General Problem Consisting all function and methods

Karan(22101028)

Contents

- Problem Description
- Pre-Processor
- Exact solution for all possible case
- Processor
- Weak form
- optimization for different boundary condition
- Solving equation
- piecewise displacement function
- error calculation
- ploting Uexact and U approx
- ploting D(Uexat) and D(Uapprox)
- error ploting between U(Fem), Uexact
- Post processing (SCR)


```
% This problem comprises Generalised method for lagrangian shape function
% Superconvergence rate
% Numerical integration
%Probelm 1: represents Bar with end loads
%Probelm 2:Repesents Bar with fixed supports
```

Problem Description

Here we are solving Bar with fixed support k=10 E*a=const p=[1,3]

```
clc
clear all
syms x
```

Pre-Processor

```
% A typical algorithm for reading the input data

% Domain data
L=1;         %length of bar
x0=0;
xl=L;        % Domain of analysis (x0,xl)

% Geometric data and Material data
Ea=1;        % Area of Bar and multiple with Youngs modulus
k=10;        % Stifness of rubber casing

% Force Data
f=10*x;    % distributed Force on Bar

%Boundary data
u0=0;        % Displacement at  x==0
ul=0;        % Displacement at  x==L
P0=5;        % Force at end x=l
Pl=10;       % Force at end x=0


%Mesh Data and shape function
Nel=[2,4,8];       % Number of element
oder=[1,3];


%p=input('enter the order of shape function');
% Switching from one problem to other
%flag=input("enter the problem which you want to solve \n Probelm 1: represents Bar with end loads \n Probelm 2:Repesents Bar with fixed supports ");
flag=2;

if flag==1
    %k=input('please enter value of rubber casing and 0 for no Rubber casing ');
   % P0=input('please enter the values for P0 if you want to change');
    %Pl=input('please enter the values for Pl if you want to change');
else
    flag=2;

end
```

Exact solution for all possible case

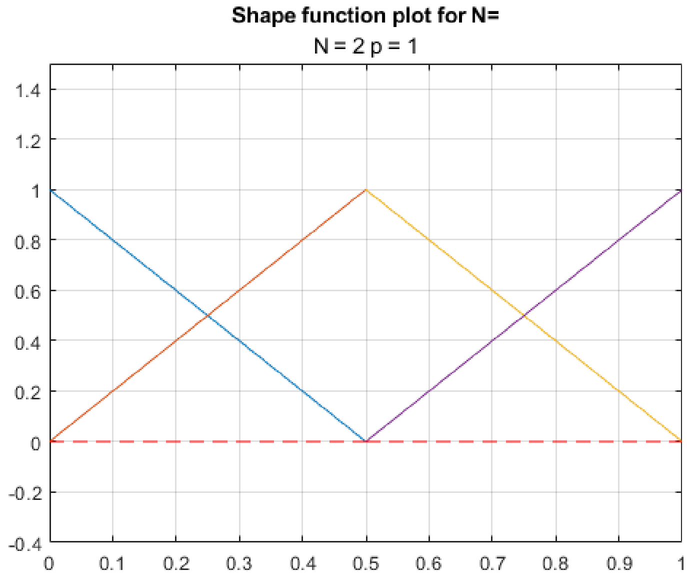
```
X=linspace(x0,xl,100);
if (flag==1)&&(k==0)
    disp("exact solution does not exist");
elseif (flag==1)&&(k~=0)
    syms U_e(x)
    DU_e=diff(U_e);
    ode=diff(U_e,x,2)==k*U_e-f;
                                                % Boundary conditions
    cond1 = DU_e(0) == P0;
    cond2 = DU_e(L) == Pl;
    conds=[cond1 cond2];
                                                % Exact solution for
    U_eSol=dsolve(ode,conds);
    U_e=simplify(U_eSol);
    DU_e=diff(U_e);

else

    syms U_e(x)
    DU_e=diff(U_e);
    ode=diff(U_e,x,2)==(1/Ea)*((k*U_e)-f);
                                                 % Boundary conditions
    cond1 = U_e(0) == 0;
    cond2 = U_e(L) == 0;
    conds=[cond1 cond2];

                                                 % Exact solution for
    U_eSol=dsolve(ode,conds);
    U_e=simplify(U_eSol);

    DU_e=diff(U_e);

end
% Array of values of U at different positions of x
for s=1:length(X)
    Uex(s)=double(subs(U_e,x,X(s)));
    DUex(s)=double(subs(DU_e,x,X(s)));
end
```

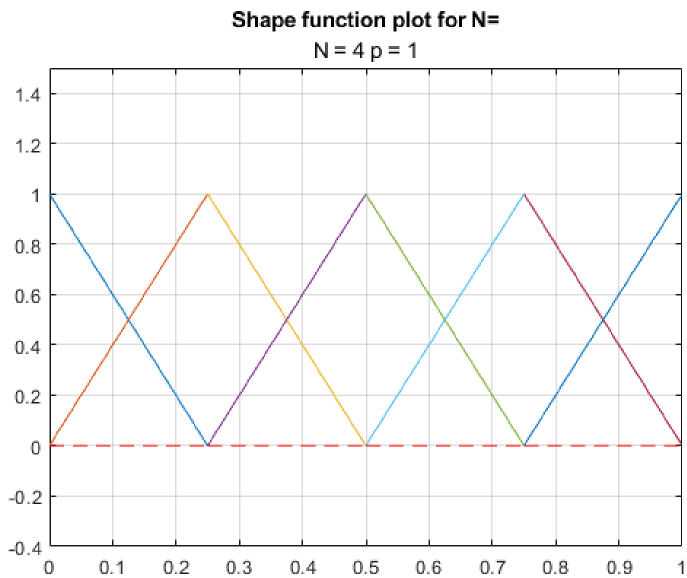
Processor

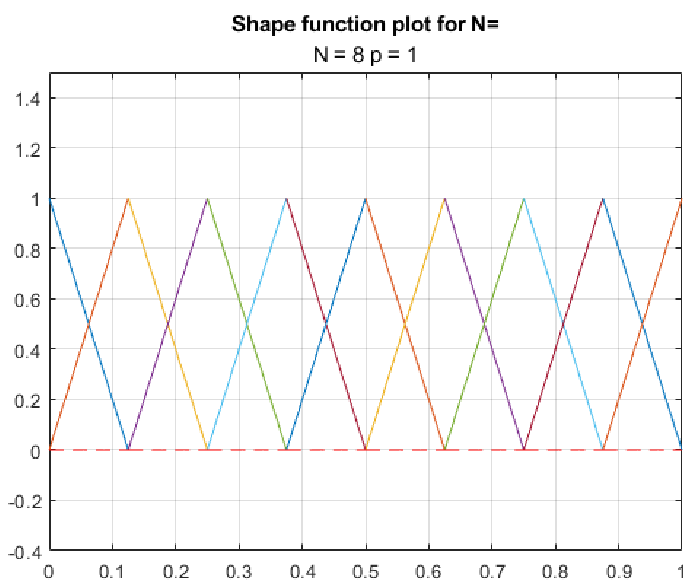
```
for Z=1:length(oder)
```

```
    p=oder(Z);
for z=1:length(Nel)
```

```
    % Discretisation of Domain and shape function for given number of
    % element and order
    n_phi=p+1;       % number of basis function in a given element for gievn p
    n_elem=Nel(z);         % number of element in the domain
    n_node=(Nel(z)*p)+1;        %total number of nodes inside domain
    figure
    [shape_fun,nodes,nodes_elem]=phi_all(p,Nel(z),L);


    phi=shape_fun;
```

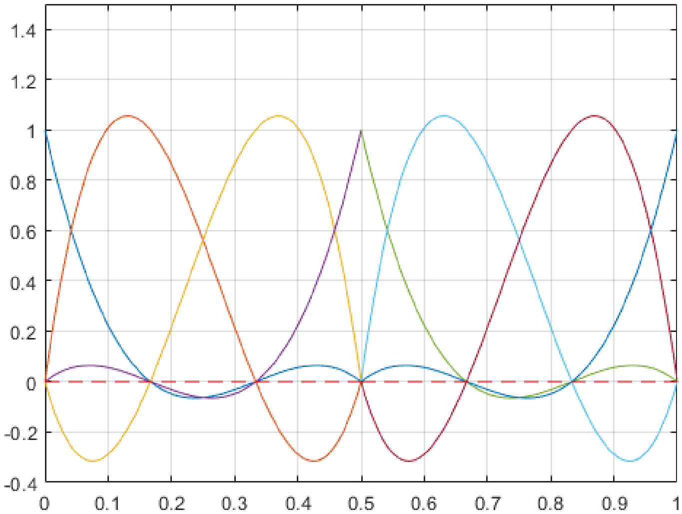

Shape function plot for N=
N = 2 p = 1


Shape function plot for N=
N = 4 p = 1


Shape function plot for N=
N = 8 p = 1

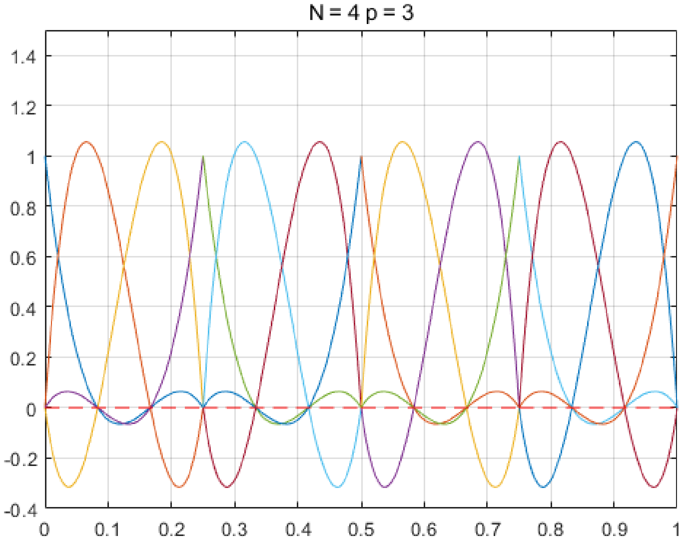
Shape function plot for N=

N = 2 p = 3



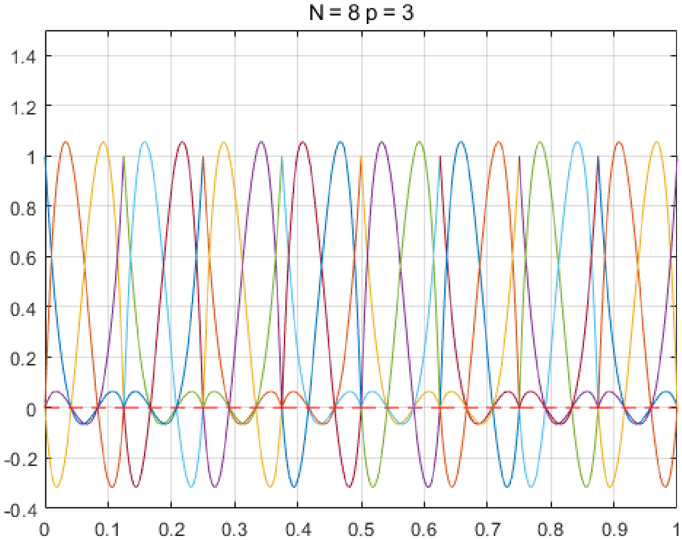
Shape function plot for N=

N = 4 p = 3



Shape function plot for N=

N = 8 p = 3



Weak form

```
%fl= local shape function
%fg= global shape function
F=zeros(n_node,1);               % global load vector
K=zeros(n_node,n_node);               % Global stiffnes matrix
K_l=zeros(n_phi,n_phi,n_elem);           % local stiffness matrix
F_l=zeros(n_phi,n_elem);              % local load vector

for a=1:n_elem                  % a=kth element
    l_a=((a-1)*p)+1;            % element node counter
    u_a=(a*p)+1;
    l_lim=nodes(l_a);   % lower limit of element
    u_lim=nodes(u_a);   % upper limit of element
    for i=1:n_phi               % for local load vector
        f_l=f*phi(i,a);
        od1=polynomialDegree(f_l);
        F_l(i,a)=gauss_quad(f_l,od1,l_lim,u_lim);
        I=p*(a-1)+i;            % location for global load vector
        F(I,1)=F(I,1)+F_l(i,a);  % Assembly process for load vector

        for j=1:n_phi           % for local stiffness matrix
            f_lk=((Ea*diff(phi(i,a))*diff(phi(j,a)))+(k*(phi(i,a))*(phi(j,a))));
            od2=polynomialDegree(f_lk);
            K_l(i,j,a)=gauss_quad(f_lk,od2,l_lim,u_lim);
            J=p*(a-1)+j;            % location for global stiffness matrix
            K(I,J)=K(I,J)+K_l(i,j,a); % Assembly process for stiffness Matrix
        end
    end
end
```

optimization for different boundary condition

```
if flag==2

    % for end condition (u=0 at x=0) and (u=0 at x=L)
    % step-1:putting K(1,1)=1 and other elements of row 1 = 0 and tempering
    % F vector acordingly
    K(1,1)=1;
    F(1)=u0;
    for b=2:n_node
        K(1,b)=0;
        F(b)=F(b)-u0*K(b,1);
        K(b,1)=0;
    end

    % Step-2: Similar proceedure for K(n,n)=1 and so on
    K(n_node,n_node)=1;
    F(n_node)=ul;
    for c=(n_node-1):-1:2
        K(n_node,c)=0;
        F(c)=F(c)-ul*K(c,n_node);
        K(c,n_node)=0;
    end
elseif ((flag==1)&&(k==0))
    %K matrix is singular, no unique solution exists\n'
    %change Pl such that F(N)=0 and change K(N,N)=1\n
    K(:,n_node)=0;
    K(n_node,:)=0;
    K(n_node,n_node)=1;
    Pl=-F(n_node);
    F(1)=F(1)-P0;
    F(n_node)=F(n_node)+Pl;
else
    F(1)=F(1)-P0;
    F(n_node)=F(n_node)+Pl;


end
```

Solving equation

```
alpha1=zeros(n_node,1);
alpha1(:,Z)=K\F;
t=linspace(0,1,100);
```

piecewise displacement function

```
    U=0;
    alpha_elem=zeros((p+1),Nel(z));
    for g=1:Nel(z)
        alpha=alpha1(:,Z);
        alpha_elem(:,g)=alpha([(((g-1)*p)+1):(((g-1)*p)+p+1)]); %storing alpha elementwise
        for d=1:n_phi
            U_loc_n(d,g)=phi(d,g)*alpha_elem(d,g);
        end

        U_loc(g,1)=sum(U_loc_n(:,g));
        U=piecewise((nodes_elem(g,1)<=x)&(x<=nodes_elem(g,n_phi)),U_loc(g,1),U);
    end

    Val(z,:)=subs(U,x,t);   % array of approximate solution (val, z)
    DVal1(z,1)=Ea*diff(U);
    DU(z,1)=diff(U);
    DVal(z,:)=subs(DVal1(z,1),x,t);       % array of dval   (DVal,1)
```

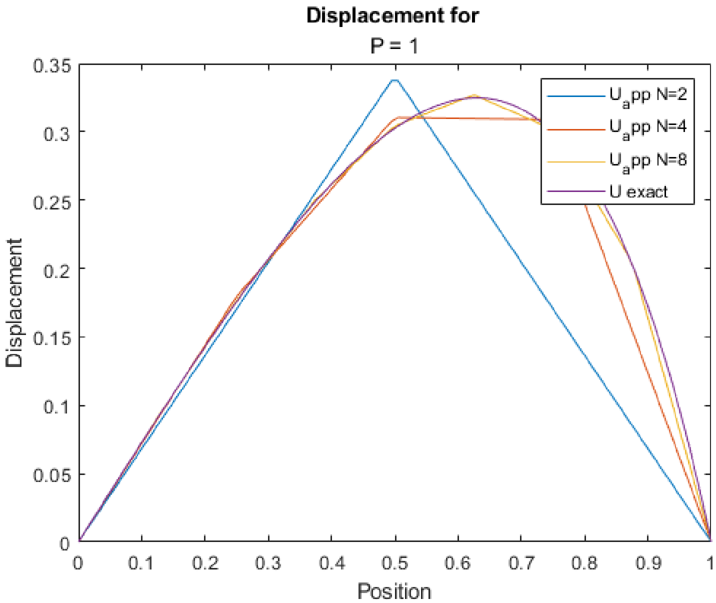
error calculation

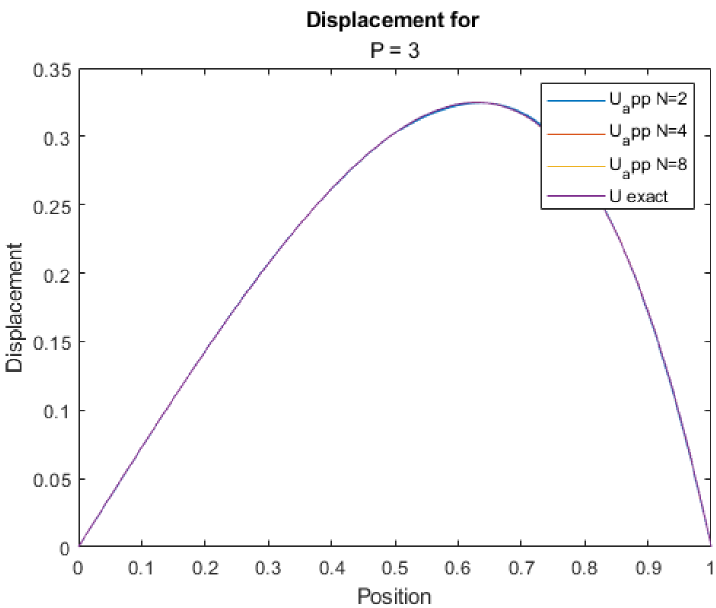
```
    err(z)=U_e-U;
    Derr=diff(err(z));
    od3=10;             % Taking truncation order for exponential function
    B(z)=gauss_quad(((Ea*Derr*Derr)+k*Derr*Derr),od3,0,L);
    errs=B(z);
    err(z)=sqrt(double(errs));
```

```
end
```

ploting Uexact and U approx

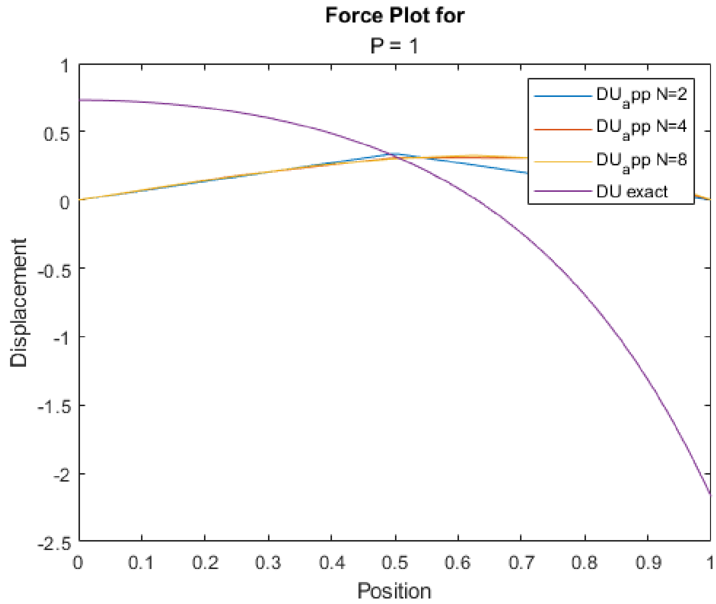
```
figure
for z5=1:length(Nel)

    plot(t,Val(z5,:));
    hold on
end
Val_e=subs(U_e,x,t);
plot(t,Val_e)
A1=num2str(p);
A2=strcat("P = ",A1);
[q,s]=title("Displacement for ",A2);
xlabel("Position");
ylabel("Displacement");
legend("U_app N=2","U_app N=4","U_app N=8","U exact");
```

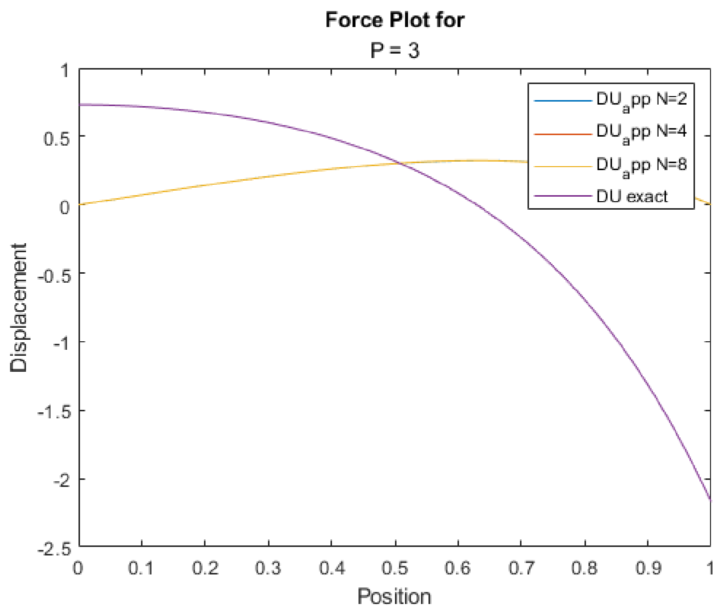




ploting D(Uexat) and D(Uapprox)

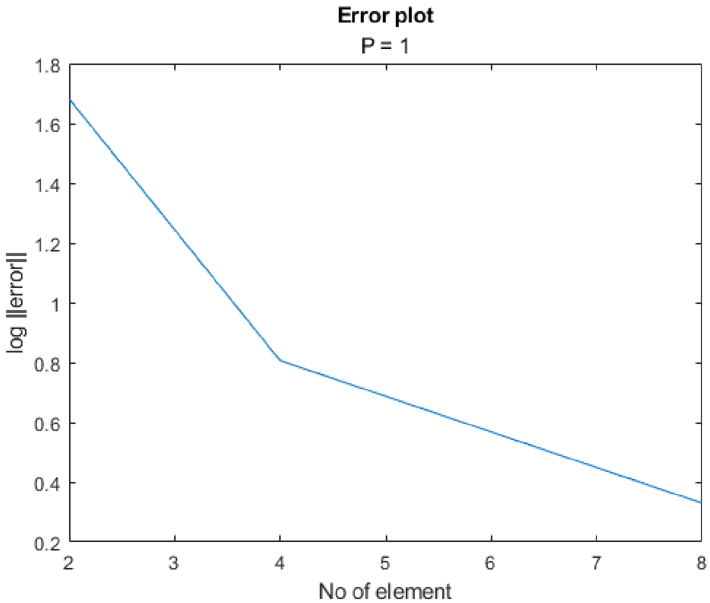
```
figure
for z6=1:length(Nel)

    plot(t,Val(z6,:));
    hold on
end
DVal_e=Ea*diff(U_e);
Val_e1=subs(DVal_e,x,t);
plot(t,Val_e1);
xlabel("Position");
ylabel("Displacement");
legend("DU_app N=2","DU_app N=4","DU_app N=8","DU exact");
[q1,s]=title("Force Plot for",A2);
```

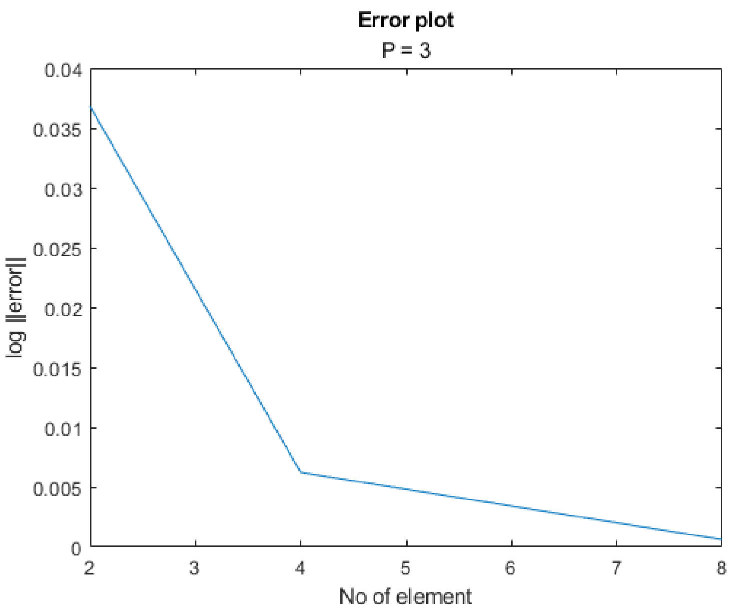




error ploting between U(Fem), Uexact

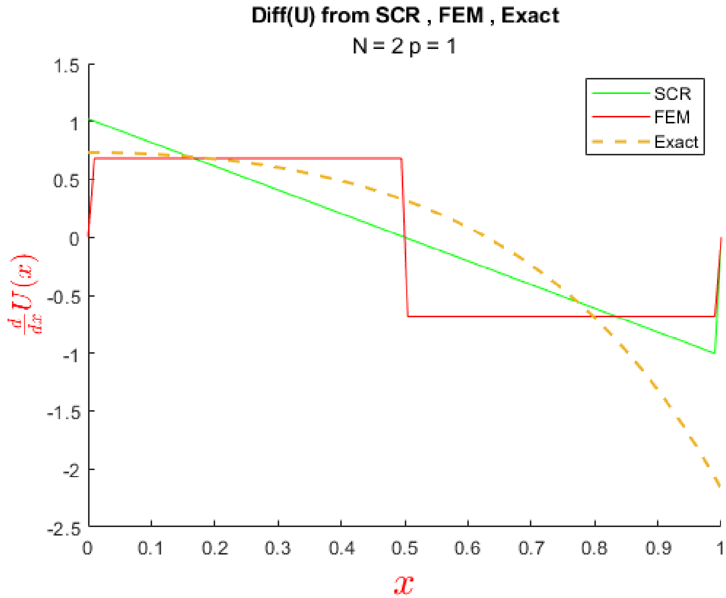
```
figure
plot(Nel,err);
xlabel("No of element");
ylabel("log ||error||");
[q2,s]=title("Error plot",A2);
```

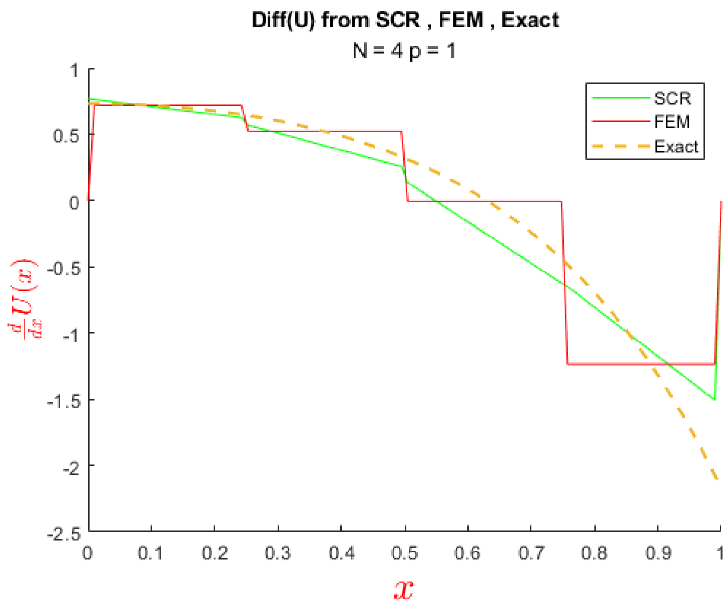


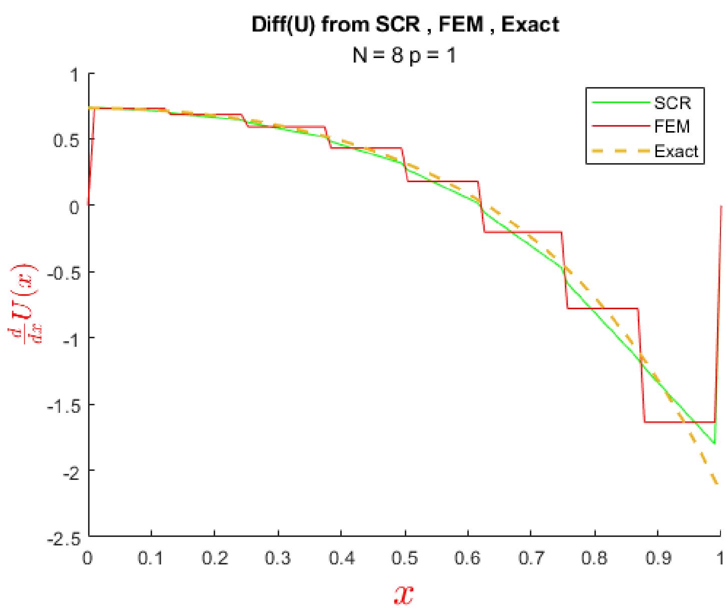


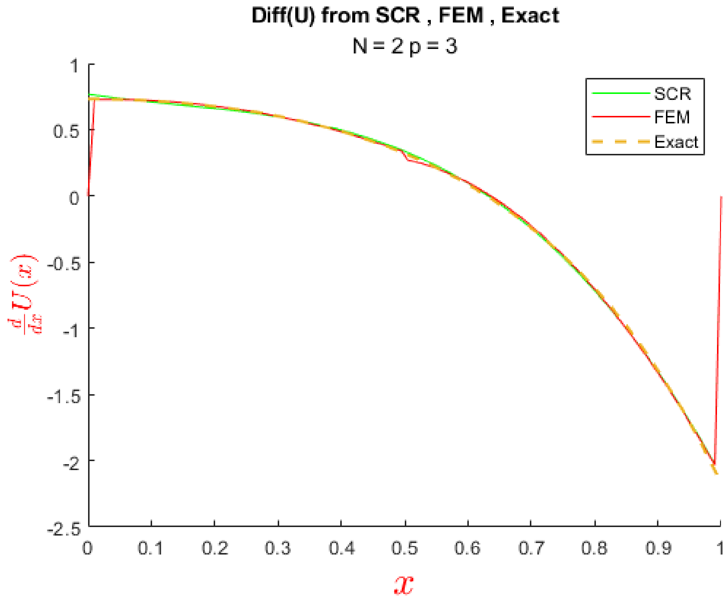
Post processing (SCR)

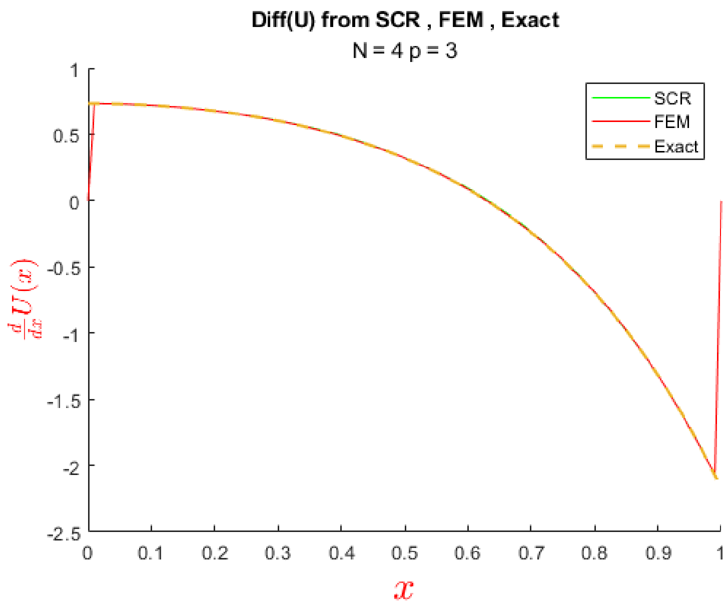
```
for i=1:length(Nel)

    figure
    [a1,b1,c1,d1]=scrf_test(Nel(i),p,DU(i,1),DU_e,L);  %test function
    num=strcat("N = ",num2str(Nel(i))," p = ",num2str(p));
    [q3,s]=title("Diff(U) from SCR , FEM , Exact",num);
    t = '$\frac{d}{dx}U(x)$';
    t1='$x$';
    ylabel(t,'interpreter','latex',FontSize=14,Color='r');
    xlabel(t1,Interpreter="latex",FontSize=20,Color='r');

end
```

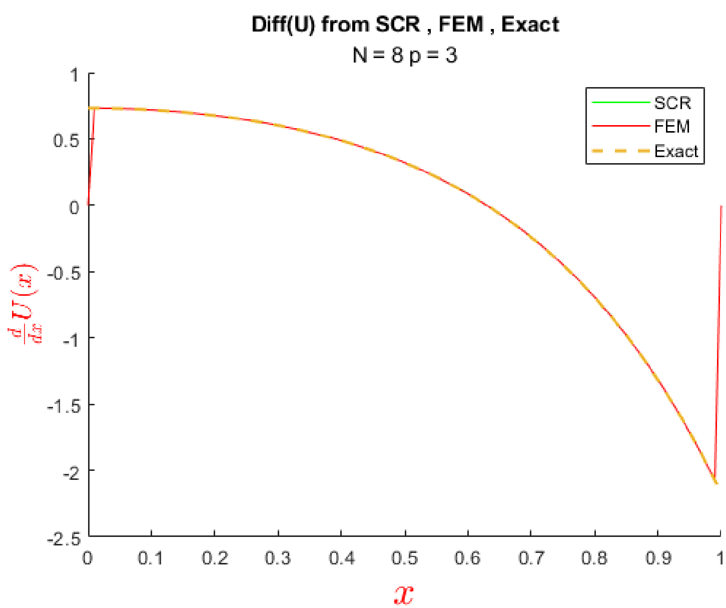

Diff(U) from SCR , FEM , Exact
N = 2 p = 1
SCR
FEM
Exact
$\frac{d}{dx}U(x)$
x


Diff(U) from SCR , FEM , Exact
N = 4 p = 1
SCR
FEM
Exact
$\frac{d}{dx}U(x)$
x


Diff(U) from SCR , FEM , Exact
N = 8 p = 1
SCR
FEM
Exact
$\frac{d}{dx}U(x)$
x


Diff(U) from SCR , FEM , Exact
N = 2 p = 3
SCR
FEM
Exact
$\frac{d}{dx}U(x)$
x


Diff(U) from SCR , FEM , Exact
N = 4 p = 3
SCR
FEM
Exact
$\frac{d}{dx}U(x)$
x


Diff(U) from SCR , FEM , Exact
N = 8 p = 3
SCR
FEM
Exact
$\frac{d}{dx}U(x)$
x

```
end
```