CH20B012

Anish Anand Pophale

AM5450 Assignment 4

Q1) $\frac{d}{dx}\left(EA\frac{du}{dx}\right) + q(x) = 0, \quad 0 \leq x \leq 2$

$u(0) = 0, \quad EA_L \frac{du}{dx}\Big|_{x=L} = P$

Assuming linear variation of A with x,

$A(x) = A_0 + \frac{x}{L}(A_L - A_0)$

As E is constant, we can write

$E\frac{d}{dx}\left(A(x)\frac{du}{dx}\right) + q(x) = 0$

$q(x) = cx$

$\therefore E\frac{d}{dx}\left(A\frac{du}{dx}\right) + cx = 0$

Domain residual $R_D = E\frac{d}{dx}\left(A\frac{du}{dx}\right) + cx$

Boundary residual $R_B = EA_L\frac{du}{dx}\Big|_{x=L} - P$

weight function $w_i = \frac{\partial u}{\partial a_i}$

We can non dimensionalize the total residual R_T considering the units of R_D and R_B as

$R_T = \frac{R_D}{CL} + \frac{R_B}{P}$

This is solved in MATLAB and the code along with the results are given below.

Q1)

MATLAB Code:

```
clc; clear all; close all; format compact; format shortg;

tic
syms x
L = 2; x0 = 0; xL = L;
xvec = x0:0.01:xL;

figure(1); hold on;
figure(2); hold on;
figure(3); hold on;

N = 1;
Rt_mean = 1;
while Rt_mean>10^-3
    N = N+1;
    [u,uprime,Rt] = galerkin(N);

    %Plotting only for polynomials above order 15
    if(N>15)
        figure(1)
        plot(xvec,subs(u,x,xvec),'-', 'DisplayName', num2str(N));

        figure(2)
        plot(xvec,subs(uprime,x,xvec),'-', 'DisplayName', num2str(N));

        figure(3)
        plot(xvec,subs(Rt,x,xvec),'-', 'DisplayName', num2str(N));
    end

    Rt_mean = mean(abs(subs(Rt,x,xvec)));
    Rt_vec(N-1) = Rt_mean;
end

figure(1); legend("show"); xlabel('x'); ylabel('u(x)'); title('u(x) vs x using
Galerkin Method')
figure(2); legend("show"); xlabel('x'); ylabel('u''(x)'); title('u''(x) vs x using
Galerkin Method')
figure(3); legend("show"); xlabel('x'); ylabel('Residual'); title('e(x) vs x using
Galerkin Method')

%Plotting the residual vs order of trial function
figure(4)
semilogy(2:1:N,Rt_vec,'-o')
xlabel("Order of polynomial")
ylabel("Total Residue")
title("Total Residue vs Order of trial function ")
toc

function [u,uprime,Rt] = galerkin(N)
    syms x
    a = sym('a', [1,N+1]);
    b = zeros(1,N+1);

    %Constants
    L = 2; x0 = 0; xL = L;
```

```
    E = 200*10^9;
    A0 = 3.1416*10^-4; AL = 1.9635*10^-5;
    P = 2*10^3;
    c = 10^3;

    %Linear variation of area
    A = A0 + (x/L)*(AL-A0);

    %Trial Function
    u = poly2sym(flip(a));
    dudx = diff(u,x,1);

    %Essential BC
    BC1 = subs(u,x,x0) == 0;
    b(1) = solve(BC1,a(1));
    u = subs(u,a(1),b(1));

    %Domain Residual
    Rd = diff(E*A*dudx,x,1) + c*x;
    %Boundary Residual
    Rb = E*AL*subs(dudx,x,xL) - P;

    %Weighted Residual Equations
    eq = sym(zeros(N,1));
    for i = 2:N+1
        w = diff(u,a(i),1);
        wb = subs(diff(u,a(i),1),x,xL);
        eq(i-1) = int(w*Rd,x,x0,xL) + wb*Rb == 0;
    end

    sol = solve(eq,a(2:end));
    for i = 2:N+1
        b(i) = sol.(['a',num2str(i)]);
    end

    u = subs(u,a,b); %Displacement
    uprime = diff(u,x,1); %Strain

    %Total Residual (Normalized)
    Rt = subs(Rd,a,b)/(c*L) + subs(Rb,a,b)/P;
end
```

- Time taken for code to converge = **41.528738 seconds**.
- Residual limit used is 10^{-3} as it was taking very long to converge to 10^{-6}.
- For very high order polynomials, the residual decreases but there are large oscillations observed in the trial function.
- The residual is non dimensionalized using the given variables considering the units of Rd and Rb. Without this we do not know how low the tolerance of the residual should be as it has some units, hence it is better to consider a dimensionless residual function.
- The minimum order of the trial function for the residual to go below the limit is **Np = 21** where the mean total residual over the domain is $\mathbf{Rt_{mean} = 8.6448*10^{-4}}$.

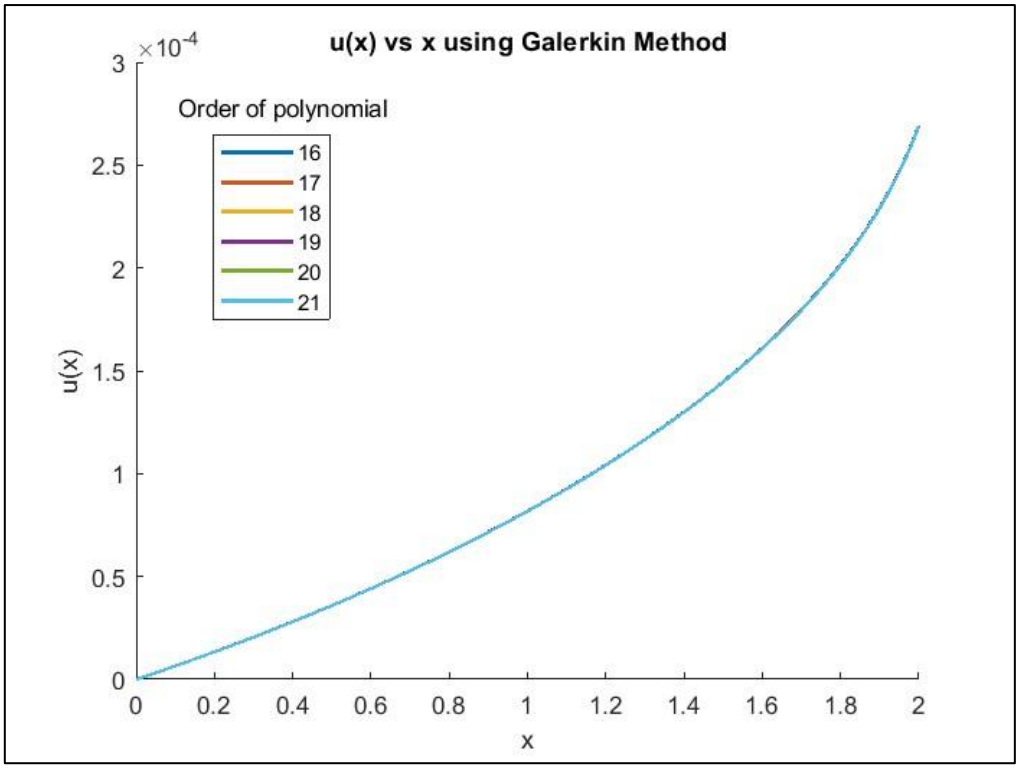


Fig 1: u(x) vs x for different order trial functions using the Galerkin method

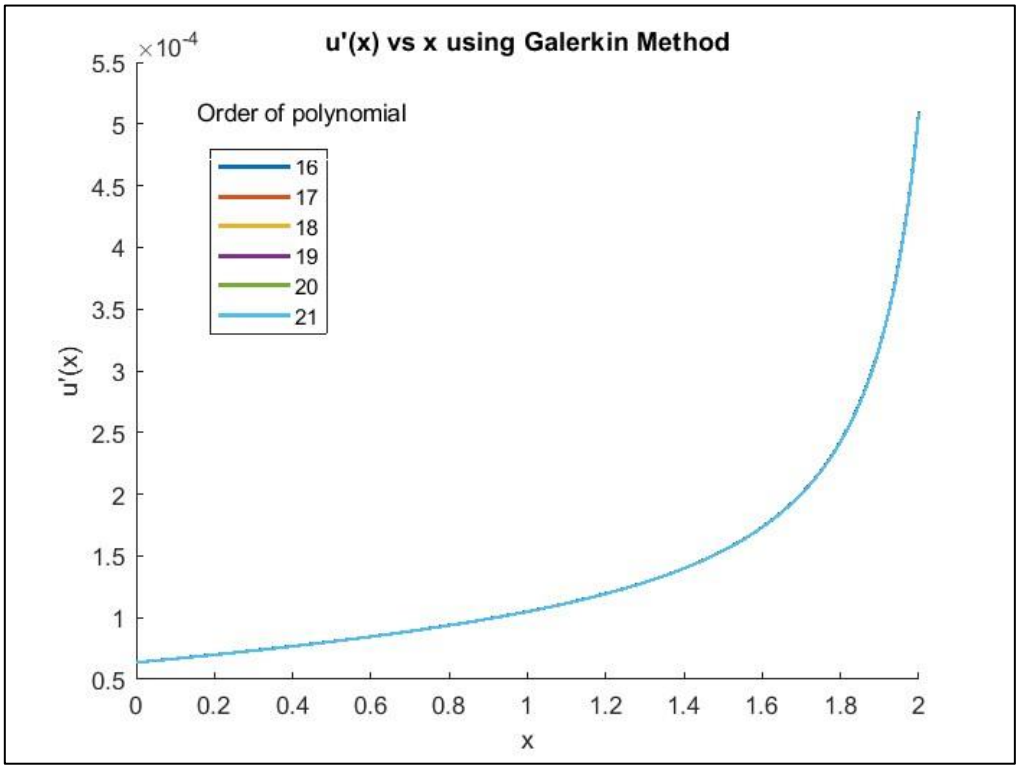


Fig 2: u’(x) vs x for different order trial functions using the Galerkin method

- The displacement, strain and the residual are plotted only for polynomials with order greater than 15 so that the plots are not overcrowded.
- Because of this, the difference between u and u' for different polynomials is not visible as the values are very close to each other.
- The difference can be observed in the residual vs x plot given below.

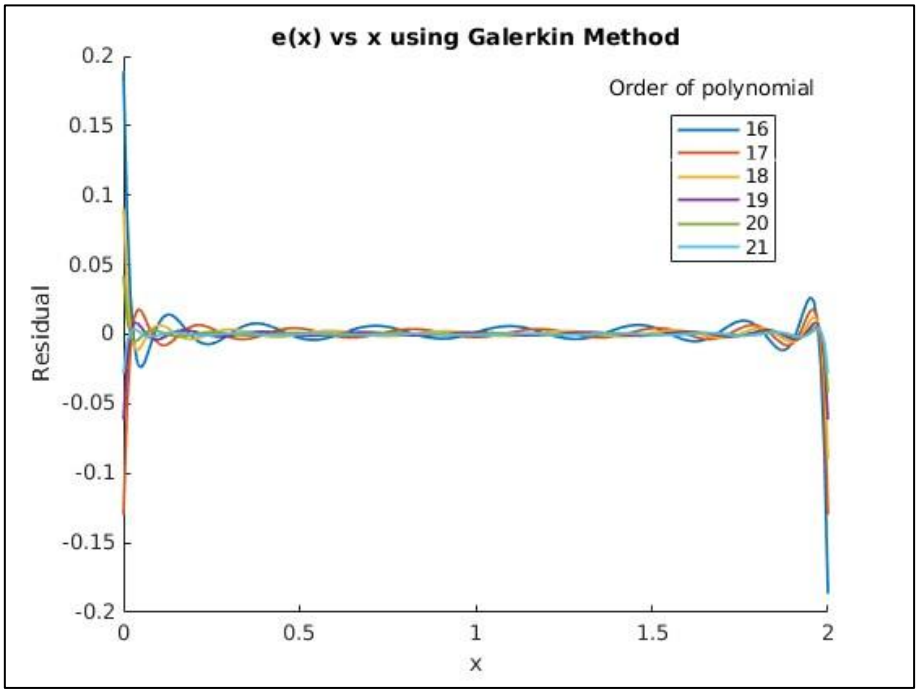


Fig 3: e(x) vs x for different order trial functions using the Galerkin method

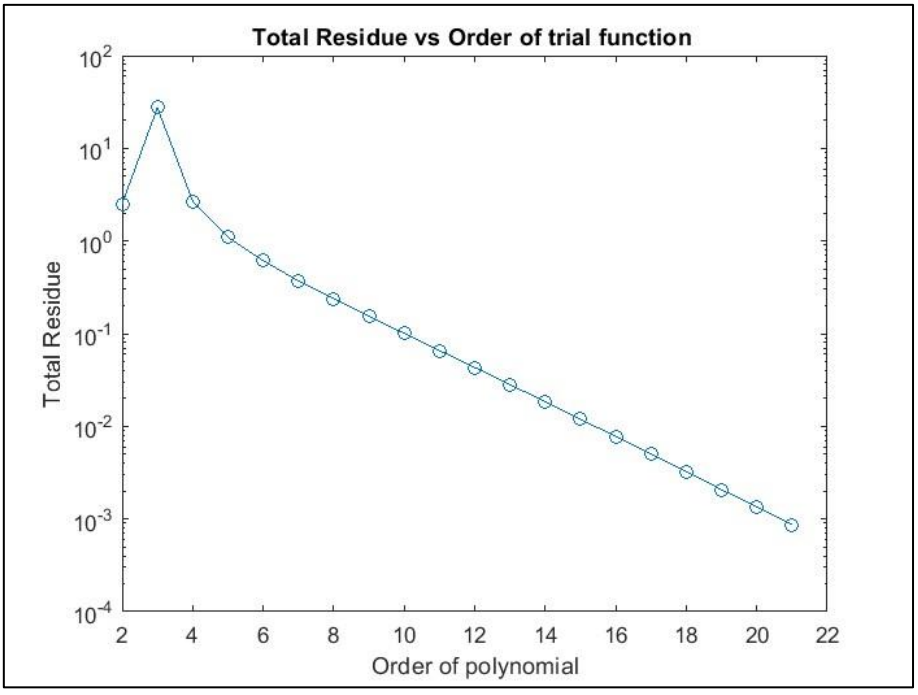


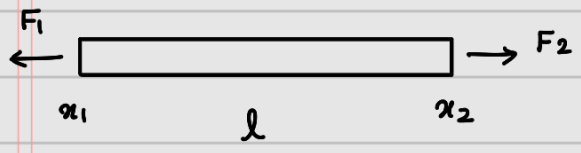
Fig 4: Residual vs Order of trial function used

Q2) Consider the element equation using MGM -

$$\int_{x_1}^{x_2} E \frac{d}{dx}\left(A \frac{du}{dx}\right) w_i^{\circ} \, dx + \int_{x_1}^{x_2} cx \, w_i^{\circ} \, dx = 0$$

$$\therefore \int_{x_1}^{x_2} -E A(x) \frac{du}{dx} \frac{dw_i^{\circ}}{dx} \, dx + \int_{x_1}^{x_2} cx \, w_i^{\circ} \, dx + \left(w_i^{\circ} EA \frac{du}{dx}\right)_{x_1}^{x_2}$$

Consider the element as follows



$$EA \frac{du}{dx} = F$$

$$\therefore \int_{x_1}^{x_2} -EA \frac{du}{dx} \frac{dw}{dx} \, dx + \int_{x_1}^{x_2} cx \, w_i^{\circ} \, dx + w_1 u_1' + w_2 u' = 0$$

Assume $u = u_1 N_1 + u_2 N_2$

we know that $N_1 = \frac{x_2 - x}{l}$ and $N_2 = -\frac{x_1 + x}{l}$

Consider the mapping of x to s s.t $[x_1, x_2] \rightarrow [-1, 1]$

$$\therefore (s-1) = \frac{1-(-1)}{x_2 - x_1} (x - x_2)$$

$$\therefore (s-1)(x_2 - x_1) = 2x - 2x_2$$

$$\therefore s(x_2 - x_1) - x_2 + x_1 = 2x - 2x_2$$

$$\therefore sl = 2x - x_1 - x_2$$

$$\therefore s = \frac{2x - x_1 - x_2}{l} \quad \text{and} \quad x = \frac{sl + x_1 + x_2}{2}$$

$$\therefore N_1(s) = \frac{x_2}{l} - \left(\frac{sl + x_1 + x_2}{2l}\right) = -\frac{sl + x_2 - x_1}{2l} = \frac{l - sl}{2l}$$

$$\therefore N_1(s) = \frac{1-s}{2}$$

$$N_2(s) = -\frac{x_1}{l} + \frac{(ls + x_1 + x_2)}{2l} = \frac{ls + x_2 - x_1}{2l} = \frac{ls + l}{2l}$$

$$\therefore N_2(s) = \frac{1+s}{2}$$

$$\therefore u = N_1 u_1 + N_2 u_2$$

$$= \left(\frac{1-s}{2}\right) u_1 + \left(\frac{1+s}{2}\right) u_2$$

$$= N^T d \quad \text{where,} \quad N^T = \begin{bmatrix} \frac{1-s}{2} & \frac{1+s}{2} \end{bmatrix} \text{ and } d = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Now, $ds = \frac{2\,dx}{l}$, $\therefore dx = \frac{l}{2}\,ds$

$$w_i^\circ = \frac{\partial u}{\partial u_i^\circ} = N_i^\circ = \begin{bmatrix} (1-s)/2 \\ (1+s)/2 \end{bmatrix}$$

$$\frac{du}{dx} = \frac{du}{ds} \times \frac{ds}{dx} = \frac{2}{l} \begin{bmatrix} -1/2 & 1/2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = B^T d$$

$$\therefore B^T = \begin{bmatrix} -1/l & 1/l \end{bmatrix}$$

$$\frac{dw}{dx} = \frac{dw}{ds} \times \frac{ds}{dx} = \frac{2}{l} \begin{bmatrix} -1/2 & 1/2 \end{bmatrix} = \begin{bmatrix} -1/l & 1/l \end{bmatrix} = B$$

Consider

$$\therefore \int_{x_1}^{x_2} -EA \frac{du}{dx} \frac{dw}{dx}\,dx + \int_{x_1}^{x_2} cx\, w_i^\circ\,dx + w_1 u_1' + w_2 u' = 0$$

For an element, we can take $A = A_{mean} = \frac{A(x_1) + A(x_2)}{2}$,

$$A(x) = A_0 + \frac{x}{L}(A_L - A_0)$$

$\therefore A_{mean}^\circ$ is constant for an element

$$w_1 = w(x = x_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$w_2 = w(x = x_2) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

∴ We get,

$$EA_{mean} \int_{x_1}^{x_2} -\frac{du}{dx}\frac{dw_i^\circ}{dx}\,dx + \int_{x_1}^{x_2} c x\, w_i^\circ\, dx + \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = 0$$

Substituting s in place of x in all terms,

$$\int_{x_1}^{x_2} \frac{du}{dx}\frac{dw_i}{dx}\,dx = \int_{-1}^{1} B^T d\, B \frac{l}{2}\,ds = \frac{dl}{2}\int_{-1}^{1} B^T B\,ds$$

$$\int_{-1}^{1} B^T B\,ds = \int_{-1}^{1} \begin{bmatrix} -1/l & 1/l \end{bmatrix}\begin{bmatrix} -1/l \\ 1/l \end{bmatrix} ds = \frac{1}{l^2}\int_{-1}^{1}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} ds$$

$$= \frac{2}{l^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$\therefore EA_{mean} \int_{x_1}^{x_2} \frac{du}{dx}\frac{dw_i}{dx}\,dx = \frac{EA_{mean}}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} d$$

Consider the term,

$$\int_{x_1}^{x_2} c x\, w_i^\circ\, dx = \int_{-1}^{1} c\left(\frac{ls + x_1 + x_2}{2}\right)\begin{bmatrix} (1-s)/2 \\ (1+s)/2 \end{bmatrix} ds$$

$$= \begin{bmatrix} l(x_1 + x_2 - l/3) \\ l(x_1 + x_2 + l/3) \end{bmatrix}$$

∴ The element equation becomes,

$$-\frac{EA_{mean}}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} l(x_1 + x_2 - l/3) \\ l(x_1 + x_2 - l/3) \end{bmatrix} + \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = 0$$

$$\therefore \frac{EA_{mean}}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} l(x_1 + x_2 - l/3) \\ l(x_1 + x_2 - l/3) \end{bmatrix} + \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$$

$$\therefore kd = r_q + r_\beta$$

When the global matrices are assembled, the internal forces F_i get cancelled and only the reaction force at $x = 0$ (which we do not solve for) and the point load P at $x = L$ remain. (NBC)

Q2)

MATLAB Code:

```
clc; clear all; close all; format compact; format shortg;

tic
% Constants
syms x;
x0 = 0; xL = 2; L = 2;
A0 = 3.1416e-4; AL = 1.9635e-5;
A = A0 + (x/L)*(AL-A0);
E = 200e9;
P = 2e3;
c = 1e3;

%Starting with 2 elements
Ne = 2;
[u_old, uprime_old, xvec_old] = element(Ne);

Re1 = 1;
Re2 = 1;
Re1_vec = [];
Re2_vec = [];
Re3_vec = [];
Re4_vec = [];
while (Re1 > 1e-3 && Re2 > 1e-3)
    Ne = Ne + 1;
    [u_new, uprime_new, xvec_new] = element(Ne);

    % Relative error in u at bar tip
    Re1 = abs((u_new(end) - u_old(end)) / u_new(end));
    Re1_vec(Ne-2) = Re1;

    % u at bar midpoint
    if mod(Ne, 2) == 0
        u_mid_old = u_old(Ne/2 + 1);
        u_mid_new = u_new(Ne/2 + 1);
    else %Linear interpolation
        u_mid_old = 0.5*(u_old((Ne+1)/2) + u_old((Ne+3)/2));
        u_mid_new = 0.5*(u_new((Ne+1)/2) + u_new((Ne+3)/2));
    end

    % Relative error in u at bar midpoint
    Re2 = abs((u_mid_new - u_mid_old) / u_mid_new);
    Re2_vec(Ne-2) = Re2;

    % Relative error in u' at bar tip
    Re3 = abs((uprime_new(end) - uprime_old(end)) / uprime_new(end));
    Re3_vec(Ne-2) = Re3;

    % u' at bar midpoint
    if mod(Ne, 2) == 0 %Linear interpolation
        uprime_mid_old = 0.5*(uprime_old(Ne/2) + uprime_old(Ne/2 + 1));
        uprime_mid_new = 0.5*(uprime_new(Ne/2) + uprime_new(Ne/2 + 1));
    else
        uprime_mid_old = uprime_old((Ne+1)/2);
        uprime_mid_new = uprime_new((Ne+1)/2);
    end
```

```
    %Relative error in u' at bar midpoint
    Re4 = abs((uprime_mid_new - uprime_mid_old) / uprime_mid_new);
    Re4_vec(Ne-2) = Re4;

    u_old = u_new;
    uprime_old = uprime_new;
    xvec_old = xvec_new;
end

fprintf('Relative Error in u at tip: %.5e with %d elements\n', Re1, Ne);
fprintf('Relative Error in u at midpoint: %.5e with %d elements\n', Re2, Ne);
fprintf('Relative Error in u'' at tip: %.5e with %d elements\n', Re3, Ne);
fprintf('Relative Error in u'' at midpoint: %.5e with %d elements\n', Re4, Ne);

figure(1);
hold on
plot(3:1:Ne,Re1_vec,'-o','DisplayName','u tip','LineWidth', 1.3)
plot(3:1:Ne,Re2_vec,'-o','DisplayName','u mid','LineWidth', 1.3)
plot(3:1:Ne,Re3_vec,'-o','DisplayName','u'' tip','LineWidth', 1.3)
plot(3:1:Ne,Re4_vec,'-o','DisplayName','u'' mid','LineWidth', 1.3)
set(gca,"YScale","log")
xlim([3,Ne])
legend('show')
xlabel('Number of elements')
ylabel('Relative Error')
title('Relative Error in displacement and strain at bar tip and midpoint')

%Plotting for a few values of N
figure(2); hold on;
figure(3); hold on;
for N = 2:3:Ne
     [u, uprime, xvec] = element(N);

     figure(2)
     plot(xvec,u,'-', 'DisplayName', num2str(N), 'LineWidth', 1.3)

     figure(3)
     stairs(xvec,[uprime;uprime(end)],'-', 'DisplayName', num2str(N), 'LineWidth',
1.3)
end
figure(2); legend("show"); xlabel('x'); ylabel('u(x)'); title('u(x) vs x using
different number of elements')
figure(3); legend("show"); xlabel('x'); ylabel('u''(x)'); title('u''(x) vs x with
different number of elements')
toc

function [u, uprime, xvec] = element(Ne)
    syms x;
    x0 = 0; xL = 2; L = 2;
    A0 = 3.1416e-4; AL = 1.9635e-5;
    A = A0 + (x/L)*(AL-A0);
    E = 200e9;
    P = 2e3;
    c = 1e3;

    xvec = x0:L/Ne:xL; % Node locations

    % Initialize matrices
```

```
    K = zeros(Ne+1, Ne+1);
    rq = zeros(Ne+1, 1);
    rb = zeros(Ne+1, 1);
    rb(end) = P;

    for i = 1:Ne
        A1 = subs(A, x, xvec(i));
        A2 = subs(A, x, xvec(i+1));
        Am = (A1 + A2)/2;
        l = xvec(i+1) - xvec(i);

        K(i, i) = K(i, i) + E*Am/l;
        K(i, i+1) = K(i, i+1) - E*Am/l;
        K(i+1, i) = K(i+1, i) - E*Am/l;
        K(i+1, i+1) = K(i+1, i+1) + E*Am/l;

        rq(i) = rq(i) + 0.25*l*c*(xvec(i) + xvec(i+1) - l/3);
        rq(i+1) = rq(i+1) + 0.25*l*c*(xvec(i) + xvec(i+1) - l/3);
    end

    % Solve for displacement
    u = zeros(Ne+1, 1); %u1 = 0 from EBC so we do not solve for u1
    % Remove first row and column of K and first row in rq and rb as we are not
solving for u1
    u(2:end) = K(2:end, 2:end)\(rq(2:end) + rb(2:end));

    % Calculate strain
    uprime = zeros(Ne, 1);
    for i = 1:Ne
        l = xvec(i+1) - xvec(i);
        uprime(i) = (u(i+1) - u(i)) / l;
    end
end
```

- Time taken for convergence = **5.203251 seconds**.
- Here also, we take the convergence criteria as 10^{-3} at the bar tip or the bar midpoint.
- Once any one of these two criteria are satisfied, the solution is considered to be converged.
- In this case, the error at the bar midpoint reduced very slowly and takes a very long time to reach 10^{-6}, hence we take the error tolerance as 10^{-3} which is same as that for Q1.
- Number of linear elements required **N_{LE} = 18 elements**.

Code output:

Command Window

```
Relative Error in u at tip: 9.25671e-04 with 18 elements
Relative Error in u at midpoint: 7.64189e-02 with 18 elements
Relative Error in u' at tip: 1.41901e-02 with 18 elements
Relative Error in u' at midpoint: 3.66878e-02 with 18 elements
Elapsed time is 5.203251 seconds.
fx >>
```

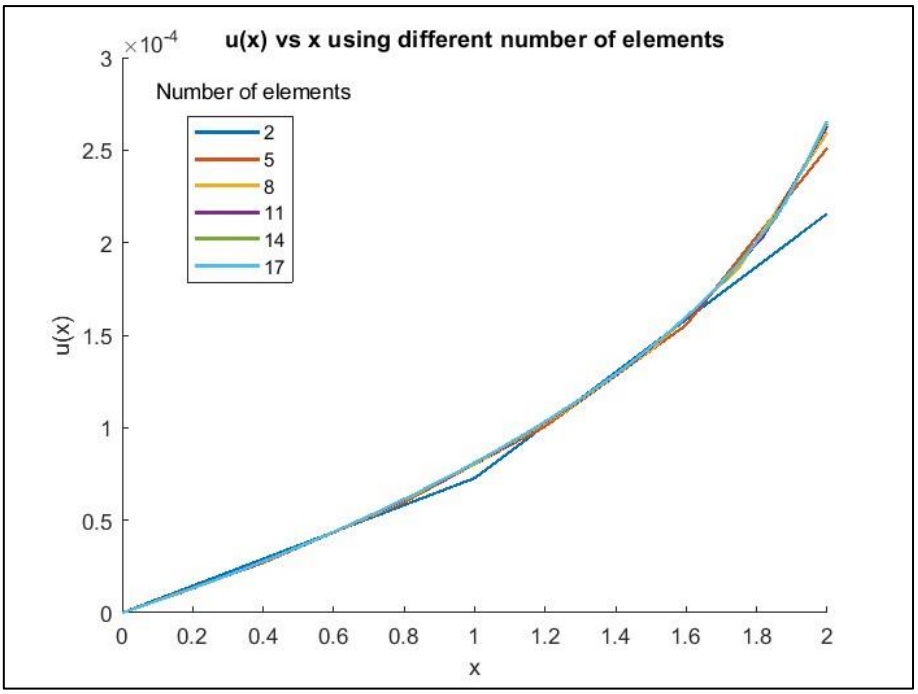


Fig 5: u(x) vs x for using different number of elements

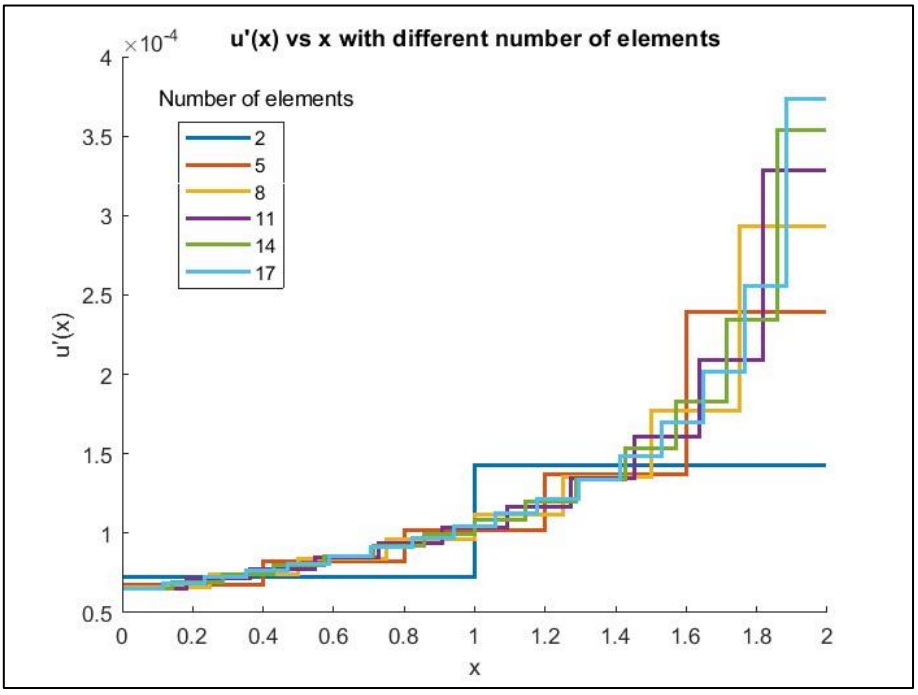


Fig 6: u’(x) vs x for using different number of elements

- The plot of the strain is similar to a step function as we have assumed a linear trial function for each element which implies that the strain in each element is a constant which leads to the discontinuity.

- Comparing the results with Q1, we can see that the displacement using higher order polynomials and more number of linear elements matches with each other.

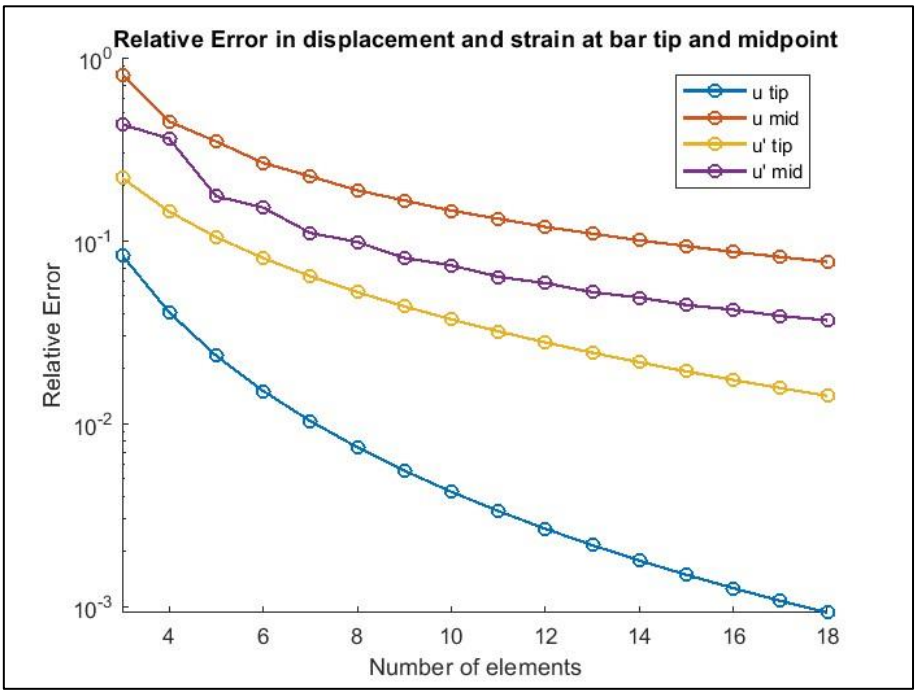


Fig 7: Relative error in u and u’ at the bar tip and midpoint

Comparison between the two approaches:

- For this case, when we use an error tolerance of 10^{-3} we can see that the h type method converges faster than the p type based on the time required for convergence.
- This need not be in general for any type of problem, in general any of the two methods or a mixed p-h type method can give faster convergence.
- The computational complexity for the p type method increases as the order of the polynomial increases, but for the h type method, we repeat the same, simple calculations for a linear element multiple times.
- The displacement is predicted well by both the methods even though for the h type method the variation of u vs x is not smooth and is made up of linear approximations.
- The strain however in the case of h type is a discontinuous function with jumps as the strain in each element is constant for a linear element, but for the p type method, the strain is well predicted and is smooth
- For this problem, we can say that the h type method converges faster, but the p type method is more accurate.
- A mixed p-h method with higher order elements may give a better result, both in terms of an acceptable convergence time as well as accuracy.