Finite element in Fluid

Exercise-1

Md Tariqul Islam

Date: 22/05/2017

A)

The main equation is:

$$\frac{d\rho}{dt} + a.\nabla\rho - \nabla.(\mu\nabla\rho) + \sigma\rho = s$$

The weak form is,

Multiply the above equation with a test function “w”

$$\int w\frac{d\rho}{dt} + \int wa.\nabla\rho - \int w\nabla.(\mu\nabla\rho) + \rho = \int ws$$

Integration of parts leads to the weak form of the equation,

$$\int w\frac{d\rho}{dt}\,d\Omega + \int wa.\nabla\rho\ d\Omega + \int \nabla w.(\mu\nabla\rho)\ d\Omega - \int w(\mu\nabla\rho).n\,d\Gamma + \int w\sigma\rho\,d\Omega = \int ws\ d\Omega$$

Test functions are chosen such that the directlet boundary conditions are automatically satisfied and the in this problem, the neumann conditions are 0.

The Implicit method, R_{11},

$$\frac{\Delta\rho}{\Delta t} - W\Delta\rho_t = w\rho_t^n$$

Where ,

W=1/2

w=1

the above equation is then multiplied with a test function “q” to get the weak form and replacing $\rho_t = S - a.\nabla\rho + \nabla.(\mu\nabla\rho) - \sigma\rho$

$$\int q\frac{\Delta\rho}{\Delta t}\,d\Omega - \int qW\Delta\rho_t\,d\Omega = \int qw\rho_t^n\ d\Omega$$

By integration by parts, the final equation for weak form is obtained,

$$\int q\frac{\Delta\rho}{\Delta t}\,d\Omega + \int qWa.\nabla(\Delta\rho)\ d\Omega + \int \nabla q.W\mu\nabla(\Delta\rho)\ d\Omega + \int qW\sigma(\Delta\rho)\,d\Omega = \int q(W\Delta s + ws^n)\,d\Omega - \int qwa.\nabla(\rho^n)\ d\Omega - \int \nabla q.w\mu\nabla(\rho^n)\ d\Omega - \int qw\,\sigma\rho^n\ d\Omega$$

For the stabilization of the above weak form equation, stabilization term has been introduced.

$$\int q\frac{\Delta\rho}{\Delta t}\,d\Omega + \int qWa.\nabla(\Delta\rho)\,d\Omega + \int \nabla q.W\mu\nabla(\Delta\rho)\,d\Omega + \int qW\sigma(\Delta\rho)\,d\Omega + \sum(\boldsymbol{\tau\mathcal{P}(q)}, \boldsymbol{\mathcal{R}(\Delta\rho)}) =$$
$$\int q(W\Delta s + ws^n)\,d\Omega - \int qwa.\nabla(\rho^n)\,d\Omega - \int \nabla q.w\mu\nabla(\rho^n)\,d\Omega - \int qw\,\sigma\rho^n\,d\Omega$$

Where,

$$\mathcal{R}(\Delta\rho) = \frac{\Delta\rho}{\Delta t} + W\mathcal{L}(\Delta\rho) - w\,[s^n - \mathcal{L}(\rho^n)] - W\Delta s$$

$$\mathcal{L} = a.\nabla - \nabla.(\mu\nabla) + \sigma$$

For **SUPG**: $\mathcal{P}(q) = W\,(a.\nabla)q$

For **GLS**: $\mathcal{P}(q) = \frac{q}{\Delta t} + W[(a.\nabla)q - \nabla.(\mu\nabla q) + \sigma q\,]$

From the equation above, different Matrix and vector can be defined :

Mass matrix $M = \int N_a N_b\,d\Omega$

Convection matrix $C = \int N_a(a.\nabla N_b)\,d\Omega$

Diffusion Matrix $D = \int N_a(\mu\nabla N_b)\,d\Omega$

Reaction Matrix $R = \int N_a\sigma N_b\,d\Omega$

Stabilization Matrix $S = \sum(\tau\mathcal{P}(q), \mathcal{R}(\Delta\rho))$

Source vector $SV = \int N_a(W\Delta s + ws^n)\,d\Omega$

The system that has to be solved is:

$$\frac{1}{\Delta t}M + WC + WD + WR + S = SV - wC - wS - w\,\sigma\,M$$

The Implicit method, R_{22},

$$\frac{\Delta\rho}{\Delta t} - W\Delta\rho_t = w\rho_t^n$$

Where ,

$$W = \frac{1}{24}\begin{bmatrix} 7 & -1 \\ 13 & 5 \end{bmatrix}$$

$$w = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

the above equation is then multiplied with a test function "q" to get the weak form and replacing $\rho_t = S - a.\nabla\rho + \nabla.(\mu\nabla\rho) - \sigma\rho$

$$\int q\frac{\Delta\rho}{\Delta t}\,d\Omega - \int qW\Delta\rho_t\,d\Omega = \int qw\rho_t^n\,d\Omega$$

By integration by parts, the final equation for weak form is obtained,

$$\int q\frac{\Delta\rho}{\Delta t}\,d\Omega + \int qWa.\nabla(\Delta\rho)\,d\Omega + \int \nabla q.W\mu\nabla(\Delta\rho)\,d\Omega + \int qW\sigma(\Delta\rho)\,d\Omega = \int q(W\Delta s + ws^n)\,d\Omega - \int qwa.\nabla(\rho^n)\,d\Omega - \int \nabla q.w\mu\nabla(\rho^n)\,d\Omega - \int qw\,\sigma\rho^n\,d\Omega$$

For the stabilization of the above weak form equation, stabilization term has been introduced.

$$\int q\frac{\Delta\rho}{\Delta t}\,d\Omega + \int qWa.\nabla(\Delta\rho)\,d\Omega + \int \nabla q.W\mu\nabla(\Delta\rho)\,d\Omega + \int qW\sigma(\Delta\rho)\,d\Omega + \sum(\boldsymbol{\tau\mathcal{P}}(\boldsymbol{q}), \boldsymbol{\mathcal{R}}(\boldsymbol{\Delta\rho})) = \int q(W\Delta s + ws^n)\,d\Omega - \int qwa.\nabla(\rho^n)\,d\Omega - \int \nabla q.w\mu\nabla(\rho^n)\,d\Omega - \int qw\,\sigma\rho^n\,d\Omega$$

Where,

$$\mathcal{R}(\Delta\rho) = \frac{\Delta\rho}{\Delta t} + W\mathcal{L}(\Delta\rho) - w\,[s^n - \mathcal{L}(\rho^n)] - W\Delta s$$

$$\mathcal{L} = a.\nabla - \nabla.(\mu\nabla) + \sigma$$

For **SUPG**: $\mathcal{P}(q) = W(a.\nabla)q$

For **GLS**: $\mathcal{P}(q) = \frac{q}{\Delta t} + W[(a.\nabla)q - \nabla.(\mu\nabla q) + \sigma q]$

From the equation above, different Matrix and vector can be defined :

Mass matrix M = $\int N_a N_b \, d\Omega$

Convection matrix C = $\int N_a (a.\nabla N_b) \, d\Omega$

Diffusion Matrix D = $\int N_a (\mu \nabla N_b) \, d\Omega$

Reaction Matrix R = $\int N_a \sigma N_b \, d\Omega$

Stabilization Matrix S = $\sum(\tau \mathcal{P}(q), \mathcal{R}(\Delta\rho))$

Source vector SV = $\int N_a (W\Delta s + w s^n) \, d\Omega$

The system that has to be solved is:

$\frac{1}{\Delta t}$ M + WC +WD + WR +S = SV – wC –wS - w σ M

The Explicit method, R_{20}

$$\rho^{n+\frac{1}{2}} = \rho^n + \frac{\Delta t}{2}\rho_t^n \quad \text{---------- 1}$$

$$\rho^{n+\frac{1}{2}} = \rho^n + \Delta t \rho_t^{n+\frac{1}{2}} \quad \text{----------2}$$

The above equations are then multiplied with a test function “w” to get the weak form and replacing ρ_t = S - a.∇ρ + ∇.(μ∇ρ) - σρ

From first equation,

$$\int w\rho^{n+\frac{1}{2}} \, d\Omega = \int w\rho^n \, d\Omega + \int w\frac{\Delta t}{2}\rho_t^n \, d\Omega$$

By integration by parts, the final equation for weak form is obtained,

$$\int w\rho^{n+\frac{1}{2}} \, d\Omega = \int w\rho^n \, d\Omega + \int w\frac{\Delta t}{2}s^n \, d\Omega - \int w\frac{\Delta t}{2}a.\nabla\rho^n \, d\Omega + \int \nabla w.\frac{\Delta t}{2}(\mu\nabla\rho^n) \, d\Omega - \int w\frac{\Delta t}{2}\sigma\rho^n \, d\Omega$$

For the stabilization of the above weak form equation, stabilization term has been introduced.

$$\int w\rho^{n+\frac{1}{2}}\ d\Omega + \sum(\boldsymbol{\tau}\boldsymbol{\mathcal{P}}(\boldsymbol{q}), \boldsymbol{\mathcal{R}}(\boldsymbol{\Delta\rho})) = \int w\rho^{n}\ d\Omega + \int w\frac{\Delta t}{2}s^{n}\ d\Omega - \int w\frac{\Delta t}{2}a.\nabla\rho^{n}\ d\Omega + \int \nabla w.\frac{\Delta t}{2}(\mu\nabla\rho^{n})\ d\Omega - \int w\frac{\Delta t}{2}\sigma\rho^{n}\ d\Omega$$

Where,

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

$$\mathcal{L} = a.\nabla - \nabla.(\mu\nabla) + \sigma$$

For **SUPG**: $\mathcal{P}(q) = W\ (a.\nabla)q$

For **GLS**: $\mathcal{P}(q) = \frac{q}{\Delta t} + W[(a.\nabla)q - \nabla.(\mu\nabla q) + \sigma q\ ]$

From the equation above, different Matrix and vector can be defined :

Mass matrix M = $\int N_a N_b\ d\Omega$

Convection matrix C = $\int N_a(a.\nabla N_b)\ d\Omega$

Diffusion Matrix D = $\int N_a(\mu\nabla N_b)\ d\Omega$

Reaction Matrix R = $\int N_a\sigma N_b\ d\Omega$

Stabilization Matrix S = $\sum(\tau\mathcal{P}(q), \mathcal{R}(\Delta\rho))$

Source vector SV = $\int N_a(s^{n})\ d\Omega$

The system that has to be solved is:

$$M + S = M + \frac{\Delta t}{2}SV - \frac{\Delta t}{2}C + \frac{\Delta t}{2}D - \frac{\Delta t}{2}R$$

From second equation,

$$\int w\rho^{n+\frac{1}{2}}\ d\Omega = \int w\rho^{n}\ d\Omega + w\Delta t\rho_t^{n+\frac{1}{2}}$$

By integration by parts, the final equation for weak form is obtained,

$$\int w\rho^{n+\frac{1}{2}}\ d\Omega = \int w\rho^{n}\ d\Omega + \int w\Delta t s^{n+\frac{1}{2}}\ d\Omega - \int w\Delta t a.\nabla\rho^{n+\frac{1}{2}}\ d\Omega + \int \nabla w.\Delta t(\mu\nabla\rho^{n+\frac{1}{2}})\ d\Omega - \int w\Delta t\sigma\rho^{n+\frac{1}{2}}\ d\Omega$$

For the stabilization of the above weak form equation, stabilization term has been introduced.

$$\int w\rho^{n+\frac{1}{2}}\ d\Omega + \sum(\boldsymbol{\tau\mathcal{P}}(\boldsymbol{q}),\boldsymbol{\mathcal{R}}(\boldsymbol{\Delta\rho})) = \int w\rho^{n}\ d\Omega + \int w\Delta t s^{n+\frac{1}{2}}\ d\Omega - \int w\Delta t a.\nabla\rho^{n+\frac{1}{2}}\ d\Omega + \int \nabla w.\Delta t(\mu\nabla\rho^{n+\frac{1}{2}})\ d\Omega - \int w\Delta t\sigma\rho^{n+\frac{1}{2}}\ d\Omega$$

Where,

$$\mathcal{R}(\Delta\rho) = \frac{\Delta\rho}{\Delta t} + W\mathcal{L}(\Delta\rho) - w\,[s^n - \mathcal{L}(\rho^n)] - W\Delta s$$

$$\mathcal{L} = a.\nabla - \nabla.(\mu\nabla) + \sigma$$

For **SUPG**: $\mathcal{P}(q) = W(a.\nabla)q$

For **GLS**: $\mathcal{P}(q) = \frac{q}{\Delta t} + W[(a.\nabla)q - \nabla.(\mu\nabla q) + \sigma q]$

From the equation above, different Matrix and vector can be defined :

Mass matrix $M = \int N_a N_b\ d\Omega$

Convection matrix $C = \int N_a(a.\nabla N_b)\ d\Omega$

Diffusion Matrix $D = \int N_a(\mu\nabla N_b)\ d\Omega$

Reaction Matrix $R = \int N_a\sigma N_b\ d\Omega$

Stabilization Matrix $S = \sum(\tau\mathcal{P}(q),\mathcal{R}(\Delta\rho))$

Source vector $SV = \int N_a(s^n)\ d\Omega$

The system that has to be solved is:

$$M + S = M + \Delta t\,SV - \Delta t C + \Delta t D - \Delta t R$$

D)

Considering the steady state case, the problem is solved with both linear and quadrilateral element for 3 different cases with various velocity, diffusion parameter, reaction and source. The results are compared with GLS and SUPG discretization and are shown below:

First case

a= (-1, 0), μ =10^{-3}, σ= 10^{-3} , s =0

Using GLS

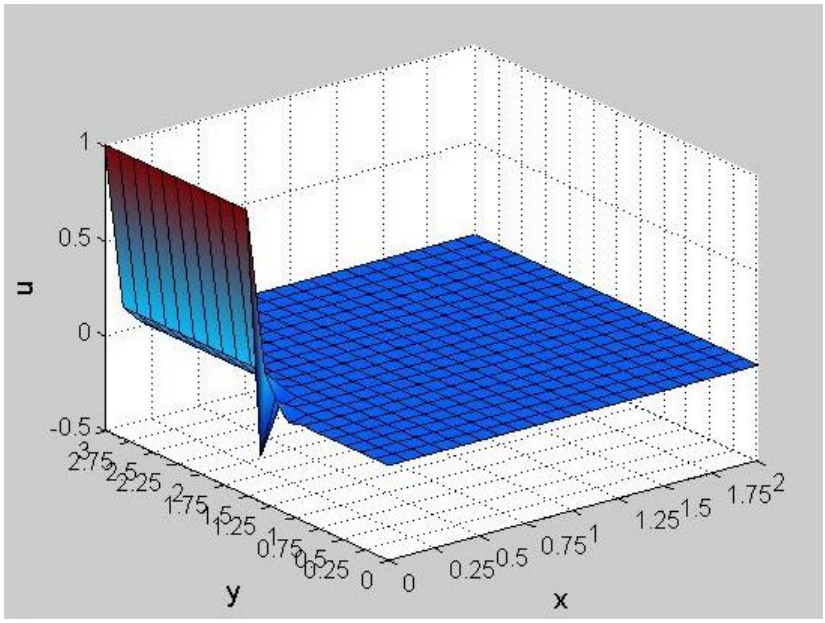

Fig 1: Linear element

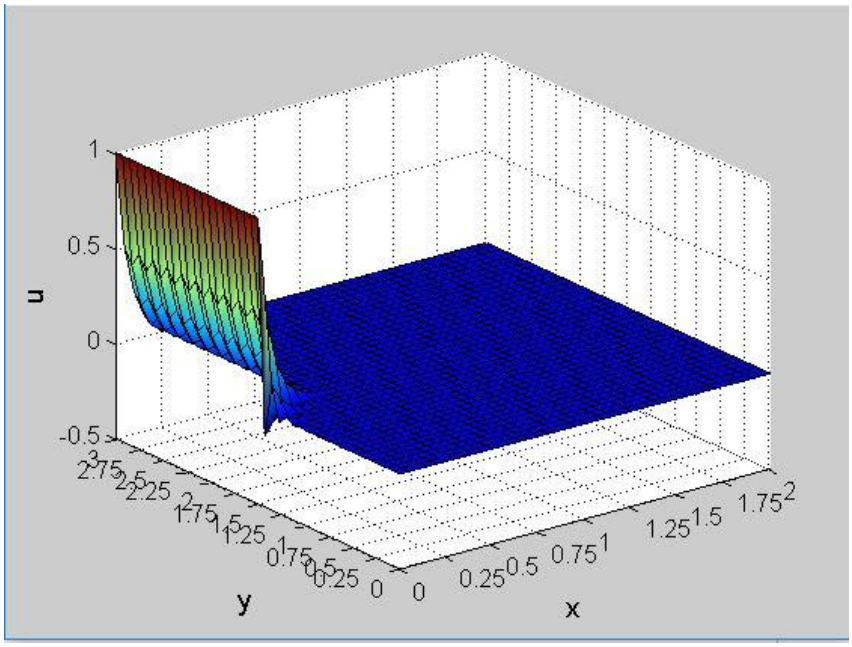

Fig 2: Quadratic element

Using SUPG

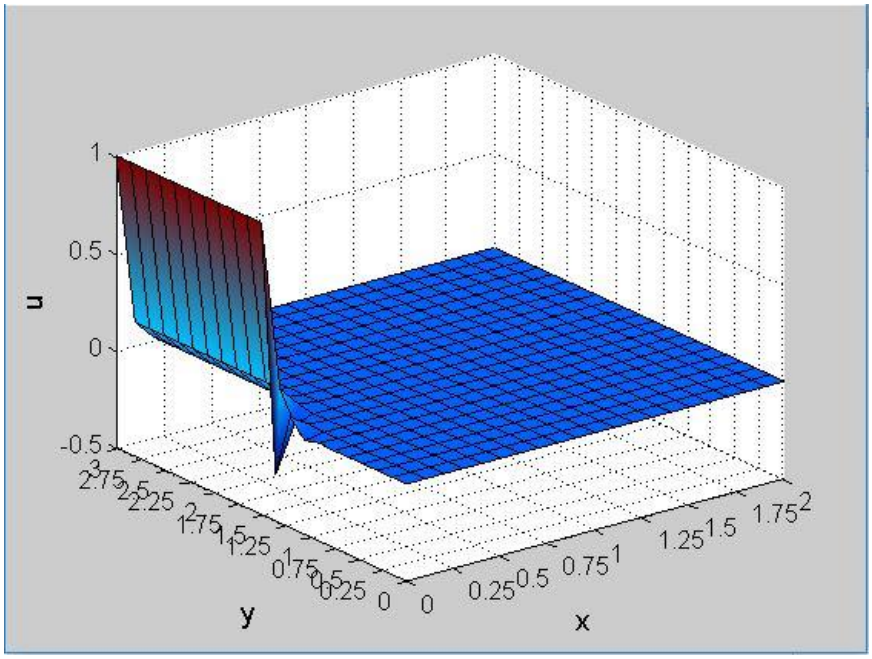

Fig 3: Linear element

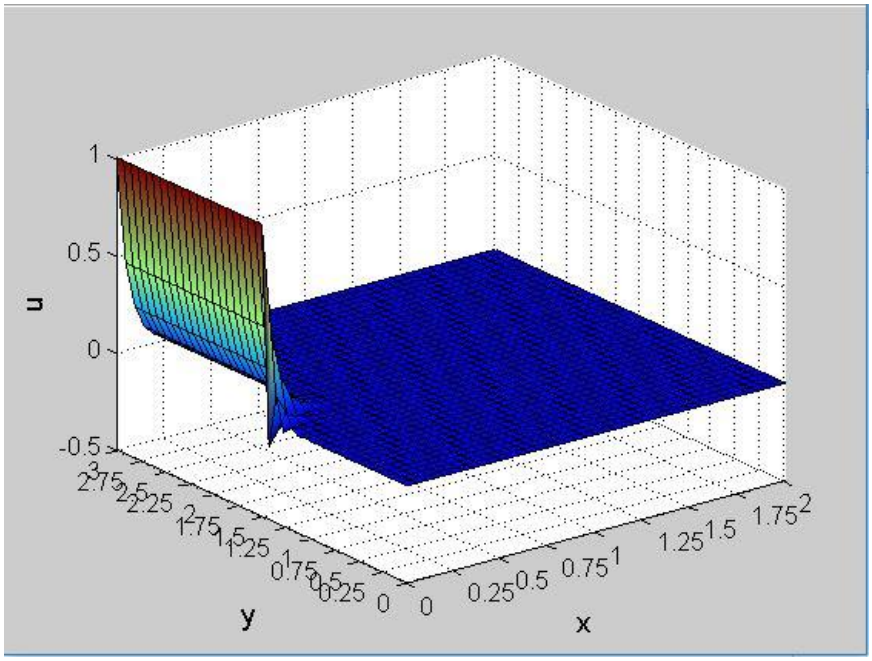

Fig 4: Quadratic element

Both GLS and SUPG showed similar results. But when compared between results using linear and quadratic element, it can be seen that the results obtained with linear element has more oscillation than the results obtained using quadratic element for both GLS and SUPG. This is due to the fact that, quadratic element can approximate result more accurately than linear element.

Second case

a= $(-10^{-3}, 0)$, μ =10^{-3}, σ= 1 , s =0

Using GLS

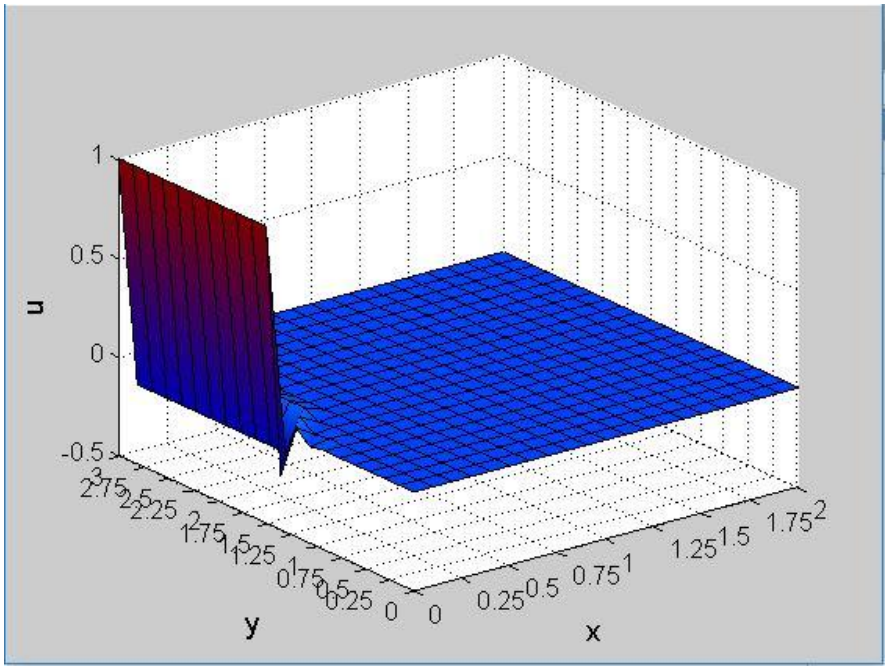

Fig 5: Linear element

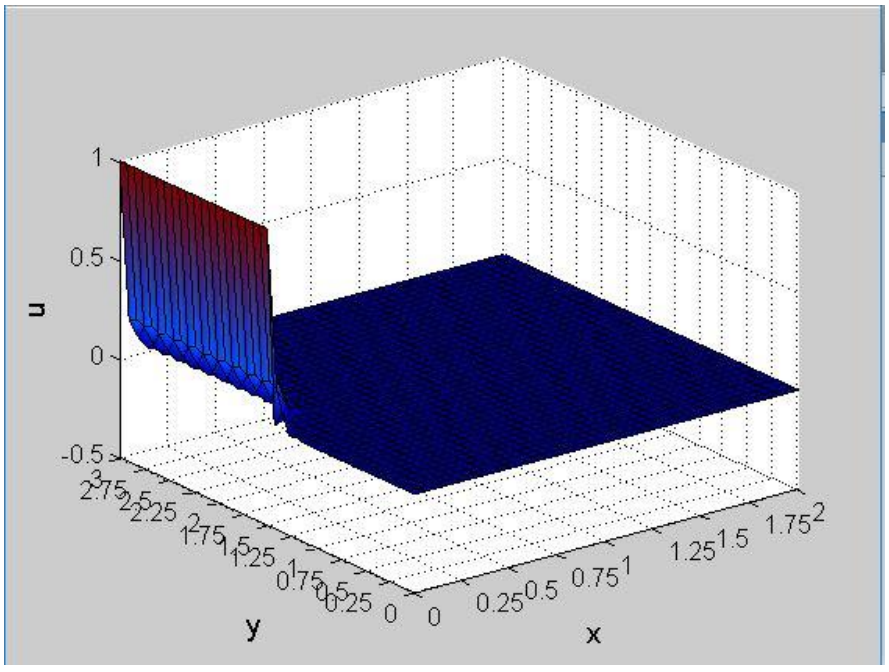

Fig 6: Quadratic element

Using SUPG

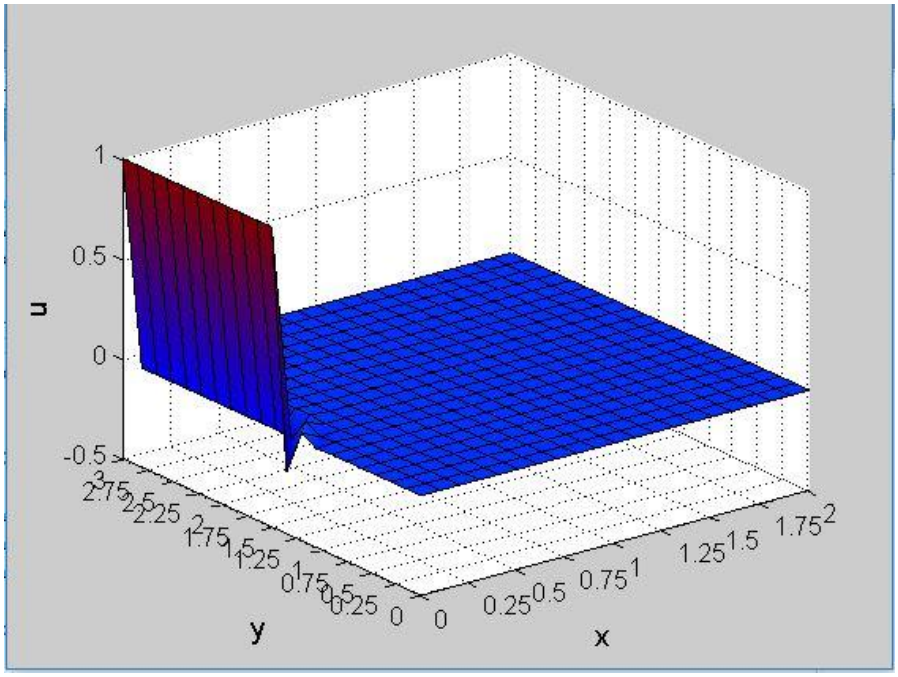

Fig 7: Linear element

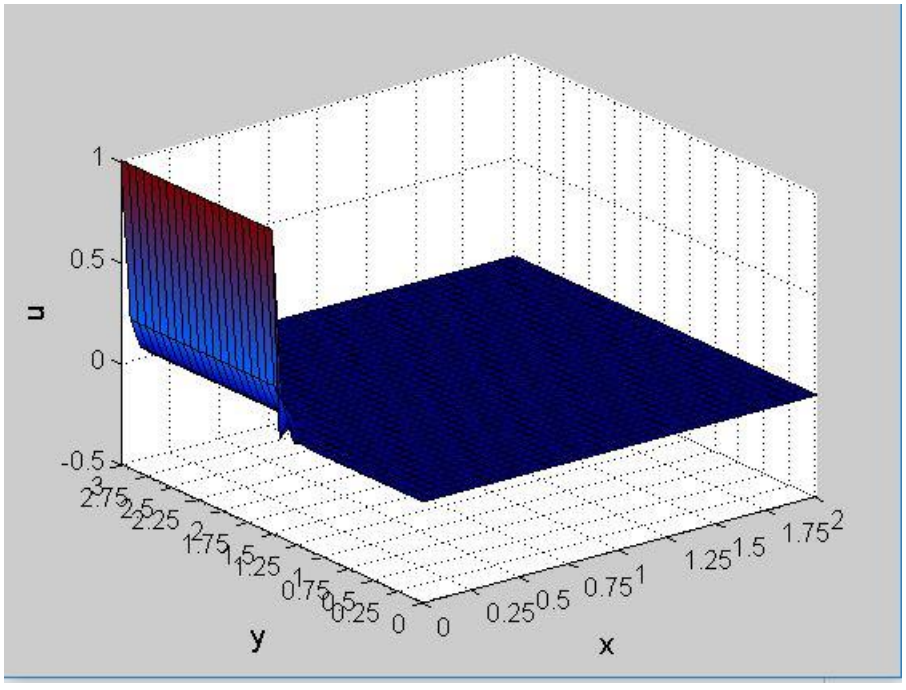

Fig 8: Quadratic element

For the second case as well, both GLS and SUPG showed similar results. But when compared between results using linear and quadratic element, the later one has much less oscillation and the solution is smoother compare to linear element. It was anticipated as quadratic element are better in approximating solutions.

Third case

a= $(-10^{-3}, 0)$, μ =10^{-3}, σ= 0 , s =1

Using GLS

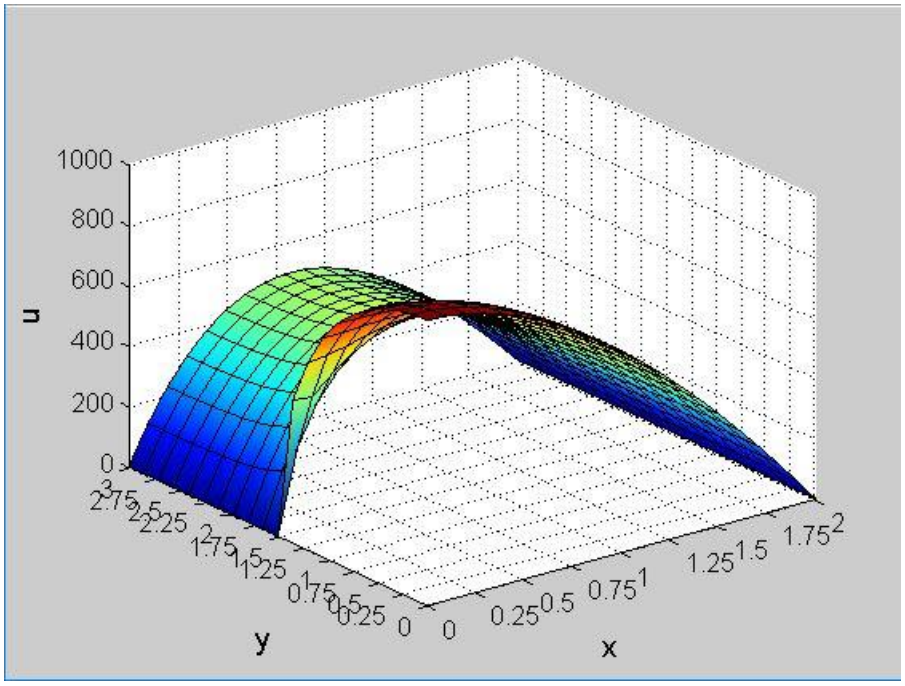

Fig 9: Linear element

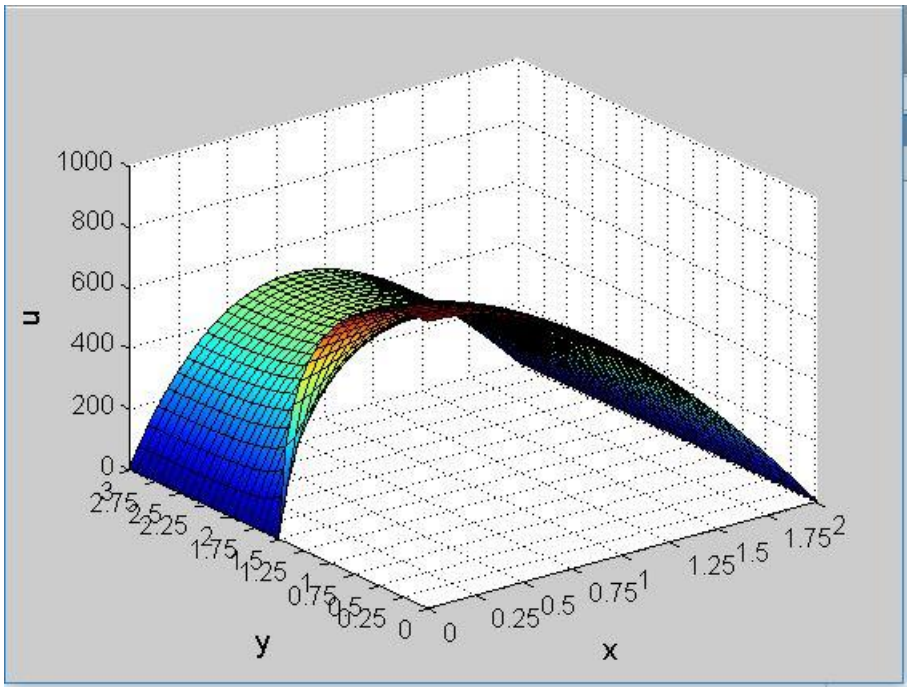

Fig 10: Quadratic element

Using SUPG

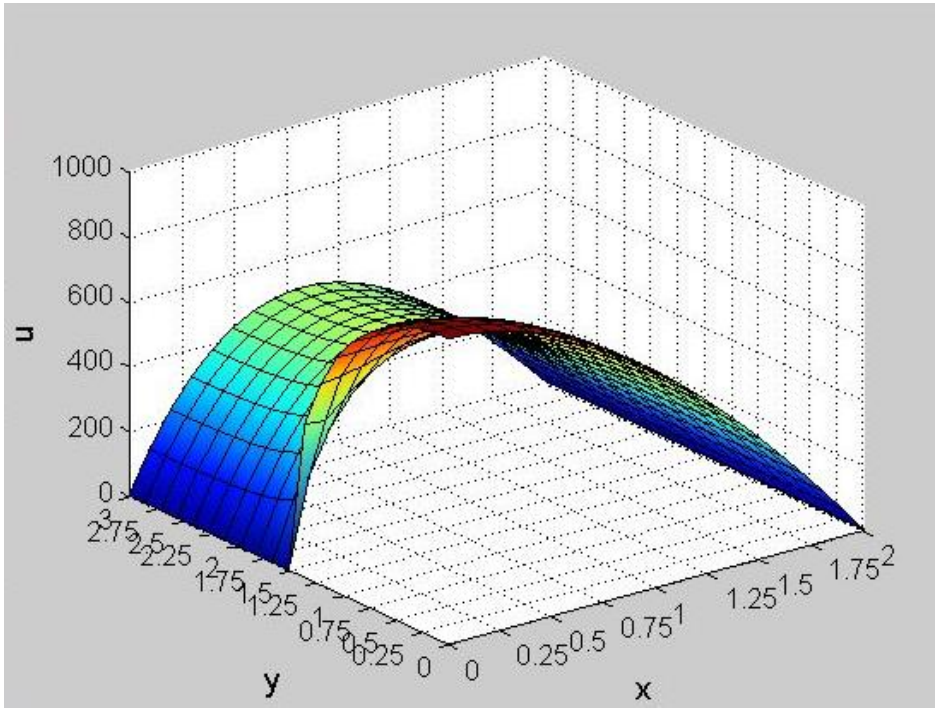

Fig 11: Linear element

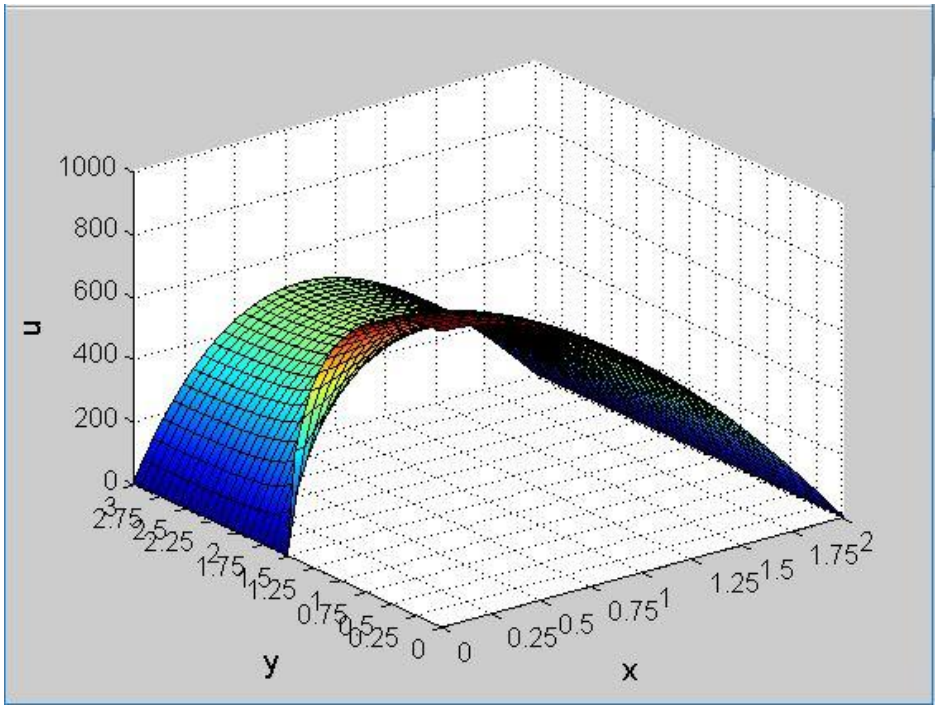

Fig 12: Quadratic element

For the third case, all the graphs look different than the other two cases as there is a source term present. Here too it can be seen that the quadratic has less oscillation than linear although the difference is quite small.

APPENDIX

For linear

```
% Number of elements and number of nodes in the mesh
nElem = size(T,1);
nPt = size(X,1);

K = zeros(nPt,nPt);
f = zeros(nPt,1);

if method == 0
     % GLS
    Pe = a*h/(2*nu);
    tau_p = h*(1 + 9/Pe^2)^(-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 == 1
     % SUPG
    Pe = a*h/(2*nu);
    tau_p = h*(1 + 9/Pe^2)^(-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

else
    error ('Unavailable method')
end
```

```
 % 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
      % GLS
      Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + sig*(N_ig'*N_ig)+ ...
            tau*(ax*Nx+ay*Ny+sig*N_ig)'*(ax*Nx+ay*Ny+sig*N_ig))*dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux,src);
        fe = fe + (N_ig+tau*(ax*Nx+ay*Ny+sig*N_ig))'*(f_ig*dvolu);
    else
        % SUPG
        Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + sig*(N_ig'*N_ig)+ ...
            tau*(ax*Nx+ay*Ny)'*(ax*Nx+ay*Ny+sig*N_ig))*dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux,src);
        fe = fe + (N_ig+tau*(ax*Nx+ay*Ny))'*(f_ig*dvolu);
    end
end
```

For Quadratic

```
 Loop on Gauss points (computation of integrals on the current element)
or ig = 1:ngaus
   N_ig = N(ig,:);
   Nxi_ig = Nxi(ig,:);
   Neta_ig = Neta(ig,:);
   N2xi_ig = N2xi(ig,:);
   N2eta_ig = N2eta(ig,:);
   N2xieta_ig = N2xieta(ig,:);
   N2etaxi_ig = N2etaxi(ig,:);

   Jacob = [Nxi_ig*(Xe(:,1))   Nxi_ig*(Xe(:,2))
       Neta_ig*(Xe(:,1))   Neta_ig*(Xe(:,2))];

   lap = [(Nxi_ig*Xe(:,1))^2 (Nxi_ig*Xe(:,2))*(Nxi_ig*Xe(:,1)) N2xi_ig*Xe
            (Neta_ig*Xe(:,1))*(Nxi_ig*Xe(:,1)) (Neta_ig*Xe(:,2))*(Nxi_ig*)
            0 0 Nxi_ig*Xe(:,1) 0 0 Nxi_ig*(Xe(:,2))
            (Nxi_ig*Xe(:,1))*(Neta_ig*Xe(:,1)) (Nxi_ig*Xe(:,2))*(Neta_ig*)
            (Neta_ig*Xe(:,1))^2 (Neta_ig*Xe(:,2))*(Neta_ig*Xe(:,1)) N2eta_
            0 0 Neta_ig*Xe(:,1) 0 0 Neta_ig*(Xe(:,2))];


   dvolu = wgp(ig)*det(Jacob);
   res   = lap\[N2xi_ig;N2xieta_ig;Nxi_ig;N2etaxi_ig;N2eta_ig;Neta_ig];
   Nx = res(3,:);
   Ny = res(6,:);
   Nxx = res(1,:);
   Nyy = res(5,:);
   fin = Nxx+Nyy;
```

```
xi = z(:,1); eta = z(:,2);
if elem == 0
    if p == 1
        N    = [(1-xi).*(1-eta)/4, (1+xi).*(1-eta)/4, (1+xi).*(1+eta)/4, (1
        Nxi  = [(eta-1)/4, (1-eta)/4, (1+eta)/4, -(1+eta)/4];
        Neta = [(xi-1)/4, -(1+xi)/4,   (1+xi)/4,  (1-xi)/4 ];
        N2xi    = [zeros(size(xi)), zeros(size(xi)), zeros(size(xi)), zeros
        N2eta   = [zeros(size(eta)), zeros(size(eta)), zeros(size(eta)), ze
        N2xieta = [1/4*ones(size(xi)), -1/4*ones(size(xi)), 1/4*ones(size(x
        N2etaxi = [1/4*ones(size(xi)), -1/4*ones(size(xi)), 1/4*ones(size(x
    elseif p == 2
        N    = [xi.*(xi-1).*eta.*(eta-1)/4, xi.*(xi+1).*eta.*(eta-1)/4, ...
            xi.*(xi+1).*eta.*(eta+1)/4, xi.*(xi-1).*eta.*(eta+1)/4, ...
            (1-xi.^2).*eta.*(eta-1)/2,  xi.*(xi+1).*(1-eta.^2)/2,   ...
            (1-xi.^2).*eta.*(eta+1)/2,  xi.*(xi-1).*(1-eta.^2)/2,   ...
            (1-xi.^2).*(1-eta.^2)];
        Nxi  = [(xi-1/2).*eta.*(eta-1)/2,   (xi+1/2).*eta.*(eta-1)/2, ...
            (xi+1/2).*eta.*(eta+1)/2,   (xi-1/2).*eta.*(eta+1)/2, ...
            -xi.*eta.*(eta-1),          (xi+1/2).*(1-eta.^2),   ...
            -xi.*eta.*(eta+1),          (xi-1/2).*(1-eta.^2),   ...
            -2*xi.*(1-eta.^2)];
        Neta = [xi.*(xi-1).*(eta-1/2)/2,    xi.*(xi+1).*(eta-1/2)/2, ...
            xi.*(xi+1).*(eta+1/2)/2,    xi.*(xi-1).*(eta+1/2)/2, ...
            (1-xi.^2).*(eta-1/2),       xi.*(xi+1).*(-eta),   ...
            (1-xi.^2).*(eta+1/2),       xi.*(xi-1).*(-eta),   ...
            (1-xi.^2).*(-2*eta)];
        N2xi    = [eta.*(eta-1)/2,   eta.*(eta-1)/2, ...
                   eta.*(eta+1)/2,   eta.*(eta+1)/2, ...
                   -eta.*(eta-1),         (1-eta.^2),   ...
                   -eta.*(eta+1),        (1-eta.^2),   ...
                   -2*(1-eta.^2)];
```

```
        N2eta   = [xi.*(xi-1).*1/2,     xi.*(xi+1)*1/2, ...
                    xi.*(xi+1).*1/2,      xi.*(xi-1)*1/2, ...
                   (1-xi.^2),       -xi.*(xi+1),    ...
                    (1-xi.^2),       -xi.*(xi-1),    ...
                   -(1-xi.^2)*2];

        N2xieta = [(xi-1/2).*(2*eta-1)/2,    (xi+1/2).*(2*eta-1)/2, ...
                   (xi+1/2).*(2*eta+1)/2,    (xi-1/2).*(2*eta+1)/2, ...
                   -xi.*(2*eta-1),           (xi+1/2).*(-2*eta),    ...
                   -xi.*(2*eta+1),           (xi-1/2).*(-2*eta),    ...
                    4*xi.*eta];

        N2etaxi = [(2*xi-1).*(eta-1/2)/2,    (2*xi+1).*(eta-1/2)/2, ...
                   (2*xi+1).*(eta+1/2)/2,    (2*xi-1).*(eta+1/2)/2, ...
                   -2*xi.*(eta-1/2),         (2*xi+1).*(-eta),    ...
                   -2*xi.*(eta+1/2),         (2*xi-1).*(-eta),    ...
                    4*xi.*eta];
    else
        error('not available interpolation degree')
    end
elseif elem == 1
    if p == 1
        N = [1-xi-eta, xi, eta];
        Nxi = [-ones(size(xi)), ones(size(xi)), zeros(size(xi))];
        Neta = [-ones(size(xi)), zeros(size(xi)), ones(size(xi))];
        N2xi  = [zeros(size(xi)), zeros(size(xi)), zeros(size(xi))];
        N2eta = [zeros(size(eta)), zeros(size(eta)), zeros(size(eta))];
        N2xieta = [zeros(size(xi)), zeros(size(xi)), zeros(size(xi))];
        N2etaxi = [zeros(size(eta)), zeros(size(eta)), zeros(size(eta))];
    else
        error('not available interpolation degree')
    end
```

```
    if method == 0
      % GLS
      Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + sig*(N_ig'*N_ig)+ ...
            tau*(ax*Nx+ay*Ny-nu*fin+sig*N_ig)'*(ax*Nx+ay*Ny-nu*fin+sig*N_ig))*dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux,src);
        fe = fe + (N_ig+tau*(ax*Nx+ay*Ny-nu*fin+sig*N_ig))'*(f_ig*dvolu);
    else
        % SUPG
        Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + sig*(N_ig'*N_ig)+ ...
            tau*(ax*Nx+ay*Ny)'*(ax*Nx+ay*Ny-nu*fin+sig*N_ig))*dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux,src);
        fe = fe + (N_ig+tau*(ax*Nx+ay*Ny))'*(f_ig*dvolu);
    end
nd
```