NAME : ADITYA PRATAP SINGH RAJAWAT ROLL NO:150053

MATLAB CODE
ASSEMBLY CODE

assem.m

```
% assembly

local; % defined local elemental matrices
read_params; % set parameters in read_params file

LM = zeros(a,e); % LM array

for i= 1:a
    for j=1:e
        LM(i,j) = j + i - 1;
    end
end

% Assemble Stiffness Matrices
K_global = zeros(e+1);
% used K_global as per node points and not wrt to no. of eqn
% Could've used K _global accordingly
% After substituting boundary condition, the K_global could be solved

for i=1:e
    for j=1:a
        for k=1:a
            K_global(LM(j,i),LM(k,i)) = K_global(LM(j,i),LM(k,i)) + K_local(j,k);
        end
    end
end

% Assemble Force Matrix
F_global = zeros(e+1,1);

for i=1:e
    F_global(LM(1,i)) = F_global(LM(1,i)) + (q*h/6)*( 2*(LM(1,i)-1)*h + (LM(2,i)-1)*h );
    F_global(LM(2,i)) = F_global(LM(2,i)) + (q*h/6)*( (LM(1,i)-1)*h + 2*(LM(2,i)-1)*h );
end

% Get displacement field
% K_global*d = F_global
% Thomas algorithm - Used for sparse matrices : Tridiagonal K_global

d = thomasalgo(K_global,F_global); % solution to nodal displacements
% else would have removed last row and column to make K non-singular
% if size of K is reduced, then direct inverse can be taken
x = 0:h:1;
figure(1)
plot(x,d,'-o','LineWidth',1.5)
xlabel('x');
ylabel('d (displacement)')
title('Displacement field ')
% slope from forward difference : will give exact answer since linear interpolation
du_h = (1/h)*(d(2:e+1)-d(1:e));

% exact solution
u = @(y)q*(1 - y^3)/6;
du = @(y)(-q*y^2/2); % derivative of exact solution

error = zeros(1,e); % initialised re_x
```

NAME : ADITYA PRATAP SINGH RAJAWAT ROLL NO:150053

```
p = zeros(1,e); % position where re_x are evaluated

for i=1:e
    p(i) = (2*i-1)*h/2;
    % error array is re_x at midpoint of the element
    error(i) = abs(du(p(i))-du_h(i))/(q/2);
end

% rat(error) for rational form of re_x
% Obtained re_x, for all values of h or no. of elements

 %% re_x plot

h_2 = [ 1 0.5 0.25 1/10 1/50 1/100];
error_2 = [1/12 1/48 1/192 1/1200 1/30000 1/120000];
figure(2)
plot(log(h_2),log(error_2),'-o','LineWidth',2);
xlabel('ln(h)');
ylabel('ln(re_x)');
title('ln(re_x) vs ln(h)');

slope;
```

LOCAL
local.m

```
clear all;
% local
read_params;

% local Stiffness matrix
K_local = e*[ 1 -1 ; -1 1];

% local Force matrix
% f(x) = q*x
% F_local = (q/(6*e))*[ 2*x1 + x2 ; x1 + 2*x2];
```

PARAMETERS
read_params.m

```
% Parameters

a = 2;
e = 4 ; % number of elements
% linear elements : nodes = e+1
h = 1/e; % uniform mesh grid
q = 16; % can vary or define as symbolic variable
```

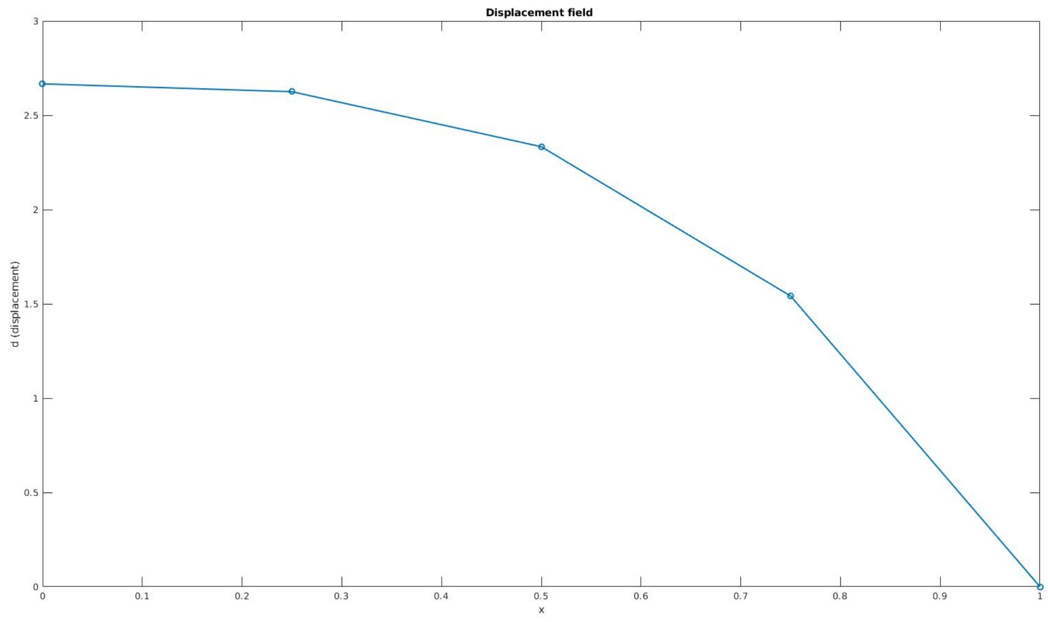
THOMAS ALGORITHM

```
%% thomasalgo algorithm
function d = thomasalgo(K,F)

p = length(F);
d = zeros(1,p);

a = zeros(1,p);
b = zeros(1,p);
c = zeros(1,p);
```

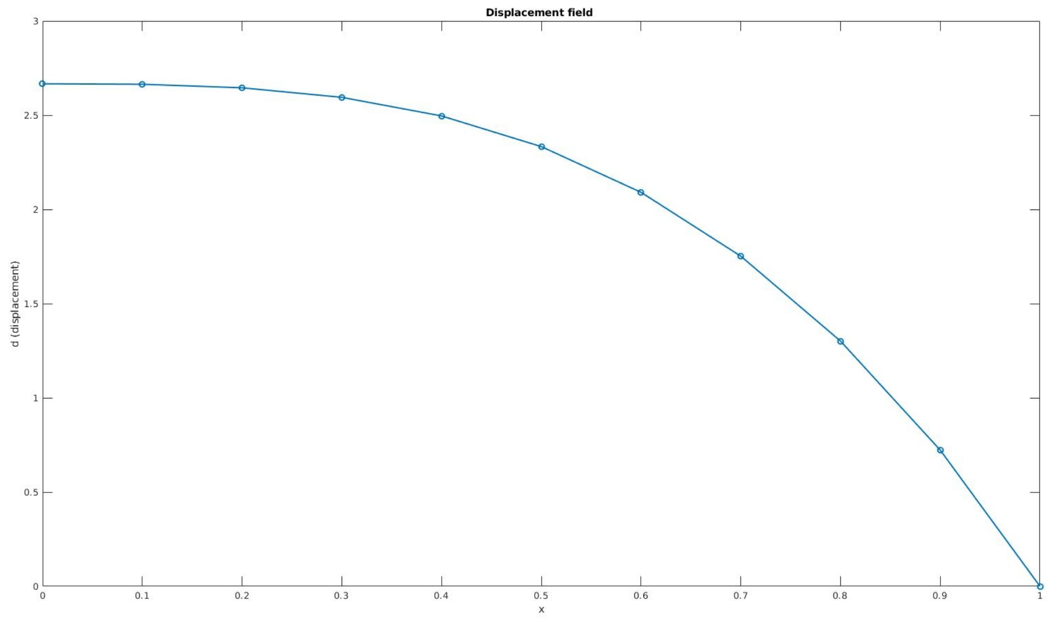
NAME : ADITYA PRATAP SINGH RAJAWAT ROLL NO:150053

```
for i = 1:p
    if i>1
    a(i) = K(i,i-1);
    end
    b(i) = K(i,i);
    if i<p
    c(i) = K(i,i+1);
    end
end

for i = 2:p
        f = a(i)/b(i-1);
        b(i) = b(i)-c(i-1)*f;
        F(i) = F(i)-F(i-1)*f;
end

d(p) = 0; % Boundary condition at right end

for i = (p-1):-1:1
       d(i) = (F(i)-c(i)*d(i+1))/b(i);
end

end


% Plot for slope for 'e' elements
x = 0:0.01:1;
du_h_1 = zeros(1,101);
for i=1:101
    for j=1:e
      if(x(i)<=j*h)
         du_h_1(i)= du_h(j);
         break;
      end
    end
end
plot(x,du_h_1,'LineWidth',2);
xlabel('x');
ylabel('slope (du\_h)');
title('Slope of displacement field');
```

NAME : ADITYA PRATAP SINGH RAJAWAT ROLL NO:150053

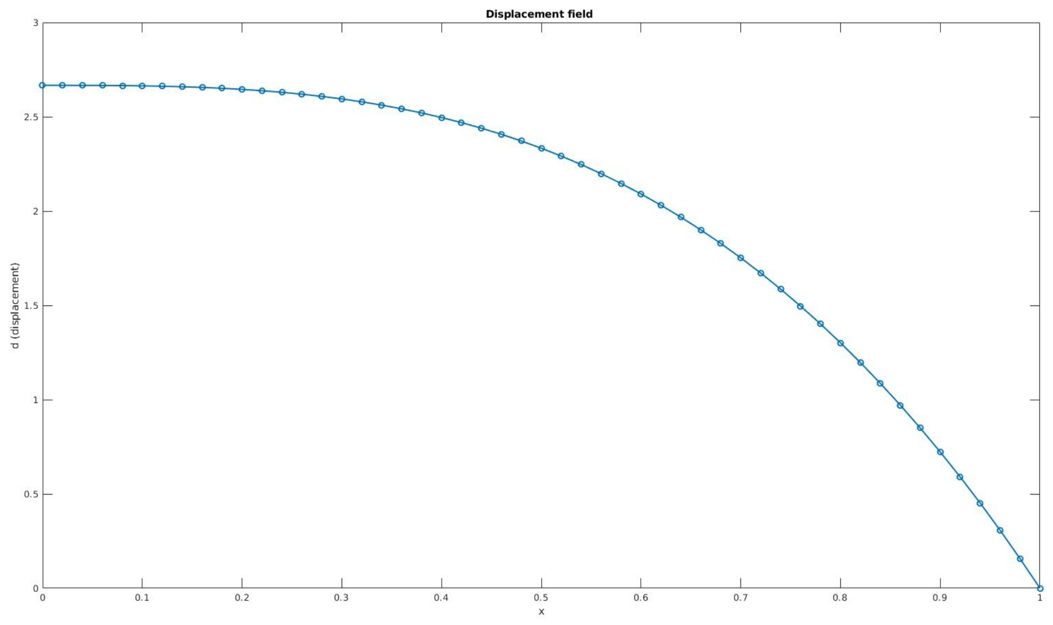


e = 4

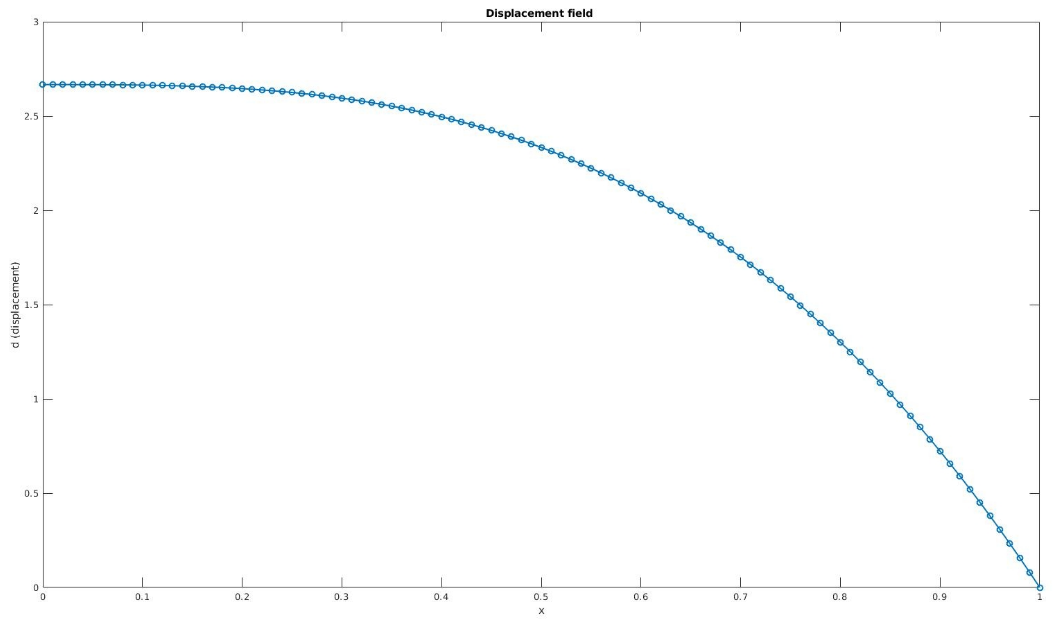


e = 10

NAME : ADITYA PRATAP SINGH RAJAWAT ROLL NO:150053

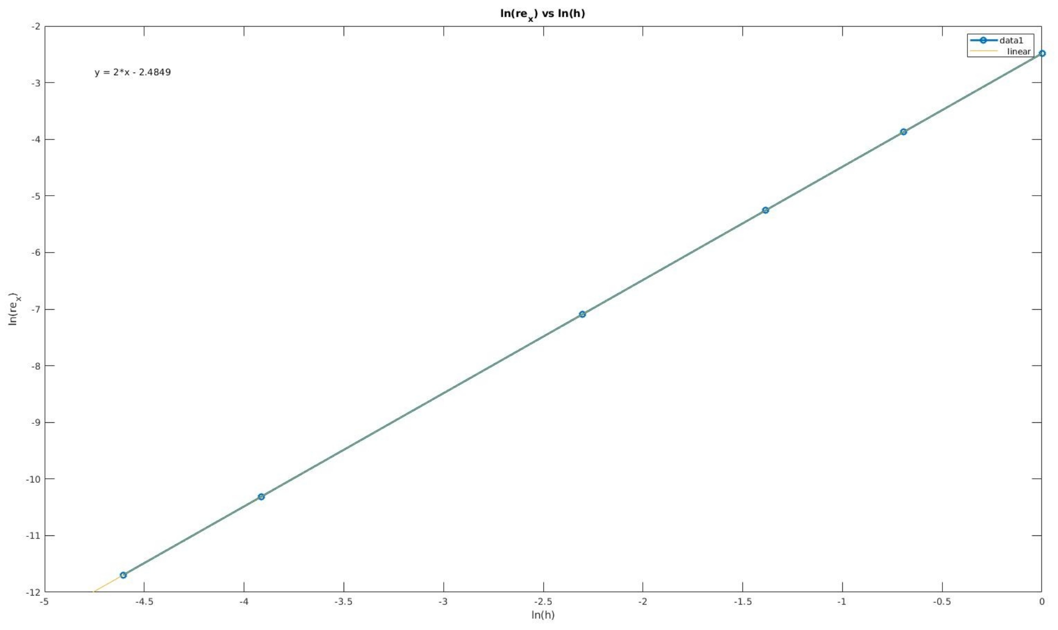


e = 50

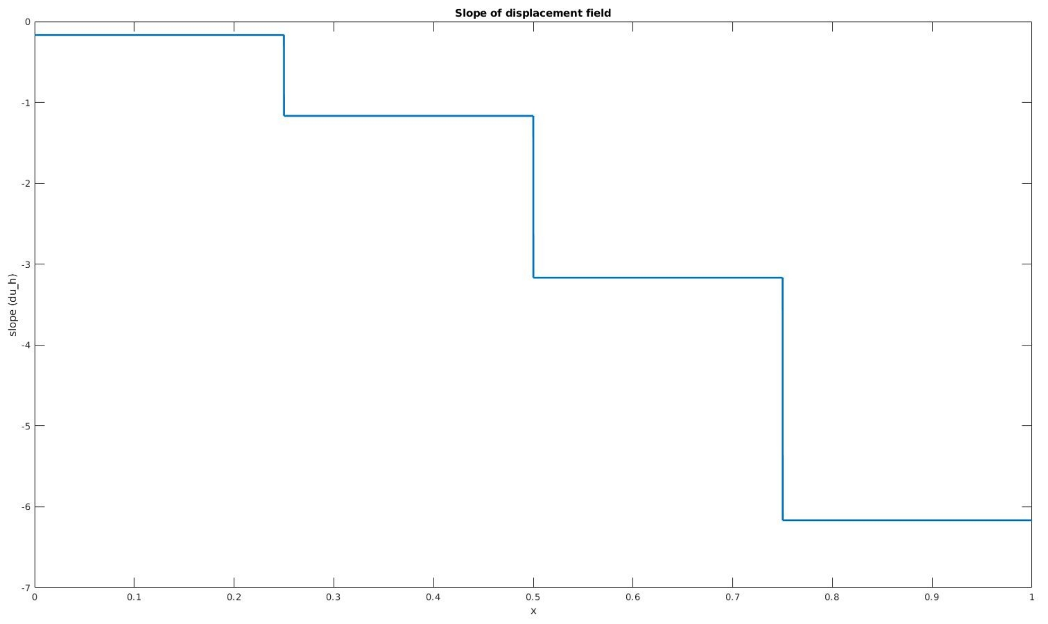


e = 100

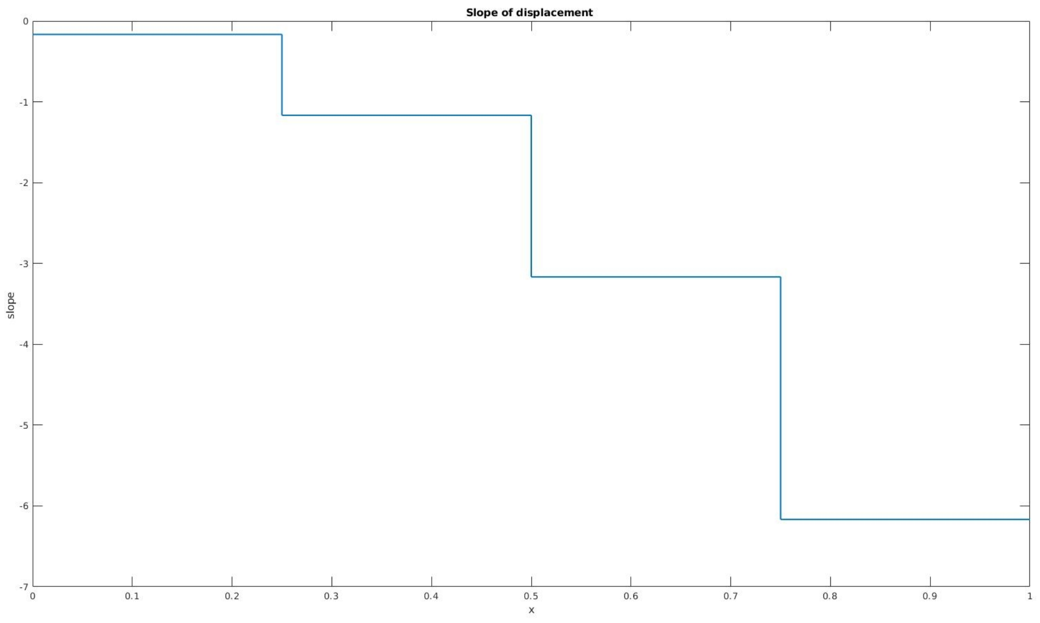
NAME : ADITYA PRATAP SINGH RAJAWAT ROLL NO:150053



ln(re_x) vs. ln(h)



Slope of displacement field (e=4)


Slope of displacement
slope
x
0
-1
-2
-3
-4
-5
-6
-7
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1