galerkin
        A = diag(sum(M));
        B = dt*C - 0.5*dt^2*K - Mout*dt + 0.5*Cout*dt^2;
        methodName = 'LW-FD';
    case 3 % Crank-Nicolson + Galerkin
        A = M - 1/2*dt*C+ 1/2*dt*Mout;
        B = dt*C-dt*Mout;
        methodName = 'CN';
    case 4 %% TG3 %%%
        alpha = 1/9;
        B = cell(3,1) ;
        A = M ;
        B{1} = dt/3*(C-Mout)+alpha*dt^2*(-K+Cout);
        B{2} = dt*(C'-Mout) ;
        B{3} = 0.5*dt^2*(-K+Cout);
        methodName = 'TG3' ;
    otherwise
        error('not available method')
```



```
disp('   [4] v(x,y) = (-x,-y)');
velo = cinput('Choose the convection field to be used on the computation',1);
Conv = ComputeVelocity(X,velo);
```

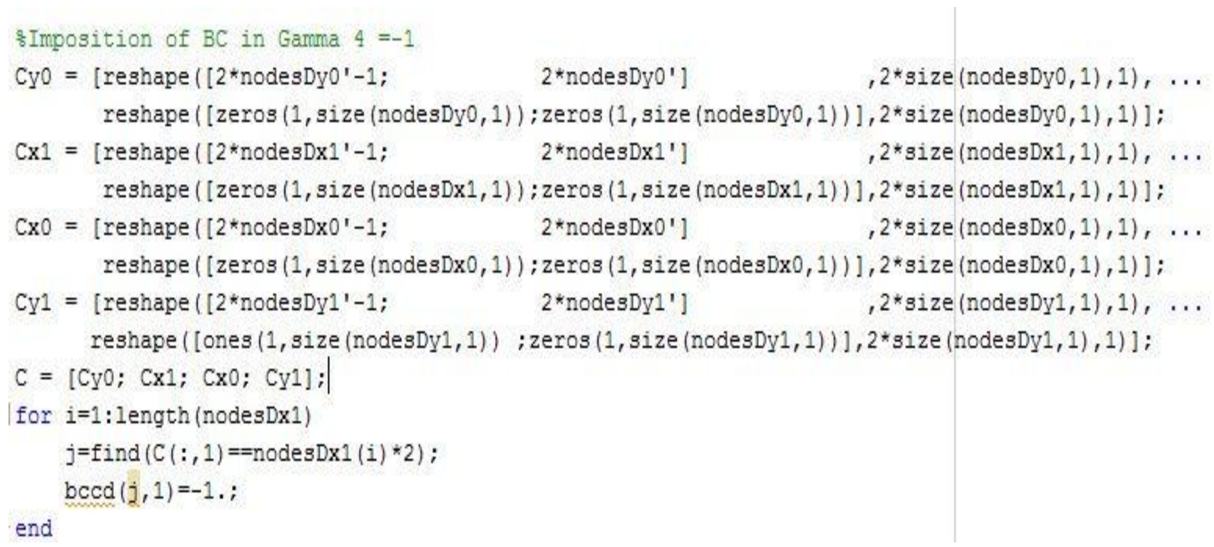


```
if method ~=4
    for n = 1:nStep
        Lb = [B*u(:,n); bDir] ;
        Du = L\Lb;
        u(:,n+1) = u(:,n) + Du(1:end-nDir);

        u_sol{n+1} = reshape(u(:,n+1), nx+1, ny+1);
    end
else
    for n = 1:nStep
        % TG3 1st step
        Lb1 = [B{1}*u(:,n); bDir] ;
        Dunt = L\Lb1;
        unt = u(:,n) + Dunt(1:end-nDir);

        % TG3 2nd step
        Lb2 = [B{2}*u(:,n) + B{3}*unt ; bDir] ;
        Du = L\Lb2;
        u(:,n+1) = u(:,n) + Du(1:end-nDir);

        u_sol{n+1} = reshape(u(:,n+1), nx+1, ny+1);
    end
end
```



```
    %%%%%% Modified Case for exe2(2,4 BC) %%%%%%
elseif velo == 4
    Conv = [-X(:,1), -X(:,2)];

else
    error('not available velocity')
end
```

Exercise 3



```
%Imposition of BC in Gamma 4 =-1
Cy0 = [reshape([2*nodesDy0'-1;          2*nodesDy0']               ,2*size(nodesDy0,1),1), ...
       reshape([zeros(1,size(nodesDy0,1));zeros(1,size(nodesDy0,1))],2*size(nodesDy0,1),1)];
Cx1 = [reshape([2*nodesDx1'-1;          2*nodesDx1']               ,2*size(nodesDx1,1),1), ...
       reshape([zeros(1,size(nodesDx1,1));zeros(1,size(nodesDx1,1))],2*size(nodesDx1,1),1)];
Cx0 = [reshape([2*nodesDx0'-1;          2*nodesDx0']               ,2*size(nodesDx0,1),1), ...
       reshape([zeros(1,size(nodesDx0,1));zeros(1,size(nodesDx0,1))],2*size(nodesDx0,1),1)];
Cy1 = [reshape([2*nodesDy1'-1;          2*nodesDy1']               ,2*size(nodesDy1,1),1), ...
      reshape([ones(1,size(nodesDy1,1)) ;zeros(1,size(nodesDy1,1))],2*size(nodesDy1,1),1)];
C = [Cy0; Cx1; Cx0; Cy1];
for i=1:length(nodesDx1)
    j=find(C(:,1)==nodesDx1(i)*2);
    bccd(j,1)=-1.;
end
```