Finite Elements in Fluids

Homework

Steday convection-diffusion problem

Author: Cristina García Albela
MS in Computational Mechanics

1. Introduction

Different methods are used to study the steady convection – diffusion problem. Following the basic steps of the standard Galerkin finite element method, it can be prove the deficiencies that the classical Galerkin approach has when the problem is convection dominated. This is the motivation to introduce other methods designed to produce stable and accurate results when convective effects are high, as Streamline-Upwind (SU), Streamline-Upwind Petrov-Galerkin method (SUPG) or Galerkin/Least-squares method.

2. Problem statement

A steady convection-diffusion transport problem is going to be solved using different finite elements methods. It is defined by the following equations:

$$\begin{aligned} a\nabla u - \nabla \cdot (\nu \nabla u) &= s \qquad in\ [0,1] \\ u(0) = 0 \ \ u(1) &= 1 \qquad on\ \Gamma_D \end{aligned}$$

where u is the unknown, a is the convective velocity, ν the coefficient of diffusivity and s the volumetric source term. The convective and diffusivity terms with remains constant during all the calculations. Therefore, the source term will change, starting the calculation with no source term, turning to a x-dependent one.

Talking about the kind of element that is going to be used, the first group of calculations will be made using linear elements. Galerkin, SU, SUPG and GLS are going to be implemented using 10 linear elements and the results and characteristic of the solutions will be discussing. Then, quadratic elements will be implement in SU, SUPG and GLS methods in order to compare the obtained results and talk about the advantages or disadvantages that this king of elements can introduced in the problem solution.

3. Description of the methods and Matlab implementation

The fourth methods will be described for linear elements and $\sigma = 0$ together with the way of write then inside a Matlab code. Then, the changes due to the quadratic elements implementation will be highlighted. It is important to highlight the fact that Gauss quadrature and natural coordinates are going to be use inside all the methods in order to define the weight and shape functions. This topic will be explained along the methods too.
As we are not going to work with

a) Galerkin

Matrix elements in Galerkin discretization of the convection – diffusion problem yields the following equation

$$\int_\Omega w(a \cdot \nabla u) d\Omega + \int_\Omega w \cdot (\nu \nabla u) d\Omega = \int_\Omega w s d\Omega$$

Galerkin is not the ideal method to solve convection – diffusion problems. Introducing the Péclet number ($Pe = \frac{ah}{2\nu}$) that characterizes the importance of convective and diffusion effects, it can be prove that Galerkin method works bad for those cases with a larger Pe number, it is said, convection dominated problems, in which ones oscillations will appear.

After the Matlab implementation, the code that represents this equations in each point and element will be like

```
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));
    x = N_ig*Xe;
    s = SourceTerm(x,example);
    fe = fe + w_ig*(N_ig')*s;
end
```

Figure 1. Galerkin code for linear elements

Gauss quadrature and Jacobian

First of all it is going to be explained how the used of Gauss quadrature is reflected in the code, standard that is going to be repeated in all the following cases.

The shape functions are represented by the variables N_ig and Nx_ig, as these case is for linear elements, the second order term is not added because it is going to be zero. These shape functions takes its value at the reference system of coordinates from the Gauss quadrature according with the number of nodes in each case, as well as the weight w_ig that multiplies all the parameters of the K_e in each Gauss node. The point here is $^2/_h$ and $^h/_2$ values which appear multiplying the shape functions in the first lines, that come from the change from reference coordinates (ξ, η) to physical coordinates (x, y).

This change is made using the Jacobian

$$dxdy = |J|d\xi d\eta$$

taking for the 1D case the values

$$J = \frac{h}{2} \qquad J^{-1} = \frac{2}{h}$$

b) SU

The SU method is created with the aim to improve the Galerkin standard method and turn it into a suitable method for any Pe number.

$$\int_\Omega w(a \cdot \nabla u)d\Omega + \int_\Omega w \cdot (\nu \nabla u)d\Omega + \int_\Omega \frac{\bar{\bar{\nu}}}{|a|^2}(a \cdot \nabla w)(a \cdot \nabla u) = \int_\Omega wsd\Omega$$

$$\bar{\nu} = \beta \frac{ah}{2} \qquad \beta = \coth Pe - \frac{1}{Pe} \qquad for\ 1D$$

With the SU diffusion $(\bar{\nu})$ is added in the streamline direction in order to obtain exact nodal solutions. The problem in this kind of methods, is that for larger Pe numbers they produces smooth solutions. Moreover, due to the fact that the solution of the differential equation is no longer the same of the weak form, the method is not consistent.

Linear elements Matlab code

The parameter tau have been added in order to represent the added diffusion.

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

Figure 2. SU code for linear elemtents

Quadratic elements Matlab code

Some general modifications are made to be able to work with quadratic elements. First of all, the definition of the parameter h and the connectivity matrix T (Figure 3) change in order to indicate to de program that now it is going to work with three nodes per element. Moreover, the code suffer some variations in order to introduce the idea that the balancing diffusion that must be added to the nodes is different from the middle to the corner nodes. So that, for quadratic elements two β will be defined.

Finally, the way to plot is going to be able to represent parabolic lines between the nodes. The following modifications are made in the code of **ALL** the methods to work with quadratic elements

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

Figure 3. Code changes in h and T parameters

```
tau = example.tau;
tauc = example.tauc;
Tau = diag([tauc,tau,tauc]);
```

Figure 4. Tau implementation in corners and middle-point

```
function [x0,y0] = plotp2(sol,X)
numnp = 11;
numel = (numnp-1)/2;

x0 =[];
y0 =[];

x = [-1:0.1:1];
N = [((x-1).*x/2)'  (1-x.^2)'  ((x+1).*x/2)'];

for i=1:numel
   isp = [2*i-1 2*i 2*i+1];
   delta = (X(2*i+1)-X(2*i-1))/2;
   x0 = [x0,(X(2*i)+x*delta)];
   y = N*sol(isp);
   y0 = [y0,y'];
end
end
```

Figure 5. Plot modifications in quadratic elements

For SU the new stabilization parameter can be described as in show in Figure 6, to finally introduce the code modifications in K as following

```
alpha = coth(Pe)-1/Pe;
tau = alpha*h/(2*a);
disp(strcat('Recommended coefficient for the stabilization =',num2str(tau)));
example.tau = cinput ('Stabilization parameter to be used',tau);

beta = 2*((coth(Pe)-1/Pe)-(cosh(Pe))^2*(coth(2*Pe)-1/(2*Pe)))/(2-(cosh(Pe))^2);
tauc = beta*h/(2*a);
disp(strcat('Recommended coefficient for the stabilization on corner nodes = ',num2str(tauc)));
example.tauc = input ('Stabilization parameter to be used (press return for using the recommended one)=
```

Figure 6. Stabilization parameter for SU quadratic elements

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

Figure 7. SU code for quadratic elements

In order to be able to stabilized the convective term in a consistent manner, it have been developed different techniques that follows a similar structure. An extra term, function of the residual, is added to the Galerkin weak form. This new stabilization term can be written as

$$\sum_e \int_{\Omega^e} P(w)\tau R(u)d\Omega$$

$$R(u) = a\nabla u - \nabla\cdot(\nu\nabla u) + \sigma u - s$$

where $P(w)$ is a certain operator, τ is the stabilization parameter and $R(u)$ the residual of the differential equation. For this case, the stabilization parameter is defined as

$$\tau = \beta\frac{h}{2a}$$

Two stabilization methods will be use, SUPG and GLS.

c) SUPG

As $P(w) = (a\cdot\nabla w)$, the equation remains as

$$\int_\Omega w(a\cdot\nabla u)d\Omega + \int_\Omega w\cdot(\nu\nabla u)d\Omega + \sum_e\int_{\Omega^e}(a\cdot\nabla w)\tau\big(a\nabla u - \nabla\cdot(\nu\nabla u)\big)d\Omega$$

$$= \int_\Omega wsd\Omega + \sum_e\int_{\Omega^e}(a\cdot\nabla w)\tau s d\Omega$$

Linear elements Matlab code

```
for ig = 1:ngaus
    N_ig = N(ig,:);
    Nx_ig = Nxi(ig,:)*2/h;
    N2x_ig=N2xi(ig,:)*2/h;
    w_ig = wgp(ig)*h/2;
    Ke = Ke + w_ig*(N_ig'*a*Nx_ig + Nx_ig'*nu*Nx_ig) ...
        + w_ig*(a*Nx_ig')tau*(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)'*s+a*Nx_ig'*tau*s);
end
```

Figure 8. SUPG for linear elements

The new elements of the equation have been added.

Quadratic elements Matlab code

Introducing the established modification modifications for quadratic elements and defining the SUPG/GLS stabilization parameter (Figure 9), the code remains as following

```
alpha = coth(Pe)-1/Pe;
    tau = alpha*h/(2*a);
    disp(strcat('Recommended coefficient for the stabilization =',num2str(tau)));
    example.tau = cinput ('Stabilization parameter to be used',tau);

    beta = ((2*Pe-1)*exp(3*Pe)+(-6*Pe+7)*exp(Pe)+(-6*Pe-7)*exp(-Pe)+(2*Pe+1)*exp(-3*Pe))...
           /( (Pe+3)*exp(3*Pe)+(-7*Pe-3)*exp(Pe)+ (7*Pe-3)*exp(-Pe)-  (Pe+3)*exp(-3*Pe));
    tauc = beta*h/(2*a);
    disp(strcat('Recommended coefficient for the stabilization on corner nodes = ',num2str(tauc)));
    example.tauc = input ('Stabilization parameter to be used (press return for using the recommended one)=
```

Figure 9. SUPG/GLS "tau" quadratic elements implementation

d) GLS

```
for ig = 1:ngaus
    N_ig = N(ig,:);
    Nx_ig = Nxi(ig,:)*2/h;
    N2x_ig=N2xi(ig,:)*2/h;
    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)'*s+a*Nx_ig'*tau*s);
```

Figure 10. SUPG code for quadratic elements

Finally, for GLS as $P(w) = a\nabla w - \nabla \cdot (\nu \nabla w) + \sigma w$, so that the equation is written as

$$\int_\Omega w(a \cdot \nabla u)d\Omega + \int_\Omega w \cdot (\nu \nabla u)d\Omega + \sum_e \int_{\Omega^e} \big(a\nabla w - \nabla \cdot (\nu \nabla w)\big)\tau\big(a\nabla u - \nabla \cdot (\nu \nabla u)\big)d\Omega$$

$$= \int_\Omega w s d\Omega + \sum_e \int_{\Omega^e} \big(a \cdot \nabla w - \nabla \cdot (\nu \nabla w)\big)\tau s d\Omega$$

Linear elements Matlab code

```
for ig = 1:ngaus
    N_ig = N(ig,:);
    Nx_ig = Nxi(ig,:)*2/h;
    N2x_ig=N2xi(ig,:)*2/h;
    w_ig = wgp(ig)*h/2;
    Ke = Ke + w_ig*(N_ig'*a*Nx_ig + Nx_ig'*nu*Nx_ig) ...
        + w_ig*(a*Nx_ig'-nu*N2x_ig')*tau*(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)'*s+(a*Nx_ig'-nu*N2x_ig')*tau*s);
end
```

Figure 11. GLS code for linear elements

Quadratic elements Matlab code

As in the previous cases, the modifications for quadratic elements are implemented. Using the same τ than in SUPG (Figure 9), the code remains as

```
for ig = 1:ngaus
    N_ig = N(ig,:);
    Nx_ig = Nxi(ig,:)*2/h;
    N2x_ig=N2xi(ig,:)*2/h;
    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)'*s+Tau*(a*Nx_ig'-nu*N2x_ig')*s);
end
```

Figure 11. GLS for quadratric elements

4. Results and conclusions

The problem described at the beginning of the report will be solve several times, changing the methods and the source term. Then, the results will be compared and discussed in order to get some successful conclusions.

a) *Solve the probem for* $a = 1,\ \nu = 0.01,\ s = 0$ *and 10 linear elements.*

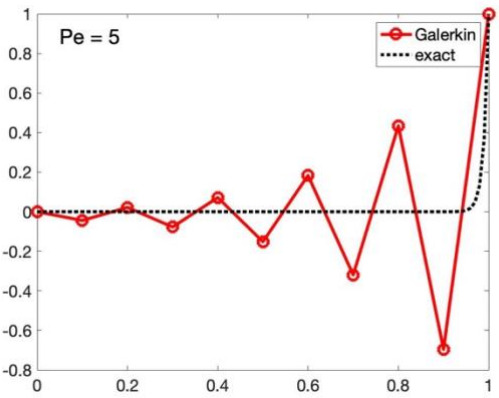

Figure 12. Galerkin method for 10 linear elements

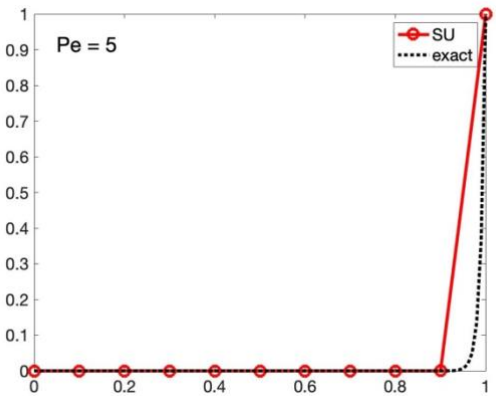

Figure 13. SU method for 10 linear elements

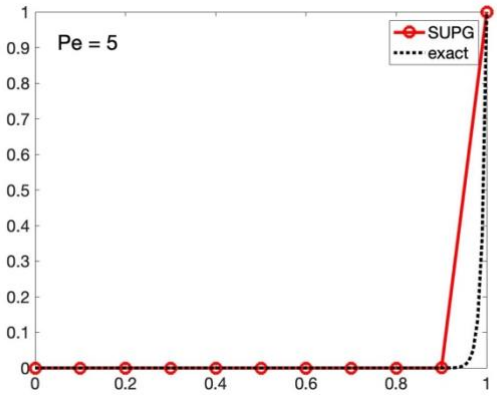

Figure 14. SUPG method for 10 linear elements

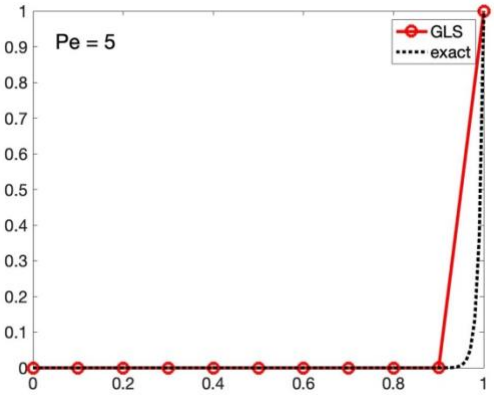

Figure 15. GLS method for 10 linear elements

As might be expected for $Pe = 5$, Galerkin performance method shows notable oscillations and no exact solutions in the nodes, while the other three methods obtain the exact ones. It can be seen that the other three solutions are exactly the same, this is due to the fact that when we are working with 1D linear elements with constant coefficients and no source term, due to the fact that the 2nd order derivatives are equal to zero and the τ parameter has the same definition.

b) *Solve the problem for* $a = 1,\ \nu = 0.01,\ s = 10\exp(-5x) - 4\exp(-x)$ *and 10 linear elements for SU, SUPG and GLS.*

First of all, it is needed to remark that a change in the "ExactSol" and "SourceTerm" codes are made in order to introduce the new variable source term.

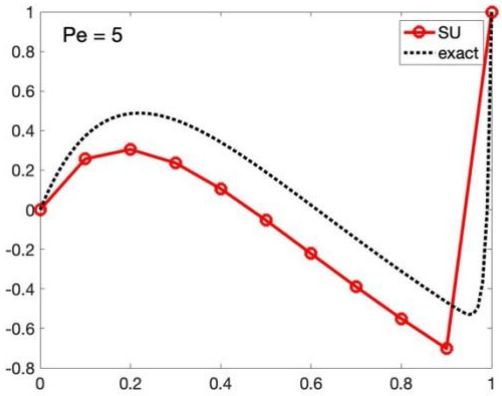

Figure 15. SU method with variable source term

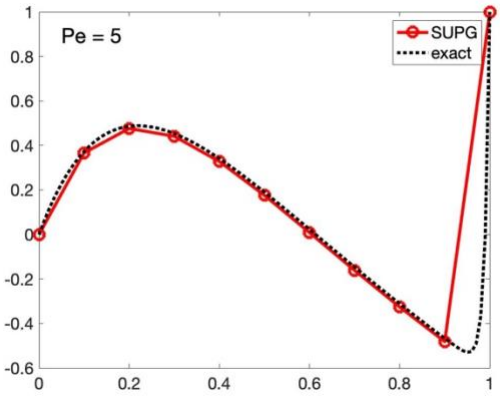

Figure 16. SUPG method with variable source term

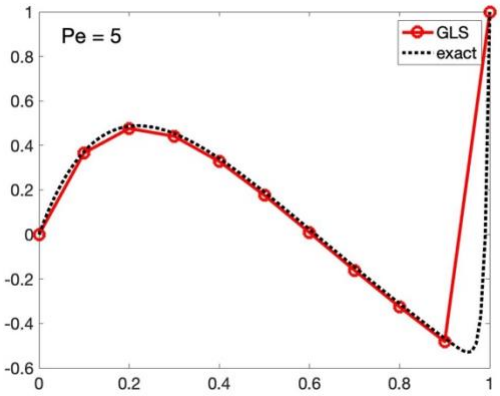

Figure 17. GLS method with variable source term

Introducing a no constant source term, the differences in the precision of the results obtained from SU and SUPG/GLS methods can be clearly observed. As it has been explained, SU method is not a consistent method and as Péclet number increases

(convection-dominated), the differences between the exact solution and the solution at the nodes is higher too. However, the remaining methods performs properly, obtained almost the same solution in which the nodal solutions are the exact ones.

c) *Solve the problem for* $a = 1,\ \nu = 0.01,\ s = 10\exp(-5x) - 4\exp(-x)$ *and 5 quadratic elements for SU, SUPG and GLS.*

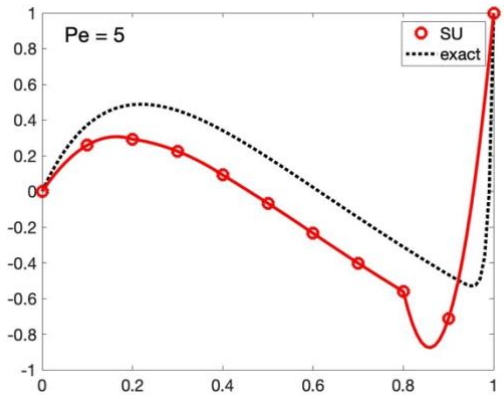

Figura 18. SU method for quadratic elements and s term

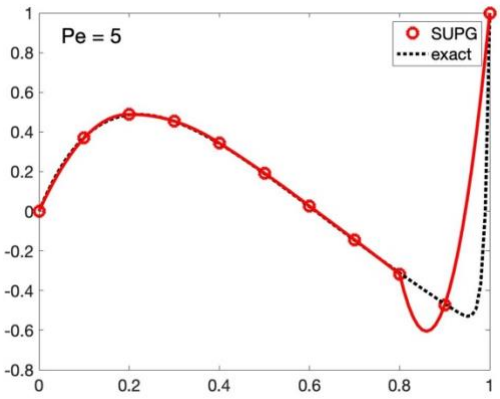

Figura 19. SUPG method for quadratic elements and s term

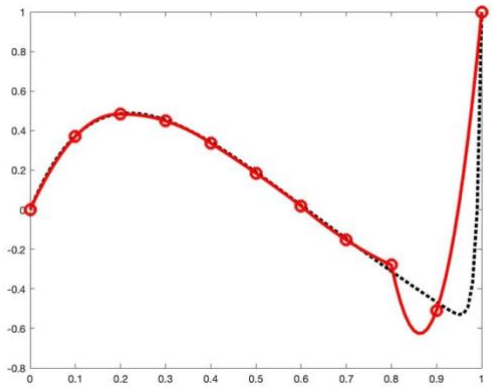

Figura 20. GLS method for quadratic elements and s term

Finally the results obtained using quadratic elements are compered. The general behaviour of the methods is quite similar to the previous case, it is said, for SU due to the variable source term, the high Péclet number and the fact that it is not a consistent method, the nodes solution are quite away from the exact ones. However, SUPG and GLS obtain exact solution in the nodes. It can highlighted that as parabolic lines are being used between the nodes, more accurate idea of the real method solution is achieved.