Assignment 4

PRADEEP KUMAR BAL

March 30, 2018

2D Steady Convection-Diffusion-Reaction Problem:

The given problem is a homogeneous convection-diffusion-reaction problem on a unit square domain. The convective velocity is skew to the mesh with an angle of 30^o. Discontinuous Dirichlet boundary conditions are imposed on the inlet boundary. The source term s is considered to be 0. The domain has been shown in the figure below. The equation which is needed to be solved in the square domain $[0,1] * [0,1]$ can be expressed as

$$\mathbf{a}.\boldsymbol{\nabla} u - \boldsymbol{\nabla}.(\nu \boldsymbol{\nabla} u) + \sigma u = 1 \; in \quad \Omega$$

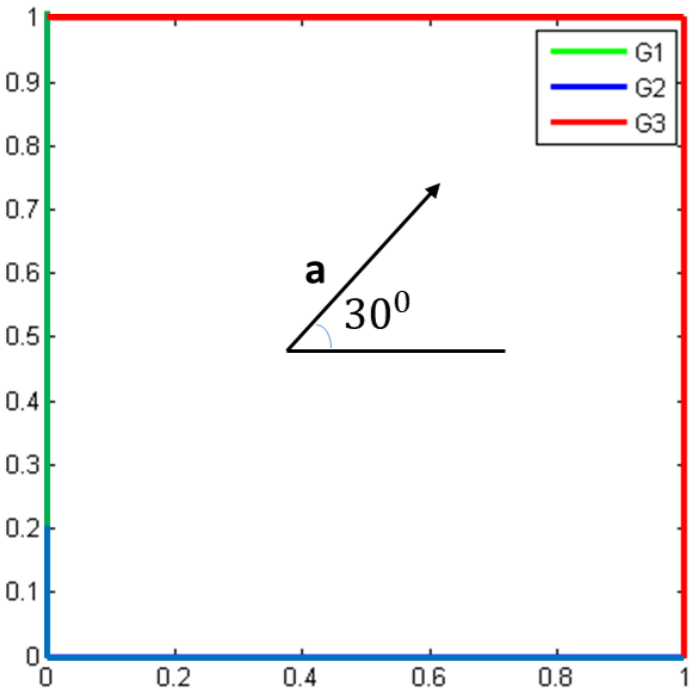

Figure 1: The Domain (On $G1 : u = 1; On\ G2 : u = 0;\ On\ G3$: Neumann or Dirichlet boundary Conditions can be applied as outlet boundary conditions)

Goal:

The goal is to implement the GLS method. The supplied modified code should work for both the linear and quadratics elements. To compare the obtained results using the zero Dirichlet boundary conditions at the outlet Boundaries (G3) with the results by implementing zero Neumann boundary conditions on the same outflow boundary. At the end we need to solve the given problem for both the Convection-reaction dominated case and the reaction dominated case.

Solution:

1. The GLS method has been implemented. For linear elements it gives similar results as we obtained using the SUPG method. The instabilities introduced by Galerkin are a little more amplified in GLS compared with SUPG. In the GLS method the stabilization term that affects the l.h.s. is symmetric. This symmetry is a major advantage in establishing stability. From a practical point of view there is no major difference between SUPG and GLS for linear elements.

$\mathcal{R}(u) = \mathbf{a}.\boldsymbol{\nabla} u - \boldsymbol{\nabla}.(\nu \boldsymbol{\nabla} u) + \sigma u - s = \mathcal{L}(u) - s;$

$where\ \mathcal{L}(u) = \mathbf{a}.\boldsymbol{\nabla} u - \boldsymbol{\nabla}.(\nu \boldsymbol{\nabla} u) + \sigma u$
For the GLS method the formulation can be written as:

$a(w,u) + c(\mathbf{a};w,u) + (w,\sigma u) + \sum_e \int_{\Omega^e} \mathcal{P}(w)\ \ \tau\ \mathcal{R}(u)\ d\Omega = (w,s) + (w,h)_{\Gamma_N}$
$\mathcal{P}(w) = \mathcal{L}(w) = \mathbf{a}.\boldsymbol{\nabla} w - \boldsymbol{\nabla}.(\nu \boldsymbol{\nabla} w) + \sigma w$

2. The code has been delineated in the annex for both the linear and quadratic elements. It is verified that the code works well for both linear and quadratic elements for the given 2D-steady-convection-diffusion-reaction problem.

3. The code has been modified to solve the steady convection-diffusion -reaction problem with zero Dirichlet boundary conditions on the outlet boundary and for the Neumann boundary conditions on the outlet boundary. The obtained results from the Galerkin,SUPG, Artificial diffusion and GLS methods have been depicted in the figures below. The case where $\sigma = 0$ has been considered for the comparisons of the behaviours of the methods. A mesh of 10 x 10 bilinear quadrilateral elements, $\nu = 10^{-4}\ and\ ||\mathbf{a}|| = 1$ have been considered to obtain the results corresponding to a mesh Peclet number of 10^4.

Downwind homogeneous essential boundary conditions: Here, we impose u = 0 on the outlet portion of the boundary. The solution now involves a thin boundary layer at the outlet. As shown in Figure 2, the crude 10 by 10 mesh has difficulty capturing the details of the solution. The Galerkin results are wildly oscillatory and bear no resemblance to the exact results. Better results are obtained with the stabilized formulations. The artificial diffusion technique introduces excessive numerical diffusion.

Downwind homogeneous natural boundary conditions: The results for this case are displayed in the Figure 3. Given the elevated value of the Peclet number, the solution is practically one of pure convection. The Galerkin method is not able to satisfactorily resolve the discontinuity and produces spurious oscillations.The artificial diffusion method and the SUPG method yield better results, but SUPG introduces less crosswind diffusion.

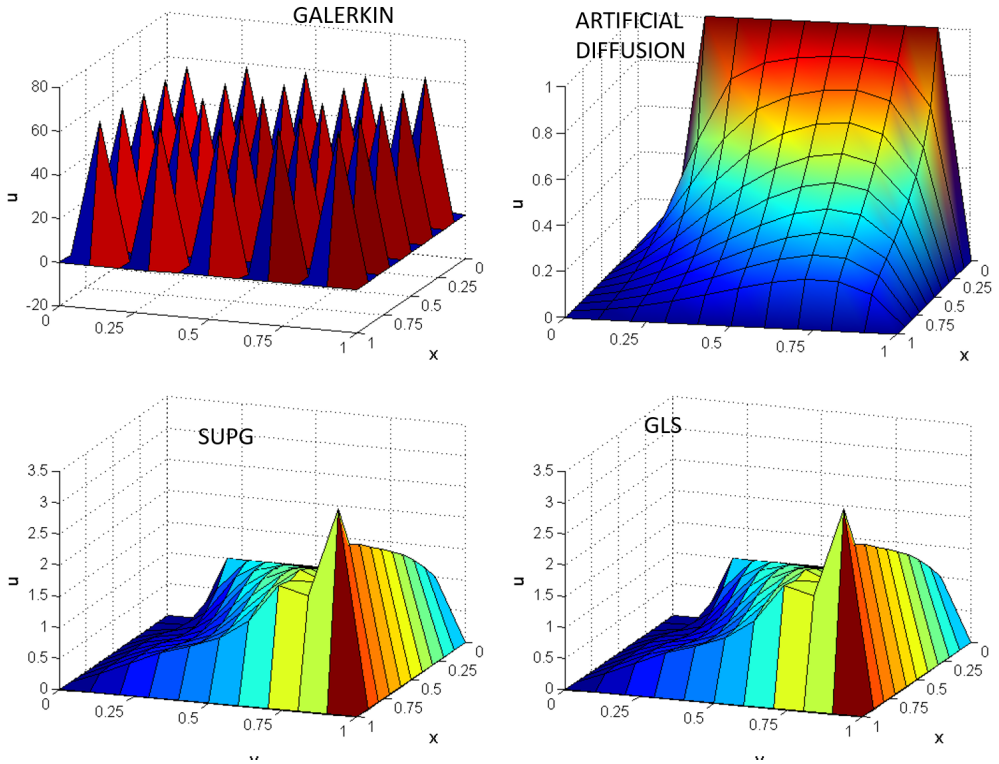

Figure 2: Downwind homogeneous essential boundary conditions with $\sigma = 0$ with bilinear quadrilateral elements

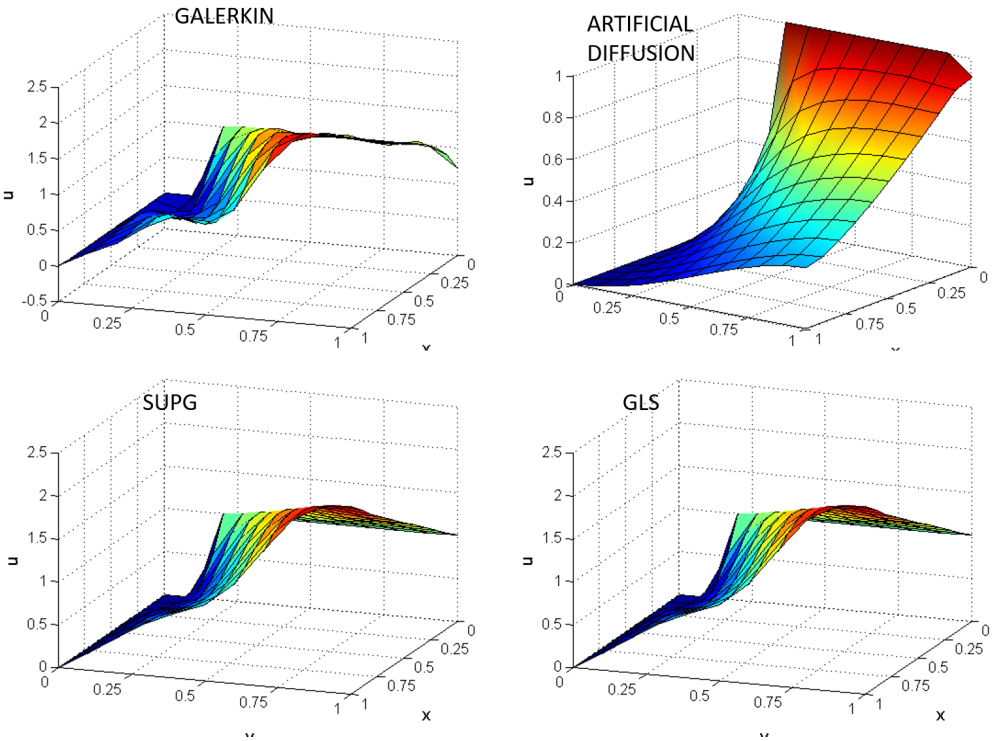

Figure 3: Downwind homogeneous natural boundary conditions with $\sigma = 0$ with bilinear quadrilateral elements

4.(a) **Convection-reaction dominated case**

The problem has been solved On the same unit square domain, the influence of the convection and reaction terms are compared for the Galerkin,Artificial diffusion, SUPG,and GLS formulations. The parameters $||\mathbf{a}|| = \frac{1}{2}; \nu = 10^{-4}; \sigma = 1$have been set. A mesh of $10 * 10$ elements have been considered. The obtained results have been depicted below in the Figure 4 with the zero Dirichlet boundary conditions on the outlet boundary. For this observation linear elements have been considered. It can be observed that the stabilized methods produce similar results.

4.(b) **Reaction dominated case**

The problem has been solved for a reaction dominated case using the parameters $||\mathbf{a}|| = 10^{-3}; \nu = 10^{-4}; \sigma = 1$. A mesh of $10 * 10$ elements have been considered. The obtained results from different methods have been depicted below in the Figure 5 with the zero Dirichlet boundary conditions on the outlet boundary. For this observation linear elements have been considered. It can be observed that in the presence of reaction (in particular for the reaction-dominated example), SGS reduces the oscillations near the boundary layers.

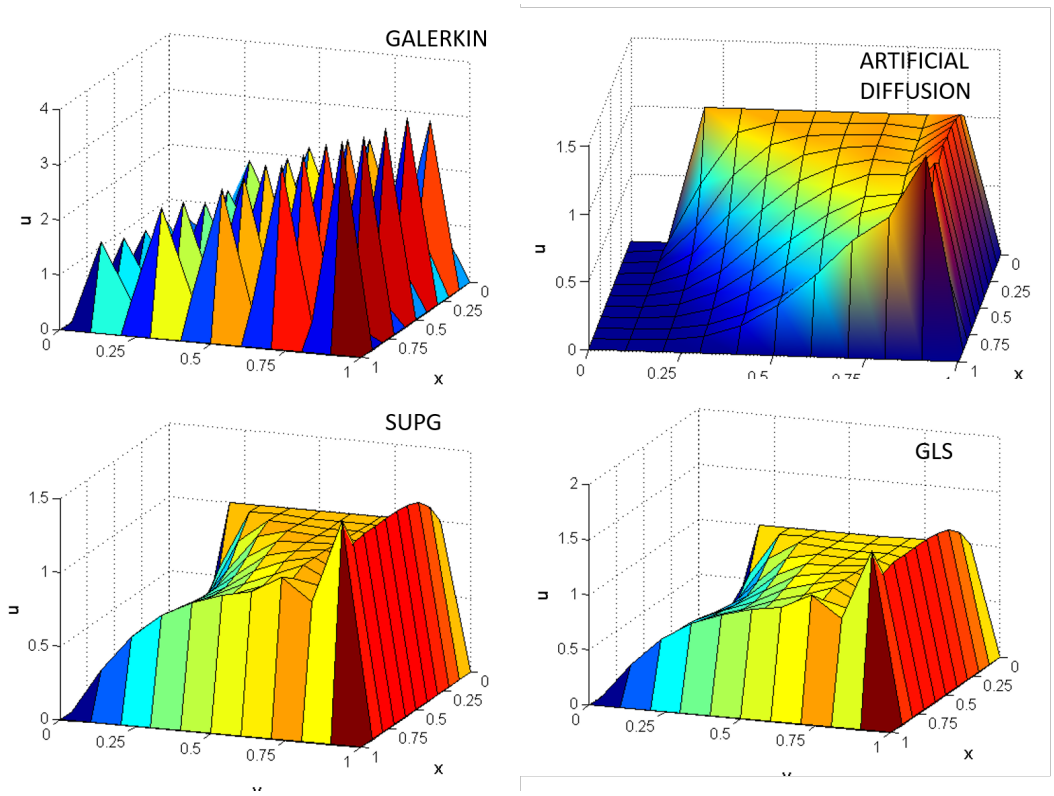

Figure 4: Convection-reaction dominated case with homogeneous essential boundary conditions with bilinear quadrilateral elements

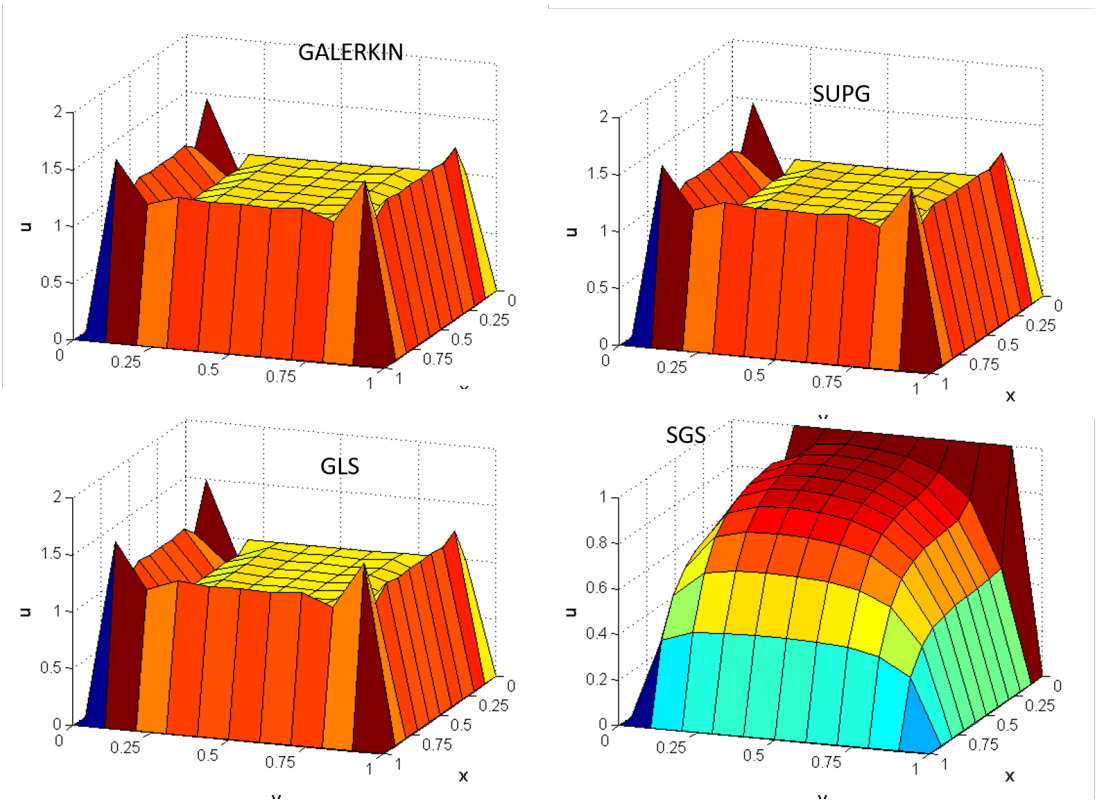

Figure 5: Reaction dominated case with homogeneous essential boundary conditions with bilinear quadrilateral elements

1. Modifications to solve Convection-diffusion Reaction Equation

Linear Elements:

```
function [K,f] = FEM_system(X,T,referenceElement,example)

if method == 1
    disp('Galerkin formulation');
    tau = 0;
elseif method == 2
    disp ('SUPG formulation');
    Pe = a*h/(2*nu);
    % alpha = coth(Pe)-1/Pe;
    % tau_p = alpha*h/(2*a);
    tau_p = h*(1 + 9/Pe^2 +(h*sigma/(2*a))^2)^(-1/2)/(2*a);
    disp(strcat('Recommended value for the stabilization parameter =
',num2str(tau_p)));
    tau = input('Stabilization parameter to be used (press return for using
the recommended one)= ');
    if isempty(tau)
        tau = tau_p;
    end
elseif method == 3
    disp('GLS formulation');
    Pe = a*h/(2*nu);
    % alpha = coth(Pe)-1/Pe;
    % tau_p = alpha*h/(2*a);
    tau_p = h*(1 + 9/Pe^2 +(h*sigma/(2*a))^2)^(-1/2)/(2*a);
    disp(strcat('Recommended value for the stabilization parameter =
',num2str(tau_p)));
    tau = input('Stabilization parameter to be used (press return for using
the recommended one)= ');
    if isempty(tau)
        tau = tau_p;
    end

elseif method==4
     % Artificial Diffusion
    Pe_x = ax*h/(2*nu);
    Pe_y = ay*h/(2*nu);
    alpha_x = coth(Pe_x)-1/Pe_x;
    alpha_y = coth(Pe_y)-1/Pe_y;
    tau = h*(ax*alpha_x + ay*alpha_y)/2;
    nubar_p = tau*a*a;
    disp(strcat('Recommended value for the artificial diffusion =
',num2str(nubar_p)));
    nubar = cinput('Artificial diffusion to be used', nubar_p);
    if isempty(nubar)
        nubar = nubar_p;
    end
    nu = nu + nubar;
    tau=0;
end
```

```
function [Ke,fe] = EleMat(Xe,nen,ngaus,wgp,N,Nxi,Neta,method,tau,velo,nu)
if method == 1          % Galerkin
        convi = ax*Nx+ay*Ny;
        Ke = Ke + (N_ig'*convi + nu*(Nx'*Nx+Ny'*Ny) +
sigma*N_ig'*N_ig)*dvolu;
        aux = Isopar(Xe,N_ig);
        f_ig = SourceTerm(aux);
        fe = fe + N_ig'*(f_ig*dvolu);
    elseif method == 2      % SUPG
        convi = ax*Nx+ay*Ny;
        Ke = Ke + (N_ig'*convi + nu*(Nx'*Nx+Ny'*Ny) + sigma*N_ig'*N_ig  ...
                + tau*(convi'*convi + convi'*sigma*N_ig))*dvolu ;
        aux = Isopar(Xe,N_ig);
        f_ig = SourceTerm(aux);
        fe = fe + (N_ig + tau*(ax*Nx+ay*Ny))'*(f_ig*dvolu);


    elseif method == 3      % GLS
        convi = ax*Nx + ay*Ny;
        Ke = Ke + (N_ig'*convi + nu*(Nx'*Nx+Ny'*Ny) + sigma*N_ig'*N_ig ...
                + tau*((convi + sigma*N_ig)'*(convi +
sigma*N_ig)))*dvolu ;%one term to be added
        aux = Isopar(Xe,N_ig);
        f_ig = SourceTerm(aux);
        fe = fe + (N_ig+ tau*(convi+sigma*N_ig))'*(f_ig*dvolu);

    elseif method==4                                %Artificial Diffusion
  % Artifficial diffusion
        Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) +
N_ig'*(ax*Nx+ay*Ny))*dvolu+sigma*(N_ig'*N_ig)*dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux);
        fe = fe + N_ig'*(f_ig*dvolu);
```

2. Boundary Conditions in main.m

```
% BOUNDARY CONDITIONS
% Essential boundary conditions are imposed on the whole boundary
% 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=1
nodes_y1 = [ny*(nx+1)+nx:-1:ny*(nx+1)+1]' ;     % Nodes on the boundary y=1
nodes_x0 = [(ny-1)*(nx+1)+1:-(nx+1):nx+2]';     % Nodes on the boundary x=0

plot_BC;
disp ('On G1 solution is set to 1');
disp ('On G2 solution is set to 0');
disp ('On G3 either natural or essential homogeneous boundary conditions
can be imposed.')
BC = input ('Condiciones en G3 (1:Neumann, 2: Dirichlet):  ');
if BC == 1
    % nodes on which solution is u=1
    nodesDir1 = [nodes_x0(1:ny-2)];
    % nodes on which solution is u=0
    nodesDir0 = [nodes_x0(ny-1);nodes_y0];
    % Boundary condition matrix
    C = [nodesDir1, ones(length(nodesDir1),1);
         nodesDir0, zeros(length(nodesDir0),1)];
elseif BC == 2
    % nodes on which solution is u=1
```

```
    nodesDir1 = [nodes_x0(1:ny-2)];
    % nodes on which solution is u=0
    nodesDir0 = [nodes_x0(ny-1); nodes_y0; nodes_x1; nodes_y1 ];
    % Boundary condition matrix
    C = [nodesDir1, ones(length(nodesDir1),1);
         nodesDir0, zeros(length(nodesDir0),1)];
else
    error ('Error imposing boundary conditions' );
end
```

Quadratic Elements (GLS)

```
 % GLS
Pe = a*h/(2*nu);
tau = h*(1 + 9/Pe^2 +(sigma*h)/(2*a))^(-1/2)/(2*a);
Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) +N_ig'*(ax*Nx+ay*Ny)+ N_ig'*sigma*N_ig + ...
tau*(ax*Nx+ay*Ny+sigma*N_ig)'*(ax*Nx+ay*Ny + N_ig*sigma))*dvolu;
for i=1:nedofV
if mod(i,2)==1
DKe(:,:,i) = DKe(:,:,i) + (N_ig'*(N_ig((i+1)/2)*Nx) ...
+ tau*(ax*Nx+ay*Ny)'*(N_ig((i+1)/2)*Nx) ...
+ tau*(N_ig((i+1)/2)*Nx)'*(ax*Nx+ay*Ny) )*dvolu ;
else
DKe(:,:,i) = DKe(:,:,i) + (N_ig'*(N_ig(i/2)*Ny) ...
+ tau*(ax*Nx+ay*Ny)'*(N_ig(i/2)*Ny) ...
+ tau*(N_ig(i/2)*Ny)'*(ax*Nx+ay*Ny) )*dvolu ;
end
end
f_ig = SourceTerm(a);
fe = fe + (N_ig+tau*(ax*Nx+ay*Ny+sigma*N_ig))'*(f_ig*dvolu);
for i=1:nedofV
if mod(i,2)==1
Dfe(:,i) = Dfe(:,i) +
(N_ig+tau*(ax*Nx+ay*Ny+sigma*N_ig))'*(Df_ig*dvolu)*N_ig((i+1)/2)^2*ue((i+1)
/2)/a;
else
Dfe(:,i) = Dfe(:,i) +
(N_ig+tau*(ax*Nx+ay*Ny+sigma*N_ig))'*(Df_ig*dvolu)*N_ig(i/2)^2*ve(i/2)/a;
end
end
```