```
function Project_01_Linear

L=0.5;                                  % Rod Length %
Le=zeros(5,1);                          % Element Size Matrix %
d=zeros(5,1);                           % L2 Error Values Matrix %
e=zeros(5,1);                           % Energy Error Values Matrix %
nn=[11,41,161,641,2565];                % No. of nodes %
x=zeros(5,1);
y1=zeros(5,1);                          % Extra Matrices used for Storage %
y2=zeros(5,1);
for i=1:5
    Le(i)=L/(nn(i)-1);                  % Le = Element Length %
    [d(i),e(i)]=solve_fem(nn(i));       % Solves model for given set of values %
    x(i)=log10(Le(i));
    y1(i)=log10(d(i));
    y2(i)=log10(e(i));
end
plot(x,y1,'ro-');                       % L2 Error Graph Plotting %
xlabel('Element Size');
ylabel('L2 Error');
title('Linear Element L2 Error');
figure;
plot(x,y2,'bo-');                       % Energy Error Graph Plotting %
xlabel('Element Size');
ylabel('Energy Error');
title('Linear Element Energy Error');

Convergence_L2=(y1(4)-y1(3))/(x(4)-x(3))% Calculates rate of Convergence for ↙
displacement%
Convergence_EE=(y2(4)-y2(3))/(x(4)-x(3))% Calculates rate of Convergence for energy%

end

function [L2,EE]=solve_fem(nn)

% Inputs %
N=nn;
L=0.5;                                 % Length of Al Rod %
Le=L/(N-1);                            % Calculation of Element Length %

% Connectivity and Nodes %
connect=[1:N-1;2:N];
nodes=[0:(L/(N-1)):L];

Ae=(pi/4)*0.01*0.01*70e9;              % Area * Elastic Modulus %

% Stiffness Matrix Calculation %
tic;                                   % Time Start %
k=Ae/Le;                               % Individual Element Stiffness (N/m) %
K=zeros(N);
for i=2:N
    K(i,i-1)=(-1)*k;
    K(i-1,i)=(-1)*k;
    K(i,i)=2*k;
end
```

```
K(1,1)=k;
K(N,N)=k;
toc;                                    % Time Stop %

% Apply Traction and Body Force %
tic;                                    % Time Start %
f=zeros(N,1);
for c=connect
    Xe=nodes(c);
    le=Xe(2)-Xe(1);
for q=[-1,1]/sqrt(3)
    n=0.5*[1-q,1+q];
    x=n*Xe';
    f(c)=f(c)+n'*2000*(sin(10*pi*x))*(le/2);
end
end
f(N)=f(N)-((pi/4)*0.01*0.01*1000000);   % Traction Force %
f(1)=0;                                 % body force at x=0 %

% Boundary Conditions %
K(1,2)=0;                               % Boundary Condition at x=0, u=0 %
K(2,1)=0;

% Solution %
u=zeros(N,1);
u=K\f;
toc;                                    % Stop Time %

% Calculation of new node positions %
new_nodes=zeros(N,1);
for i=1:N
    new_nodes(i,1)=nodes(i)+u(i,1);
end

% Plotting the Displacement Graphs %
figure;
plot(nodes,u,'ro');                     % Plotting graph for Displacement Field %
xlabel('X distance');
ylabel('Displacement U');
title('Displacement Field against X Distance');
legend('Displacement Field',N);

% Elements lengths and strain calculations %

new_length=zeros(N-1,1);
for i=1:N-1
    new_length(i,1)=new_nodes(i+1,1)-new_nodes(i,1);
    strain(i,1)=((new_length(i,1)-Le)/Le);
end

% Plotting the Strain Graphs %
figure;
plot(nodes(1:N-1),strain(1:N-1),'b');
title('Strain Field against X Distance'); % Plotting Graphs for Strain %
legend('Strain Field FEM');
```

```
xlabel('X distance');
ylabel('Strain Value');

% Calculation of L2 and Energy Error Norm %

L2_error=0;
EE_error=0;
E=70e9;
A=(pi/4)*0.01*0.01;

for c=connect
    Xe=nodes(c);
    le=Xe(2)-Xe(1);
    w=[5/9 8/9 5/9];                  % Weight Functions for 3 point Gauss Quadrature ↙
%
    mm=1;
    for q=[-1,0,1]/sqrt(3/5)          % 3 point Gauss Quadrature %
        n=0.5*[1-q,1+q];              % Shape Function %
        B=(1/le)*[-1,1];              % Derivative of the Shape Function %
        x=n*Xe';
        u_h=n*u(c);
        strain_h=B*u(c);
        exact_u=((20*sin(10*pi*x))/(E*A*(pi^2))+(((200/(pi*A*E))-(1e6/E))*x));
        exact_strain=(((200*cos(10*pi*x))/(E*A*pi))+((200/(pi*A*E))-(1e6/E)));
        L2_error = L2_error + ((exact_u-u_h)^2)*(le/2)*w(mm);
        EE_error = EE_error + (0.5*70e9*(exact_strain-strain_h)^2)*(le/2)*w(mm);
        mm=mm+1;
    end
end

L2=sqrt(L2_error);                    % Final Value of L2_Error %
EE=sqrt(EE_error);                    % Final Value of Energy Error %

end
```