HW1 - REPORT

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

Finite Element in Fluids

Álvaro Rodríguez Luis

1. Introduction

The equation we want to solve to solve is:

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

with $u_0 = 0$ and $u_1 = 1$. Using Galerkin method, we will solve it for the following cases:

1) a = 1, v = 0.2, 10 elements.
2) a = 20, v = 0.2, 10 elements.
3) a = 1, v = 0.01, 10 elements. (This case will be also solved with SU, SUPG and GLS methods.)
4) a = 1, v = 0.01, 50 elements.

This will be done using both linear and quadratic elements and for two different functions for the source term. (s = 0 and $s = 10 \cdot e^{-5x} - 4 \cdot e^{-x}$)

2. Code modifications

In order to follow the instructions of the report, presented in the introduction, some modifications where made to the provided code, and some new functions where implemented. In the following pages these modifications are shown and highlighted in yellow.

main.m (only the modifications)

```
...
% Discretization
disp(' ')
nElem = cinput('Number of elements',10);
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 % An extra point is added to the center of the elements
    nPt = 2*nElem + 1;
    h = (dom(2) - dom(1))/nElem;
    X = (dom(1):0.5*h:dom(2))';
    T = [1:nPt-2; 2:nPt-1; 3:nPt]';
end
...
```

SUPG_system.m (modifications with respect to SU_system.m are highlighted)

```
function [K,f] = SUPG_system(X,T,referenceElement,example)
% [K,f] = SU_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 SUPG 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 = referenceElement.N2xi;
% 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);

% 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,:)*(2/h)*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 + w_ig*(a*Nx_ig)'*tau*s;
    end
    % Assmebly
    K(Te,Te) = K(Te,Te) + Ke;
    f(Te) = f(Te) + fe;
end
```

GLS_system.m (modifications with respect to SU_system.m are highlighted)

```
function [K,f] = GLS_system(X,T,referenceElement,example)
% [K,f] = SU_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 = referenceElement.N2xi;

% 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);

% 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,:)*(2/h)*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 + w_ig*(a*Nx_ig-nu*N2x_ig)'*tau*s;
    end
    % Assmebly
    K(Te,Te) = K(Te,Te) + Ke;
    f(Te) = f(Te) + fe;
end
```

SourceTerm.m

```
function res = SourceTerm(x,example)
% res = SourceTerm(x,problem)
% Source term for the convection-diffusion equation

problem = example.problem;
if problem == 1
    %res = 0;
    res = 10*exp(-5*x) - 4*exp(-x);
elseif problem == 2
    res = 1;
elseif problem == 3
    res = sin(pi*x);
end
```

ExactSol.m (explanation of how the new exact solution is obtained in Annex)

```
function res = ExactSol(x,example)
%
% res = ExactSol(x,example)
% Analytical solution of a 1D convection-diffusion problem
% with essential boundary conditions on both ends.

a = example.a;
nu = example.nu;
problem = example.problem;

if problem == 1
    %res = (1-exp(x*a/nu))/(1-exp(a/nu));
    alpha = -2/(a+5*nu);
    beta = 4/(a+nu);
    M = [1 1; 1 exp(a/nu)];
    F = [-alpha-beta;1-alpha*exp(-5)-beta*exp(-1)];
    V = M\F;
    A = V(1);
    B = V(2);
    res = A + B*exp(a*x/nu) + alpha*exp(-5*x) + beta*exp(-x);
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
```

3. Results

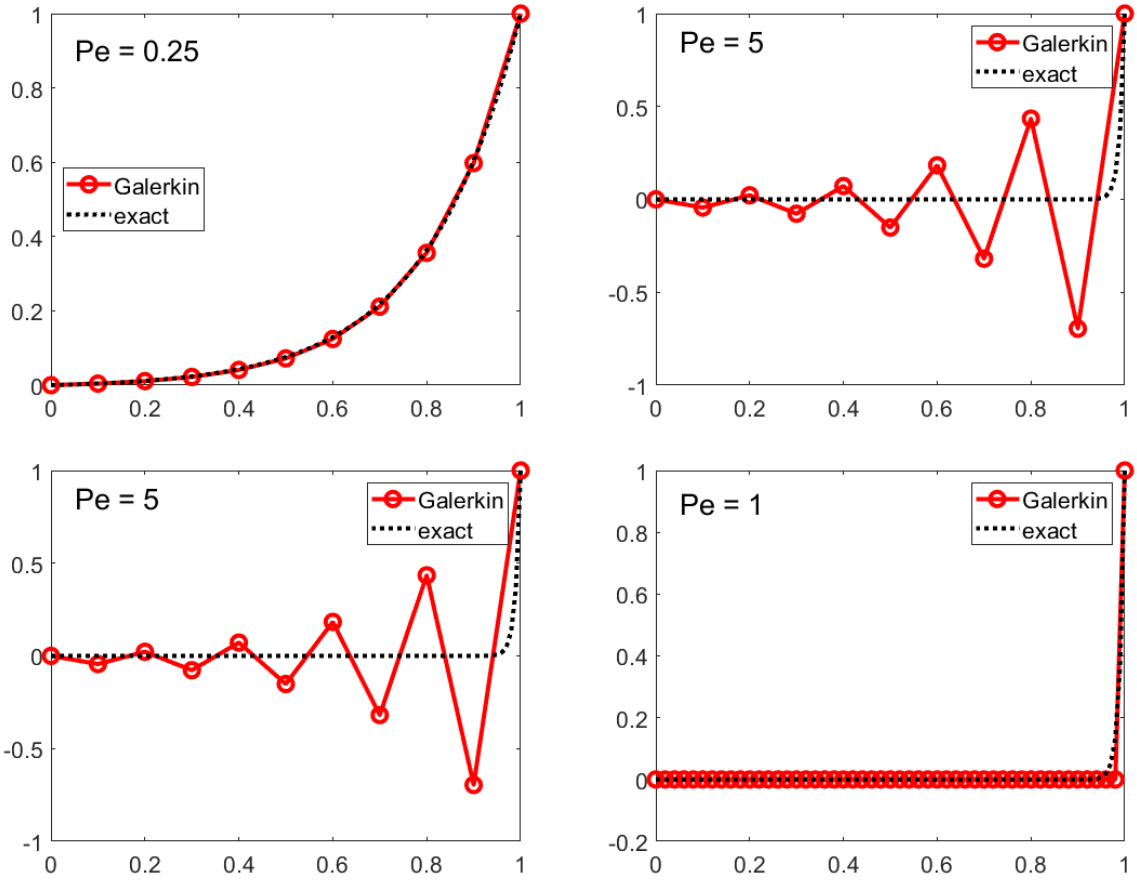

Figure 1. Results for the Galerkin method with linear elements and s = 0, for case 1 (top left), case 2 (top right), case 3 (bottom left) and case 4 (bottom right). This order holds for figures 3, 5 and 7.

It is observed that for cases 2 and 3 the Galerkin method is not stable although it shows great accuracy for case 1.

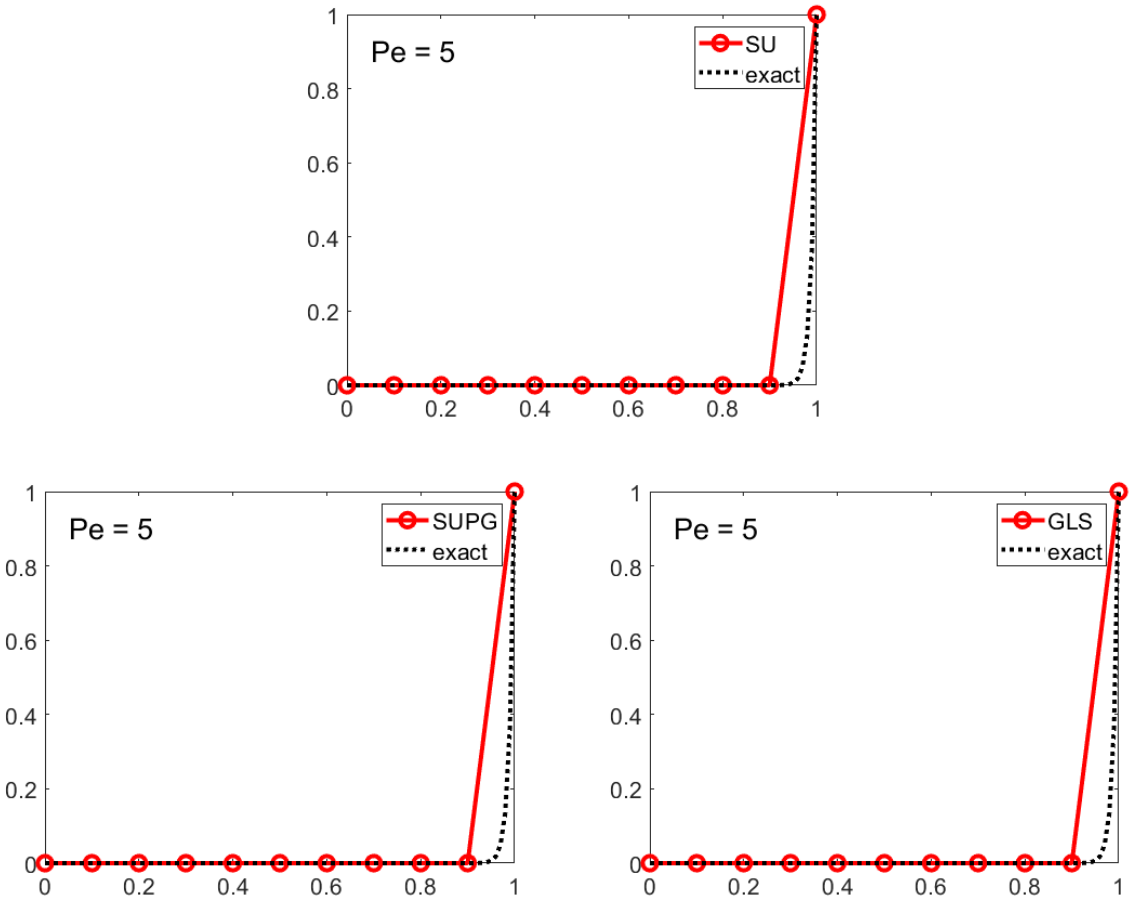

Figure 2. Results of case 3 for the SU, SUPG and GLS methods with linear elements and s=0.

It is observed that for these three methods, the result for case 3 is stable. Note that the results for the three methods are the same. This is because using linear elements, the terms that make the three methods different do not act.

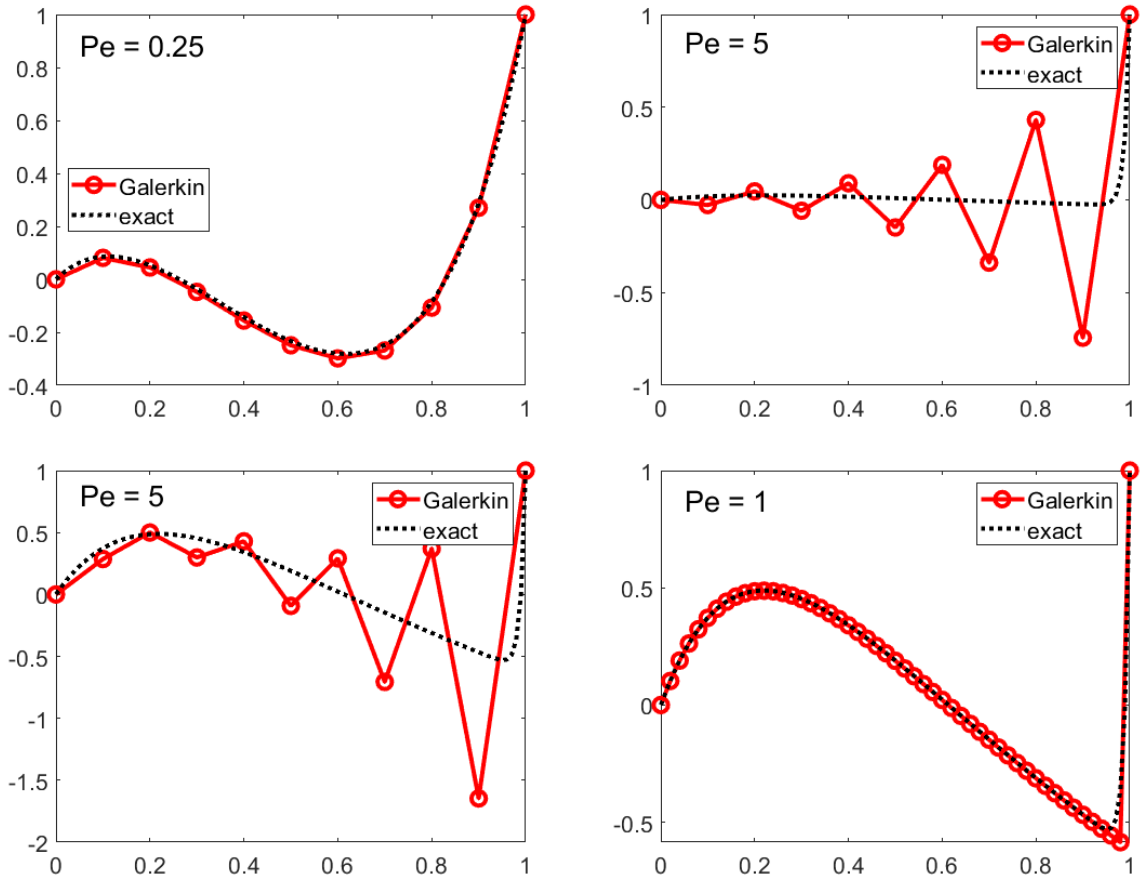

Figure 3. Results for the Galerkin method with linear elements and $s = 10 \cdot e^{-5x} - 4 \cdot e^{-x}$, for cases 1 to 4.

With the non-zero source term, similar results are obtained: the Galerkin method results for cases 2 and 3 are not stable although it shows great accuracy for the case 1.

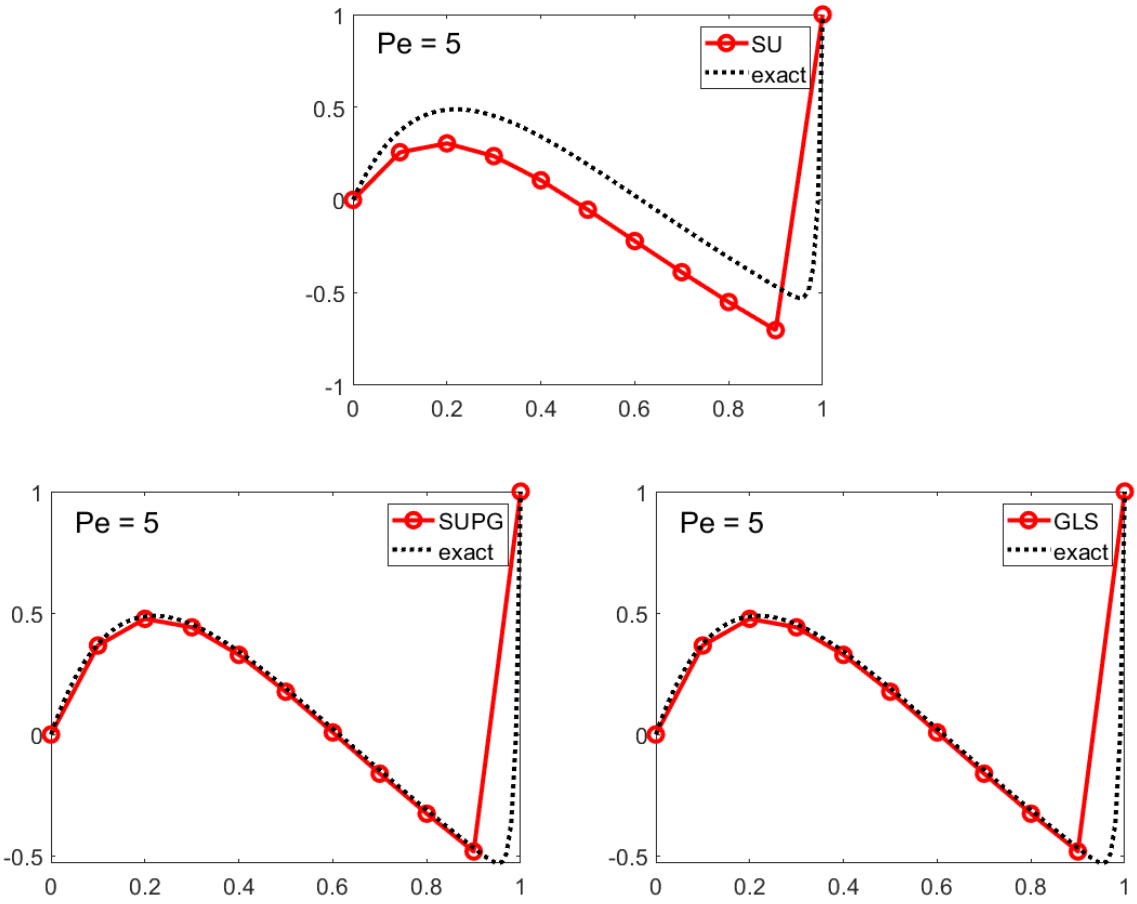

Figure 4. Results of case 3 for the SU, SUPG and GLS methods with linear elements and $s=10\cdot e^{-5x} - 4\cdot e^{-x}$.

For the non-zero source term, the method is stable for these three methods, but the precision is lost for the SU method.

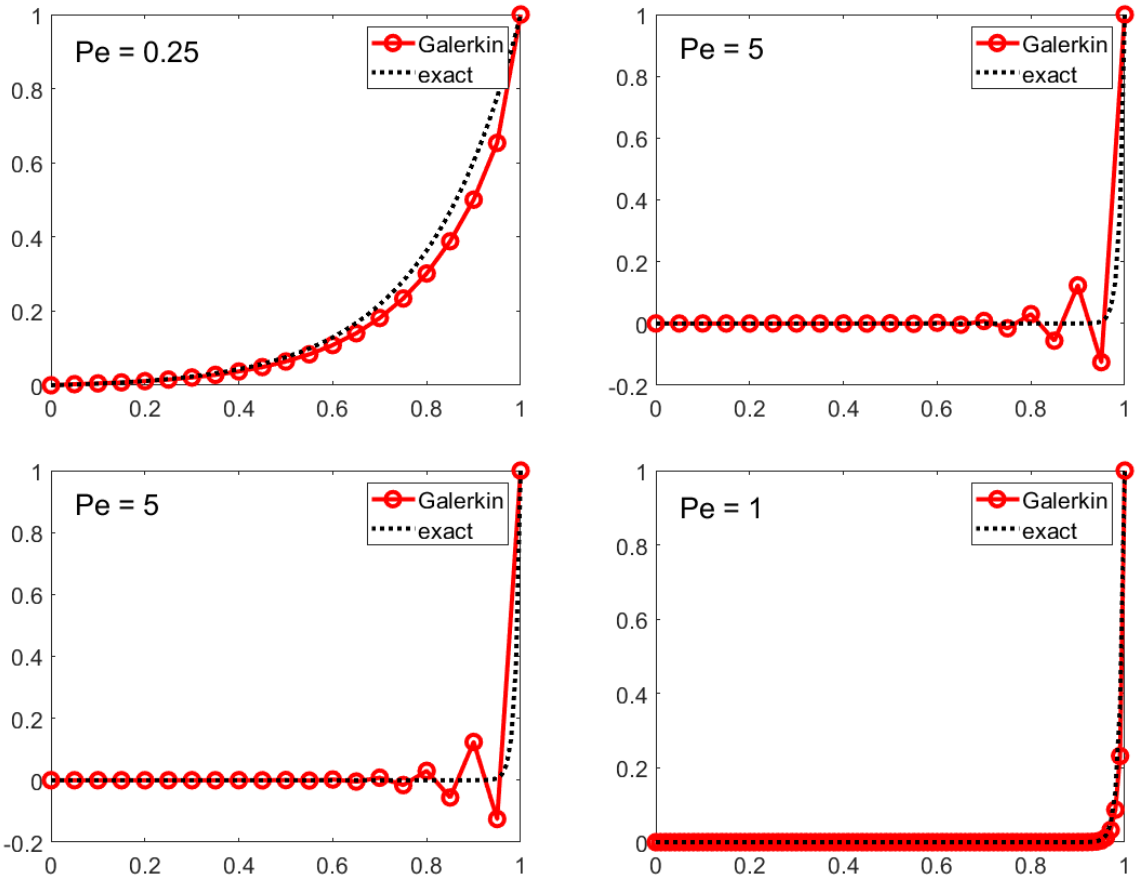

Figure 5. Results for the Galerkin method with quadratic elements and s = 0, for cases 1 to 4.

The same results of figures 1 and 3 are observed, but now, for quadratic elements, the Galerkin method does not show great accuracy for case 1.

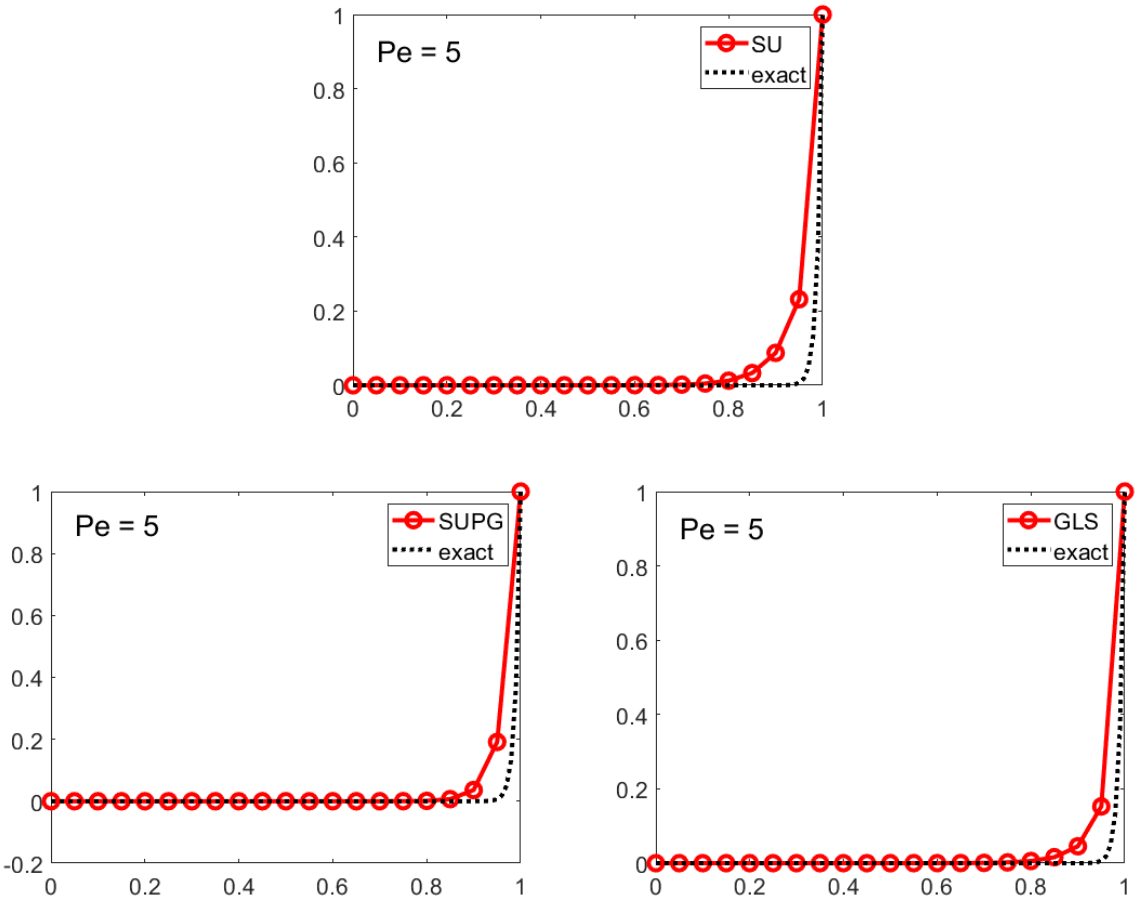

Figure 6. Results of case 3 for the SU, SUPG and GLS methods with quadratic elements and s=0.

Now, the results of the three methods differ slightly, and, in comparison with the results in figure 2, the accuracy has decreased.

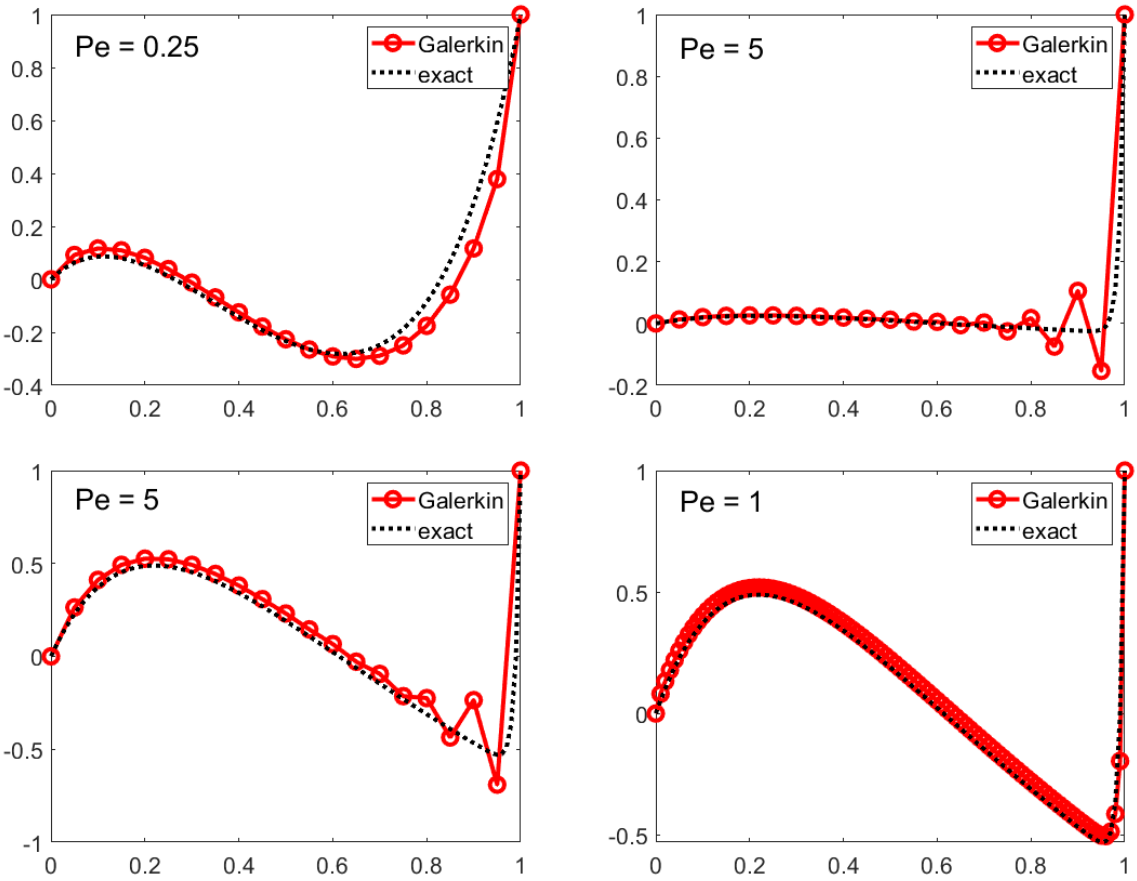

Figure 7. Results for the Galerkin method with quadratic elements and $s = 10 \cdot e^{-5x} - 4 \cdot e^{-x}$, for cases 1 to 4.

The results in this case are the same as in Figure 5.

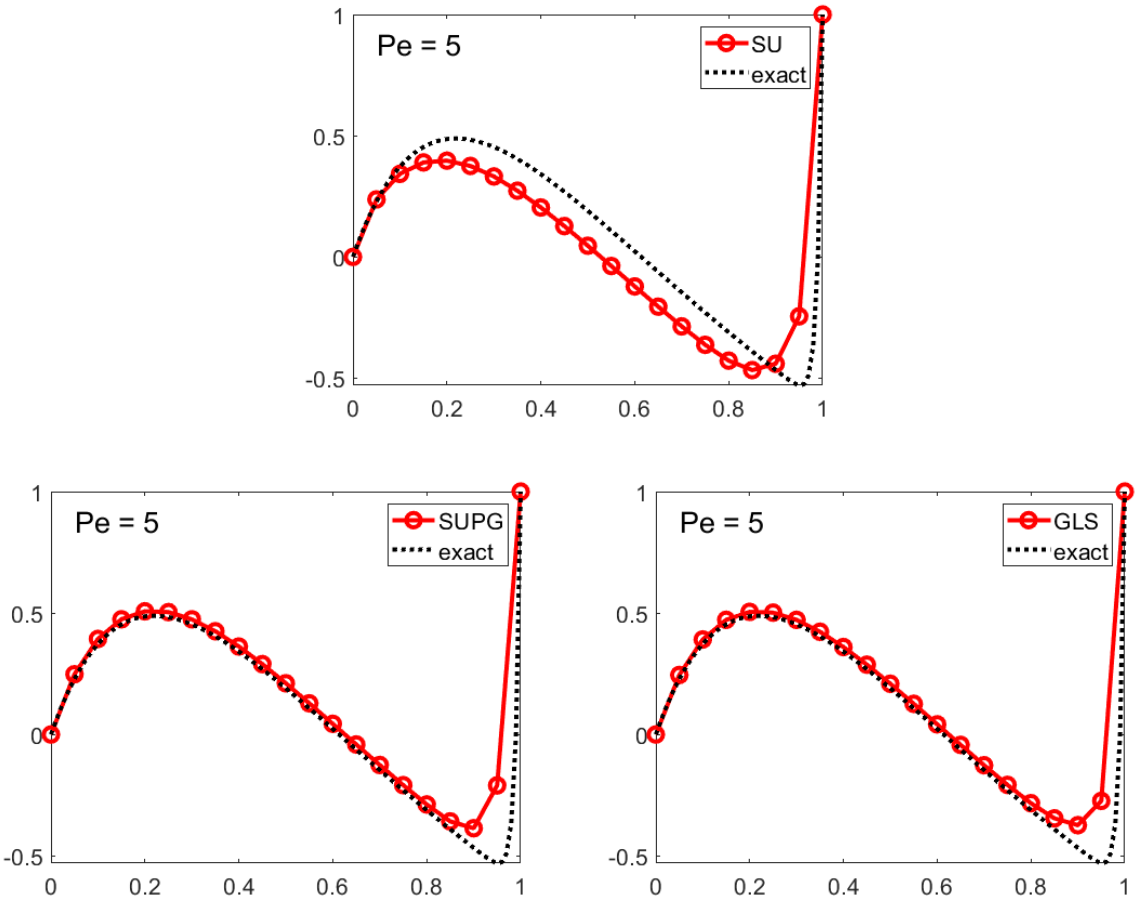

Figure 8. Results of case 3 for the SU, SUPG and GLS methods with quadratic elements and $s = 10 \cdot e^{-5x} - 4 \cdot e^{-x}$.

The results in this case are the same as in figure 4, but here the results for SUPG and GLS methods are slightly worse.

Annex

To solve the ODE:

$$-\nu \cdot u_{xx} + a \cdot u_x = 10 \cdot e^{-5x} - 4 \cdot e^{-x}$$
$$u(0) = 0;\ u(1) = 1$$

We first solve the homogeneous equation, using the characteristic polynomial:

$$-\nu \cdot y^2 + a \cdot y = 0 \rightarrow y = 0;\ y = a/\nu \rightarrow u_h = A + B \cdot e^{x \cdot a/\nu}$$

Then, we find a particular solution, in this case of the form:

$$u_p = \alpha \cdot e^{-5x} + \beta \cdot e^{-x}$$

Obtaining coefficients:

$$\alpha = \frac{-2}{a + 5\nu};\ \beta = \frac{4}{a + \nu}$$

Finally, we impose the boundary conditions to the full solution ($u = u_h + u_p$) in order to find the values of A and B in the homogeneous solution. This is done by solving a linear system of equations.