Name	Ahmed Saeed Mohamed Sherif
Course	Finite Elements in Fluids
Master	MSc Computational Mechanics
Report no.	1
Topic	1D Steady Transport Equation - Stabilized FE (SU, SUPG, GLS, SGS)

1D convection-diffusion equation with constant coefficients and Dirichlet boundary conditions:

$$\begin{cases} a u_x - \nu u_{xx} = s \quad x \in [0,1] \\ u(0) = u_0 \,;\, u(1) = u_1 \end{cases}$$

Problem no. 1 involves ⟶ $s = 0,\ u_0 = 0,\ u_1 = 1$

Argument 1: Using Galerkin formulation, four cases were tested:

Case 1: $a = 1, \nu = 0.2$, 10 linear elements
Case 2: $a = 20, \nu = 0.2$, 10 linear elements
Case 3: $a = 1, \nu = 0.01$, 10 linear elements
Case 4: $a = 1, \nu = 0.01$, 50 linear elements

A comparison between the solution obtained for each case is shown in *Figure 1*. It is clearly seen that for Peclet number, Pe, greater than 1 the solution is unstable and node-to-node oscillations arises. This is due to the domination of the convection part whose stiffness matrix is non-symmetric. Thus, a stabilization technique is needed to obtain a stable oscillation-free solution.

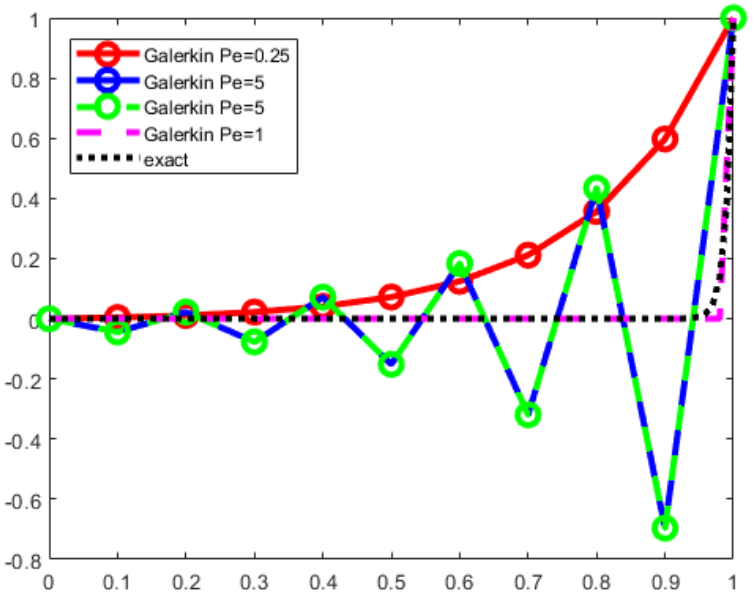

Figure 1 Solution of problem 1 obtained using Galerkin linear FE formulation for 4 cases with different Peclet number

Argument 2: Case 3 where $Pe = 5$ is solved using four different stabilized methods which are Streamline Upwind (SU), Streamline Upwind Petrov-Galerkin (SUPG), Galerkin Least-squares (GLS) and Sub-Grid Scale (SGS). For this, the optimal stabilization parameter, $\tau = h/2a$ (coth Pe - $1/Pe$), is used. The solutions obtained using linear elements are shown in *Figure 2*. It is noted that the nodal values are exact in all cases which makes all the approximations identical. The reason is the use of the optimal stabilization parameter as well as having a constant source term.

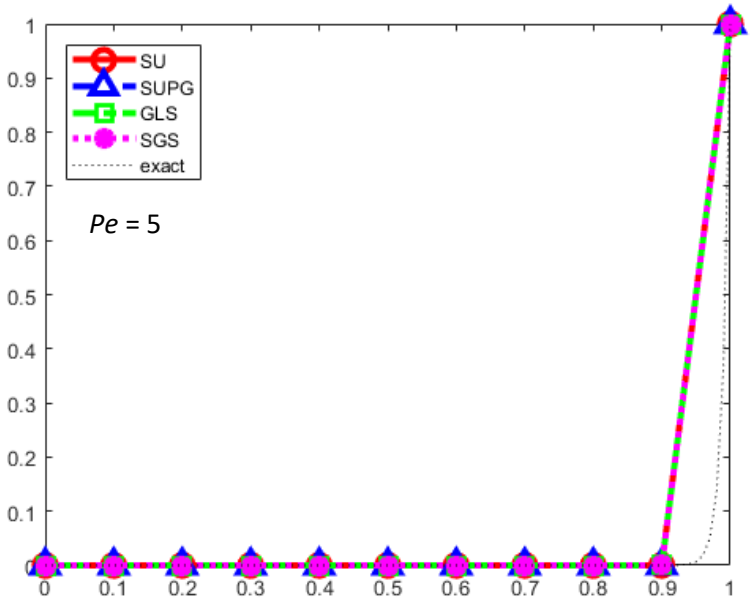

Figure 2 Solution of problem 1 - case 3 obtained using four different stabilized methods with linear elements

Argument 3: The effect of the stabilization parameter is investigated. Again, case 3 where $Pe = 5$ is solved using two different values of the stabilization parameter, $\tau = 1$ and $\tau = 0.01$. The results are shown in *Figure 3*. It is observed that for large value of τ the solution becomes very smooth, while for very small value of τ the solution is unstable. This could be interpreted if we recall that the parameter τ is directly proportional to the amount of added diffusion. Thus, extra added diffusion leads to a very smooth stable but inaccurate solution, while very small added diffusion is not enough to stabilize the convection dominated problem. Therefore, the value of the stabilization parameter should be chosen carefully to obtain a stable and accurate solution. For this 1D case with linear elements, the optimal value of τ exists and this would be the best choice.

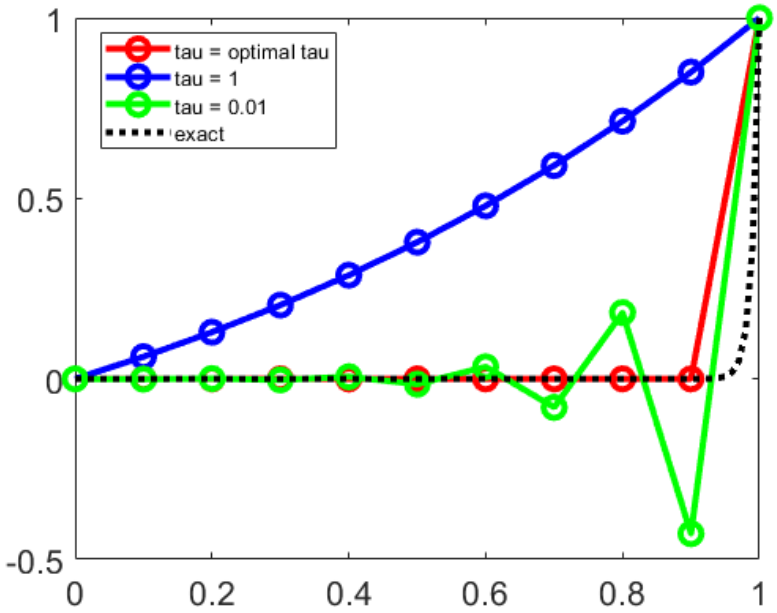

Figure 3 Solution of problem 1 - case 3 obtained using SUPG with different values of τ

Argument 4: If quadratic elements are used, we wouldn't obtain nodally exact solution if the previous optimal value of the stabilization parameter for linear elements is used, see *Figure 4* where the solution obtained using SUPG with 10 quadratic elements are used. However, to obtain nodally exact solution, the stabilization parameter should be further adjusted (Donea and Huerta, 2004).

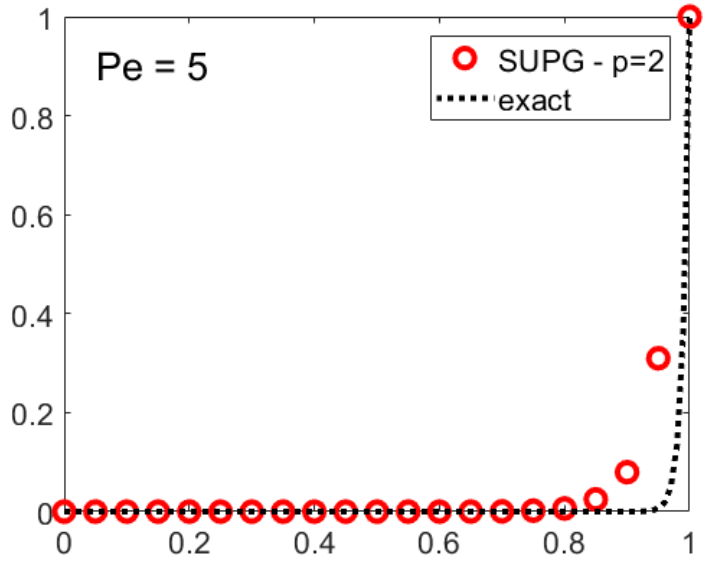

Figure 4 The nodal solution of problem 1 - case 3 obtained using SUPG with 10 quadratic elements (τ=optimal value of linear elements)

Argument 5: The problem is changed to Problem no. 3 which involves ⟶ $s = \sin(\pi x),\ u_0 = 0,\ u_1 = 1$

The source term is now non-zero and non-constant.

Repeating the previous arguments, it is again observed that for Galerkin formulation, Peclet number must be less than or equal to 1 to obtain a stable solution. See *Figure 5*.

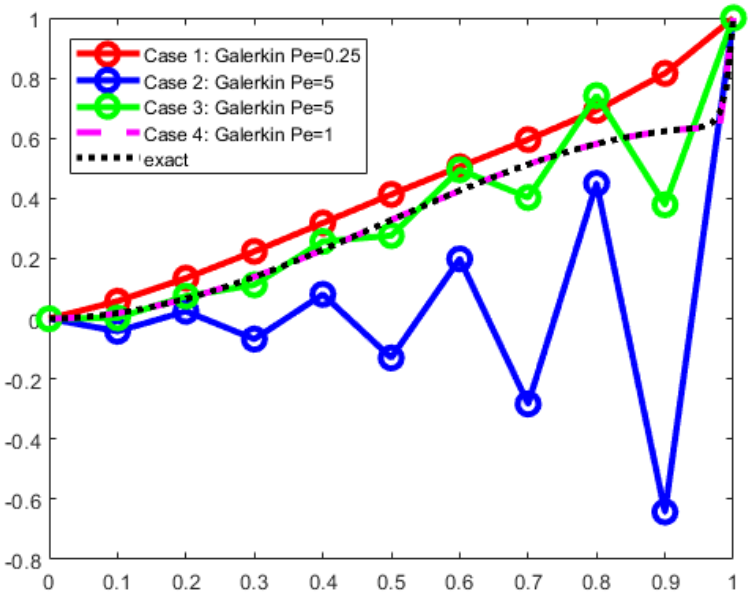

Figure 5 Solution of problem 3 obtained using Galerkin linear FE formulation for 4 cases with different Peclet number

Using the four stabilized methods mentioned earlier (SU, SUPG, GLS and SGS) for case 3 where $Pe = 5$, the results are shown in *Figure 6*. The only difference appears in the non-consistent case of SU method, where the approximated solution is far from the exact one, while the other consistent methods produce accurate solutions which are nodally exact. The reason is that SU method doesn't perform well for non-constant source terms even if the optimal stabilization parameter is used.

Moreover, the effect of the stabilization parameter is again proved to be crucial to achieve stable and yet accurate solution. *Figure 7* shows the solution of case 3 obtained using SUPG with linear elements for two different values of the stabilization parameter, $\tau = 1$ and $\tau = 0.01$. Again, large value of τ yields a very smooth stable but inaccurate solution while very small value of τ results in unstable solution. Thus, the stabilization parameter must be chosen carefully. Furthermore, the optimal value of τ yields nodally exact solution.

Again, the need for a modified optimal stabilization parameter for quadratic elements is shown, where in *Figure 8* the solution obtained using SUPG with 10 quadratic elements is shown for problem 3 - case 3. The nodal solution is observed to be inexact.

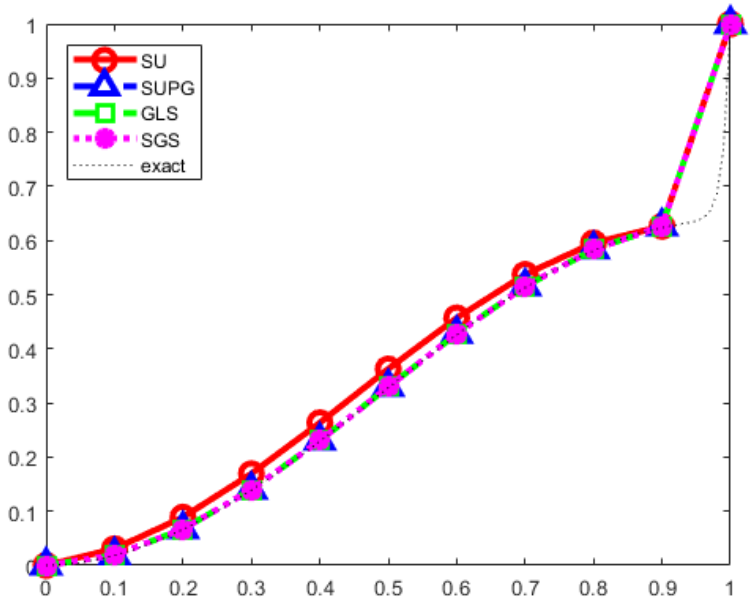

Figure 6 Solution of problem 3 - case 3 obtained using four different stabilized methods with linear elements

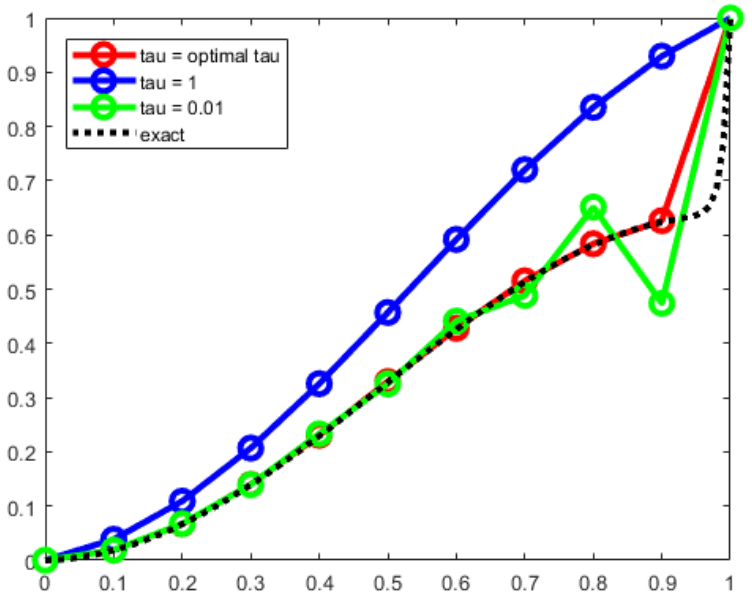

Figure 7 Solution of problem 3 - case 3 obtained using SUPG with different values of τ

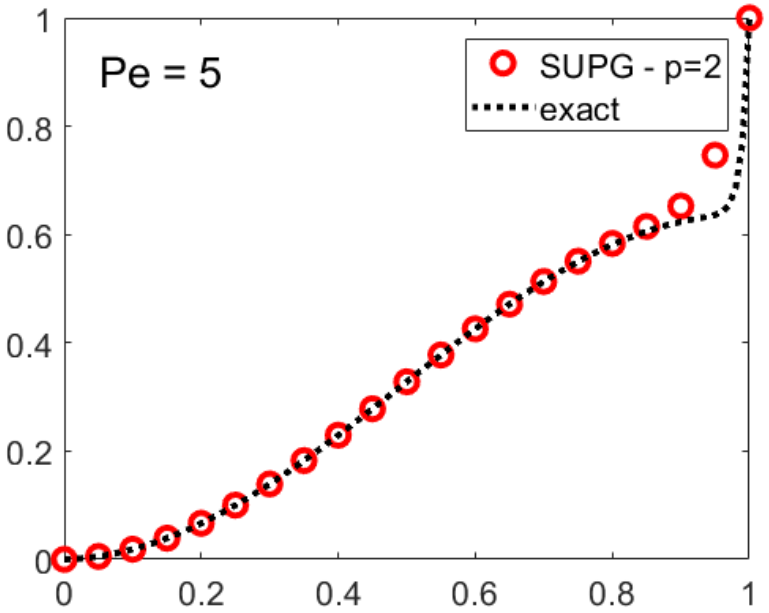

Figure 8 The nodal solution of problem 3 - case 3 obtained using SUPG with 10 quadratic elements (τ=optimal value of linear elements

Argument 6:

Consider a 1D steady convection-diffusion-reaction problem

$$au_x - \nu u_{xx} + \sigma u = s \qquad x \in (0,1)$$
$$u(0) = 0 \qquad u(1) = 1.$$

Consider $a = 1,\ \nu = 10^{-2},\ \sigma = 20,\ s = 0$

And $\tau = \left((\frac{2a}{h})^2 + 9(\frac{4\nu}{h^2})^2 + \sigma^2\right)^{-1/2} = \frac{h}{2a}\left(1 + \frac{9}{Pe^2} + (\frac{h}{2a}\sigma)^2\right)^{-1/2}$

For this problem, the exact solution is easily found by solving a second order ordinary differential equation with constant coefficients. It is obtained to be used as a reference for the numerical results.

The solution obtained using Galerkin, SUPG, GLS and SGS methods in a uniform mesh of 10 linear elements is shown in *Figure 9*. In this problem, the Peclet number is 5, it is observed that an oscillation-free solution is only obtained in cases of SGS method. It is also observed that the oscillations in case of GLS is more compared to SUPG. This is happening because of using linear elements for a problem with positive reaction coefficient. It can be further understood if we recall the GLS test function

GLS test function: $\mathcal{P}(w) = \mathcal{L}(w) = \underbrace{\boldsymbol{a}\cdot\boldsymbol{\nabla} w}_{\text{SUPG}} - \underbrace{\boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla} w)}_{0} + \underbrace{\sigma w}_{\text{Galerkin}}$

It can be seen that for linear elements the second term is zero, and for positive values of the reaction coefficient σ, GLS is SUPG with the Galerkin term weighted $1 + \sigma\tau$ times more. This issue is solved in case of SGS method where the test SGS test function is given by:

SGS test function: $\mathcal{P}(w) = -\mathcal{L}^*(w) = \boldsymbol{a} \cdot \nabla w + \nabla \cdot (\nu \nabla w) - \sigma w$

Thus, in this case the Galerkin term is weighted by $1 - \sigma\tau$ and thus SGS has no oscillations [2].

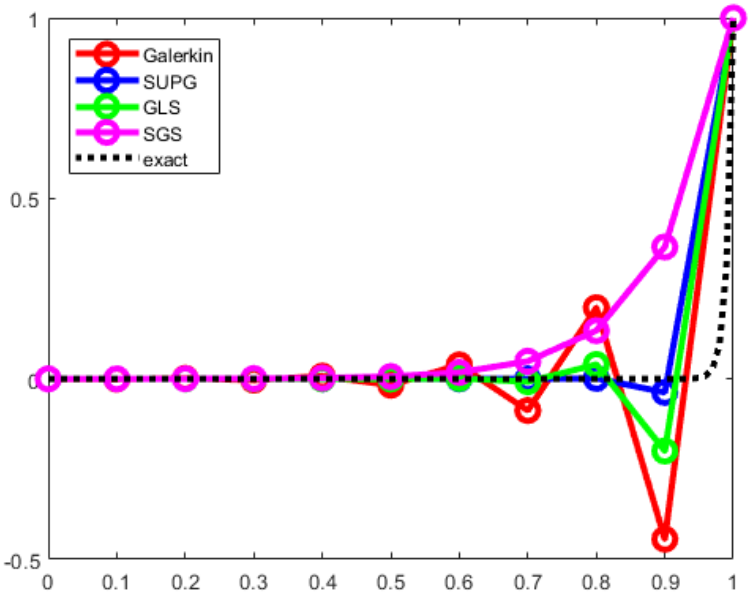

Figure 9 The solution of the steady convection-diffusion-reaction equation obtained using different stabilized methods

Argument 7: Using Galerkin formulation with a refined mesh near the sharp front yields a stable and accurate solution. Figure 10 shows the solution obtained using 20 linear elements, where the element size is 0.1 in the region [0,0.9] then smaller elements of size 0.01 is used in the region [0.9,1].

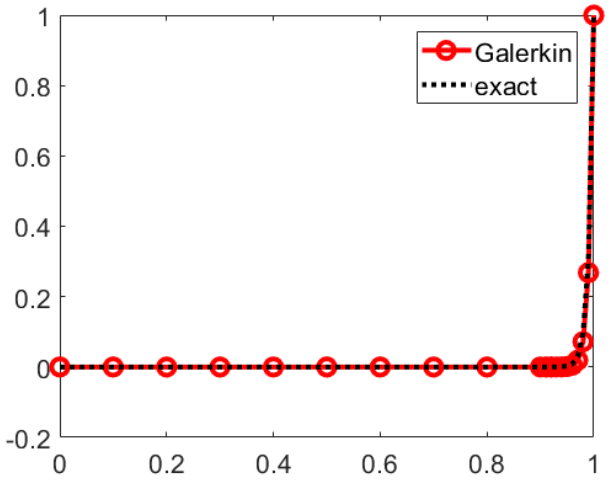

Figure 10 Solution of the steady convection-diffusion-reaction equation obtained using Galerkin FE with a refined mesh near the sharp front

References:

[1] Donea, J. and Huerta, A. (2004). *Finite element methods for flow problems*. Chichester: Wiley, pp.55-57.

[2] Donea, J. and Huerta, A. (2004). *Finite element methods for flow problems*. Chichester: Wiley, pp.63-64.

Appendix 1: Developed Parts of the code for steady convection-diffusion problem

SUPG_System

```
% Loop on Gauss points
for ig = 1:ngaus
    N_ig = N(ig,:);
    Nx_ig = Nxi(ig,:)*2/h;
    N2x_ig = N2xi(ig,:)*(2/h)^2;
    w_ig = wgp(ig)*h/2;
    Ke = Ke + w_ig*(N_ig'*(a*Nx_ig) + Nx_ig'*(nu*Nx_ig))...
        + w_ig*tau*(a*Nx_ig)'*(a*Nx_ig-nu*N2x_ig);
    x = N_ig*Xe; % x-coordinate of the gauss point
    s = SourceTerm(x,example);
    fe = fe + w_ig*(N_ig'+tau*a*Nx_ig')*s;
end
```

GLS_System

```
% Loop on Gauss points
for ig = 1:ngaus
    N_ig = N(ig,:);
    Nx_ig = Nxi(ig,:)*2/h;
    N2x_ig = N2xi(ig,:)*(2/h)^2;
    w_ig = wgp(ig)*h/2;
    Ke = Ke + w_ig*(N_ig'*(a*Nx_ig) + Nx_ig'*(nu*Nx_ig))...
        + w_ig*tau*(a*Nx_ig-nu*N2x_ig)'*(a*Nx_ig-nu*N2x_ig);
    x = N_ig*Xe; % x-coordinate of the gauss point
    s = SourceTerm(x,example);
    fe = fe + w_ig*(N_ig'+tau*(a*Nx_ig'-nu*N2x_ig'))*s;
end
```

SGS_System

```
% Loop on Gauss points
for ig = 1:ngaus
    N_ig = N(ig,:);
    Nx_ig = Nxi(ig,:)*2/h;
    N2x_ig = N2xi(ig,:)*(2/h)^2;
    w_ig = wgp(ig)*h/2;
    Ke = Ke + w_ig*(N_ig'*(a*Nx_ig) + Nx_ig'*(nu*Nx_ig))...
        + w_ig*tau*(a*Nx_ig+nu*N2x_ig)'*(a*Nx_ig-nu*N2x_ig);
    x = N_ig*Xe; % x-coordinate of the gauss point
    s = SourceTerm(x,example);
    fe = fe + w_ig*(N_ig'+tau*(a*Nx_ig'+nu*N2x_ig'))*s;
end
```

Appendix 2: Developed Parts of the code for steady convection-diffusion-reaction problem

Galerkin_System

```
    % 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'*sigma*N_ig);
        x = N_ig*Xe; % x-coordinate of the gauss point
        s = SourceTerm(x,example);
        fe = fe + w_ig*(N_ig')*s;
    end
```

SUPG_System

```
    % Loop on Gauss points
    for ig = 1:ngaus
        N_ig = N(ig,:);
        Nx_ig = Nxi(ig,:)*2/h;
        N2x_ig = N2xi(ig,:)*(2/h)^2;
        w_ig = wgp(ig)*h/2;
        Ke = Ke + w_ig*(N_ig'*(a*Nx_ig) + Nx_ig'*(nu*Nx_ig) +
N_ig'*sigma*N_ig)...
            + w_ig*tau*(a*Nx_ig)'*(a*Nx_ig - nu*N2x_ig + sigma*N_ig);
        x = N_ig*Xe; % x-coordinate of the gauss point
        s = SourceTerm(x,example);
        fe = fe + w_ig*(N_ig' + tau*a*Nx_ig')*s;
    end
```

GLS_System

```
    % Loop on Gauss points
    for ig = 1:ngaus
        N_ig = N(ig,:);
        Nx_ig = Nxi(ig,:)*2/h;
        N2x_ig = N2xi(ig,:)*(2/h)^2;
        w_ig = wgp(ig)*h/2;
        Ke = Ke + w_ig*(N_ig'*(a*Nx_ig) + Nx_ig'*(nu*Nx_ig) +
N_ig'*sigma*N_ig)...
            + w_ig*tau*(a*Nx_ig - nu*N2x_ig + sigma*N_ig)'*(a*Nx_ig -
nu*N2x_ig + sigma*N_ig);
        x = N_ig*Xe; % x-coordinate of the gauss point
        s = SourceTerm(x,example);
        fe = fe + w_ig*(N_ig'+tau*(a*Nx_ig' - nu*N2x_ig' +
sigma*N_ig'))*s;
    end
```

SGS_System

```
    % Loop on Gauss points
    for ig = 1:ngaus
        N_ig = N(ig,:);
        Nx_ig = Nxi(ig,:)*2/h;
        N2x_ig = N2xi(ig,:)*(2/h)^2;
        w_ig = wgp(ig)*h/2;
        Ke = Ke + w_ig*(N_ig'*(a*Nx_ig) + Nx_ig'*(nu*Nx_ig) +
N_ig'*sigma*N_ig)...
            + w_ig*tau*(a*Nx_ig + nu*N2x_ig - sigma*N_ig)'*(a*Nx_ig -
nu*N2x_ig + sigma*N_ig);
        x = N_ig*Xe; % x-coordinate of the gauss point
        s = SourceTerm(x,example);
        fe = fe + w_ig*(N_ig'+tau*(a*Nx_ig' + nu*N2x_ig' -
sigma*N_ig'))*s;
    end
```

Exact solution for the problem

```
if problem == 1 % already modified
    r1 = (a+sqrt(a^2 + 4*nu*sigma))/(2*nu);
    r2 = (a-sqrt(a^2 + 4*nu*sigma))/(2*nu);
    c1 = 1.418784356*10^-51;
    c2 = -c1;
    res = c1*exp(r1*x)+c2*exp(r2*x);
%     res = (1-exp(x*a/nu))/(1-exp(a/nu));
% elseif problem == 2
%     res = (x + (1 - exp(a/nu*x))/((exp(a/nu)-1)))/a;
% elseif problem == 3
%     aux = pi*(a^2+nu^2*pi^2);
%     e = exp(a/nu);
%     c1 = (-aux+a*(e+1))/(aux*(e-1));
%     c2 = (aux-2*a)/(aux*(e-1));
%     res = c1 + c2*exp(a*x/nu) + nu*pi*(sin(pi*x)-
a*cos(pi*x)/(nu*pi))/aux;
end
```

A part added to account for quadratic elements

```
if p == 1
nPt = nElem + 1;
h = (dom(2) - dom(1))/nElem;
X = (dom(1):h:dom(2))';
T = [1:nPt-1; 2:nPt]';
elseif p == 2
nPt = 2*nElem + 1;
h = (dom(2) - dom(1))/nElem;
X = (dom(1):h/2:dom(2))';
T = [1:2:nPt-2; 2:2:nPt-1; 3:2:nPt]';
end
```