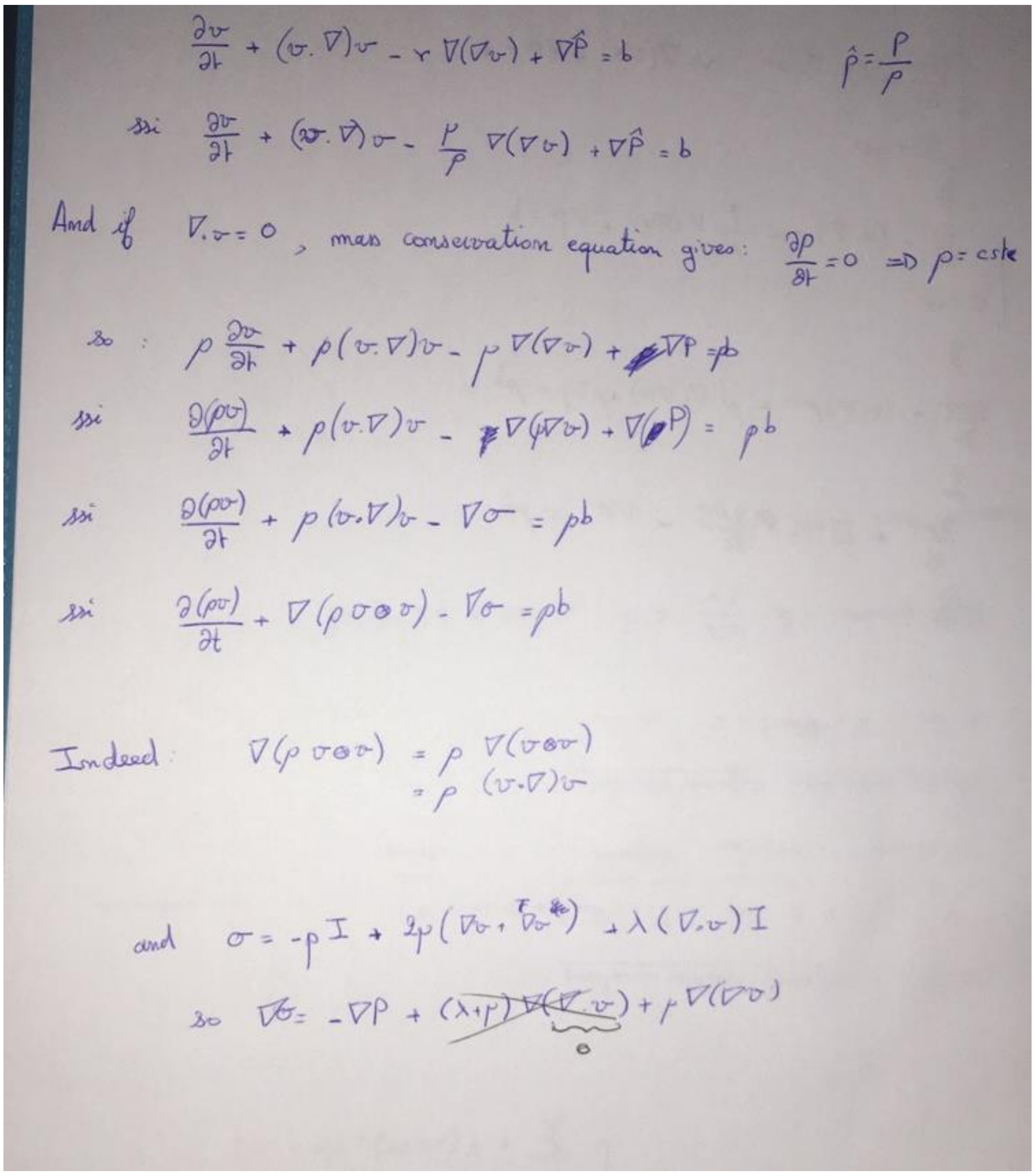

$$\frac{\partial v}{\partial t} + (v.\nabla)v - \nu \nabla(\nabla v) + \nabla \hat{P} = b \qquad \hat{P} = \frac{P}{\rho}$$

ssi $\frac{\partial v}{\partial t} + (v.\nabla)v - \frac{\mu}{\rho} \nabla(\nabla v) + \nabla \hat{P} = b$

And if $\nabla . v = 0$, mass conservation equation gives: $\frac{\partial \rho}{\partial t} = 0 \Rightarrow \rho = cste$

so : $\rho \frac{\partial v}{\partial t} + \rho(v.\nabla)v - \mu \nabla(\nabla v) + \cancel{\rho}\nabla P = \rho b$

ssi $\frac{\partial(\rho v)}{\partial t} + \rho(v.\nabla)v - \cancel{\mu}\nabla(\mu\nabla v) + \nabla(\cancel{\rho}P) = \rho b$

ssi $\frac{\partial(\rho v)}{\partial t} + \rho(v.\nabla)v - \nabla\sigma = \rho b$

ssi $\frac{\partial(\rho v)}{\partial t} + \nabla(\rho v \otimes v) - \nabla\sigma = \rho b$

Indeed: $\nabla(\rho v \otimes v) = \rho \nabla(v \otimes v)$
$= \rho (v.\nabla)v$

and $\sigma = -pI + 2\mu(\nabla v + \nabla v^{T}) + \lambda(\nabla . v)I$

so $\nabla\sigma = -\nabla P + \cancel{(\lambda+\mu)\nabla(\underbrace{\nabla . v}_{0})} + \mu\nabla(\nabla v)$

I have unfortunatly not been able to do the Matlab part as I don't have it on my computer and I still don't have Id for UPC's computers. I'll just screen shot the Matlab code I've tried to do.

It should be working wright as I took inspiration from the SU method and used the formula from the presentation to compute SUPG and GLS methods. I've just not been able to run it to see if it works and to see the influence of the source term.

Code for GLS method

```
function [K,f] = GLS_system(X,T,referenceElement,example)
% [K,f] = GLS_system(X,T,referenceElement,example)
% System obtained by discretizing the weak form associated to
% the convection-diffusion equation
%               a ux - nu uxx = f
% with gthe stabilized SU method.
% Boundary conditions are not considered.


% reference element information
nen = referenceElement.nen;
ngaus = referenceElement.ngaus;
```

```
wgp = referenceElement.GaussWeigths;
N = referenceElement.N;
Nxi = referenceElement.Nxi;

% example properties
a = example.a;
nu = example.nu;
tau = example.tau;

% Number of nodes and elements
nPt = length(X);
nElem = size(T,1);

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


if p == 1
% Loop on elements
for ielem = 1:nElem
    Te = T(ielem,:);
    Xe = X(Te);
    h = Xe(end) - Xe(1);

    Ke = zeros(nen);
    fe = zeros(nen,1);
    % Loop on Gauss points
    for ig = 1:ngaus
        N_ig = N(ig,:);
        Nx_ig = Nxi(ig,:)*2/h;
        w_ig = wgp(ig)*h/2;
        Ke = Ke + w_ig*(N_ig'*a*Nx_ig + Nx_ig'*nu*Nx_ig + N_ig'*sigm*N_ig) +
(w_ig*a*Nx_ig' + sigm*N_ig')*tau*(a*Nx_ig + sigm*N_ig );

        x = N_ig*Xe; % x-coordinate of the gauss point
        s = SourceTerm(x,example);
        fe = fe + w_ig*(N_ig)'*s + w_ig*(a*Nx_ig + sigm*N_ig)'*tau*s ;
    end
    % Assmebly
    K(Te,Te) = K(Te,Te) + Ke;
    f(Te) = f(Te) + fe;
end
end

if p == 2
% Loop on elements
for ielem = 1:nElem
    Te = T(ielem,:);
    Xe = X(2*Te-1);
    h = Xe(end) - Xe(1);

    Ke = zeros(nen);
    fe = zeros(nen,1);
    % Loop on Gauss points
    for ig = 1:ngaus
        N_ig = N(ig,:);
        Nx_ig = Nxi(ig,:)*2/h;
          N2x_ig = N2xi(ig,:)*1/h;
        w_ig = wgp(ig)*h/2;
        Ke = Ke + w_ig*(N_ig'*a*Nx_ig + Nx_ig'*nu*Nx_ig + N_ig'*sigm*N_ig) +
(w_ig*a*Nx_ig' - w_ig*nu*N2x_ig' + sigm*N_ig')*tau*(a*Nx_ig - nu*N2x_ig + sigm*N_ig
);

        x = N_ig*Xe; % x-coordinate of the gauss point
        s = SourceTerm(x,example);
        fe = fe + w_ig*(N_ig)'*s + w_ig*(a*Nx_ig - nu*N2x_ig + sigm*N_ig)'*tau*s ;

    end
    % Assmebly
    K(Te,Te) = K(Te,Te) + Ke;
    f(Te) = f(Te) + fe;
end
end
```

Code for SUPG method

```
function [K,f] = SUPG_system(X,T,referenceElement,example)
% [K,f] = SUPG_system(X,T,referenceElement,example)
% System obtained by discretizing the weak form associated to
% the convection-diffusion equation
%               a ux - nu uxx = f
% with gthe stabilized SU method.
% Boundary conditions are not considered.


% reference element information
nen = referenceElement.nen;
ngaus = referenceElement.ngaus;
wgp = referenceElement.GaussWeigths;
N = referenceElement.N;
Nxi = referenceElement.Nxi;
N2xi = Element.N2xi;

% example properties
a = example.a;
nu = example.nu;
tau = example.tau;
sigm = example.sigm

% Number of nodes and elements
nPt = length(X);
nElem = size(T,1);

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

if p == 1
% Loop on elements
for ielem = 1:nElem
    Te = T(ielem,:);
    Xe = X(Te);
    h = Xe(end) - Xe(1);

    Ke = zeros(nen);
    fe = zeros(nen,1);
    % Loop on Gauss points
    for ig = 1:ngaus
        N_ig = N(ig,:);
        Nx_ig = Nxi(ig,:)*2/h;
          N2x_ig = N2xi(ig,:)*1/h;
        w_ig = wgp(ig)*h/2;
        Ke = Ke + w_ig*(N_ig'*a*Nx_ig + Nx_ig'*nu*Nx_ig + N_ig'*sigm) +
w_ig*(tau*a*Nx_ig)'*(a*Nx_ig + sigm*N_ig );

        x = N_ig*Xe; % x-coordinate of the gauss point
        s = SourceTerm(x,example);
        fe = fe + w_ig*(N_ig)'*s + w_ig*(tau*a*Nx_ig)'*s ;
    end
    % Assmebly
    K(Te,Te) = K(Te,Te) + Ke;
    f(Te) = f(Te) + fe;
end
end

if p == 2
```

```
% Loop on elements
for ielem = 1:nElem
    Te = T(ielem,:);
    Xe = X(2*Te-1);
    h = Xe(end) - Xe(1);

    Ke = zeros(nen);
    fe = zeros(nen,1);
    % Loop on Gauss points
    for ig = 1:ngaus
        N_ig = N(ig,:);
        Nx_ig = Nxi(ig,:)*2/h;
          N2x_ig = N2xi(ig,:)*1/h;
        w_ig = wgp(ig)*h/2;
        Ke = Ke + w_ig*(N_ig'*a*Nx_ig + Nx_ig'*nu*Nx_ig + N_ig'*sigm) +
w_ig*(tau*a*Nx_ig)'*(a*Nx_ig - nu*N2x_ig + sigm*N_ig );

        x = N_ig*Xe; % x-coordinate of the gauss point
        s = SourceTerm(x,example);
        fe = fe + w_ig*(N_ig)'*s + w_ig*(tau*a*Nx_ig)'*s ;
    end
    % Assmebly
    K(Te,Te) = K(Te,Te) + Ke;
    f(Te) = f(Te) + fe;
end
end
```