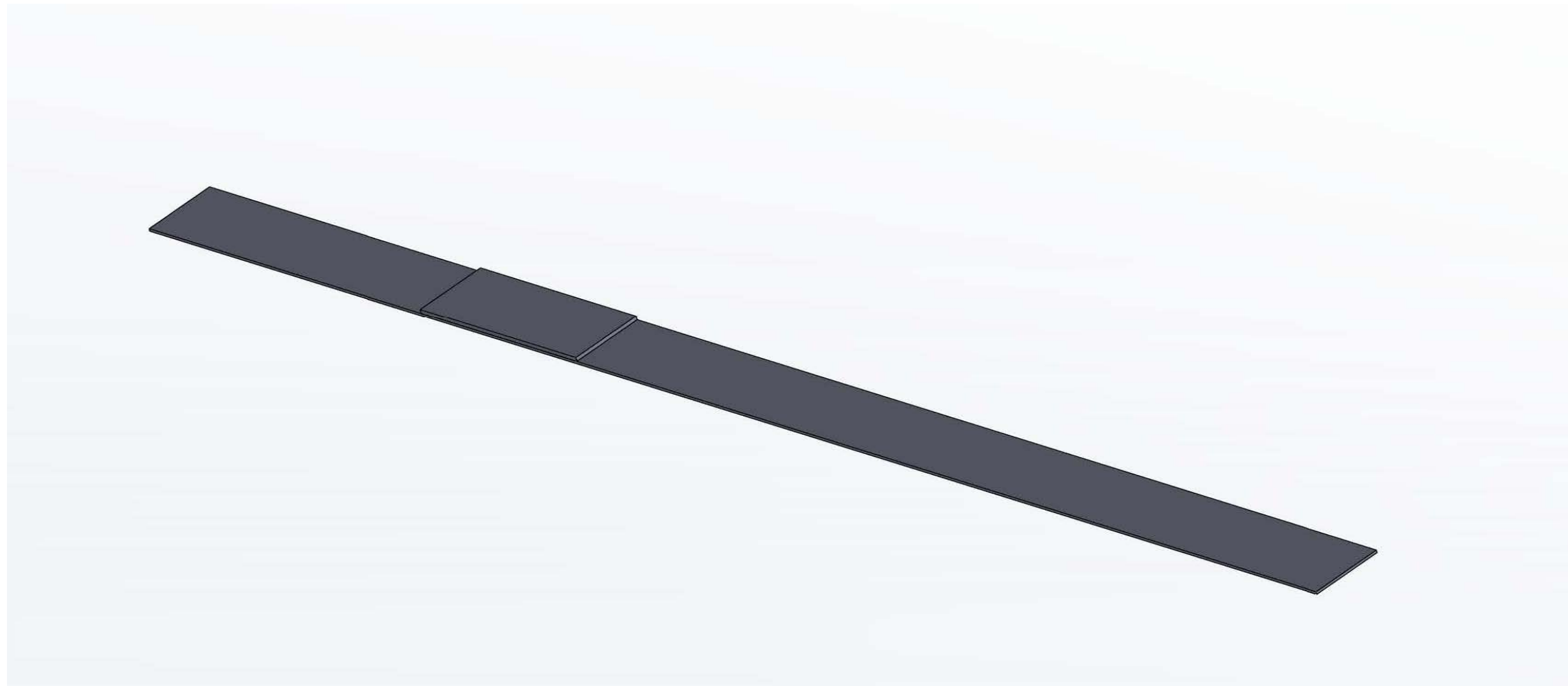


## Structural Vibration Analysis

**Determining the Natural frequencies of the Composite beam having a patch of Piezo-electric layer using;**

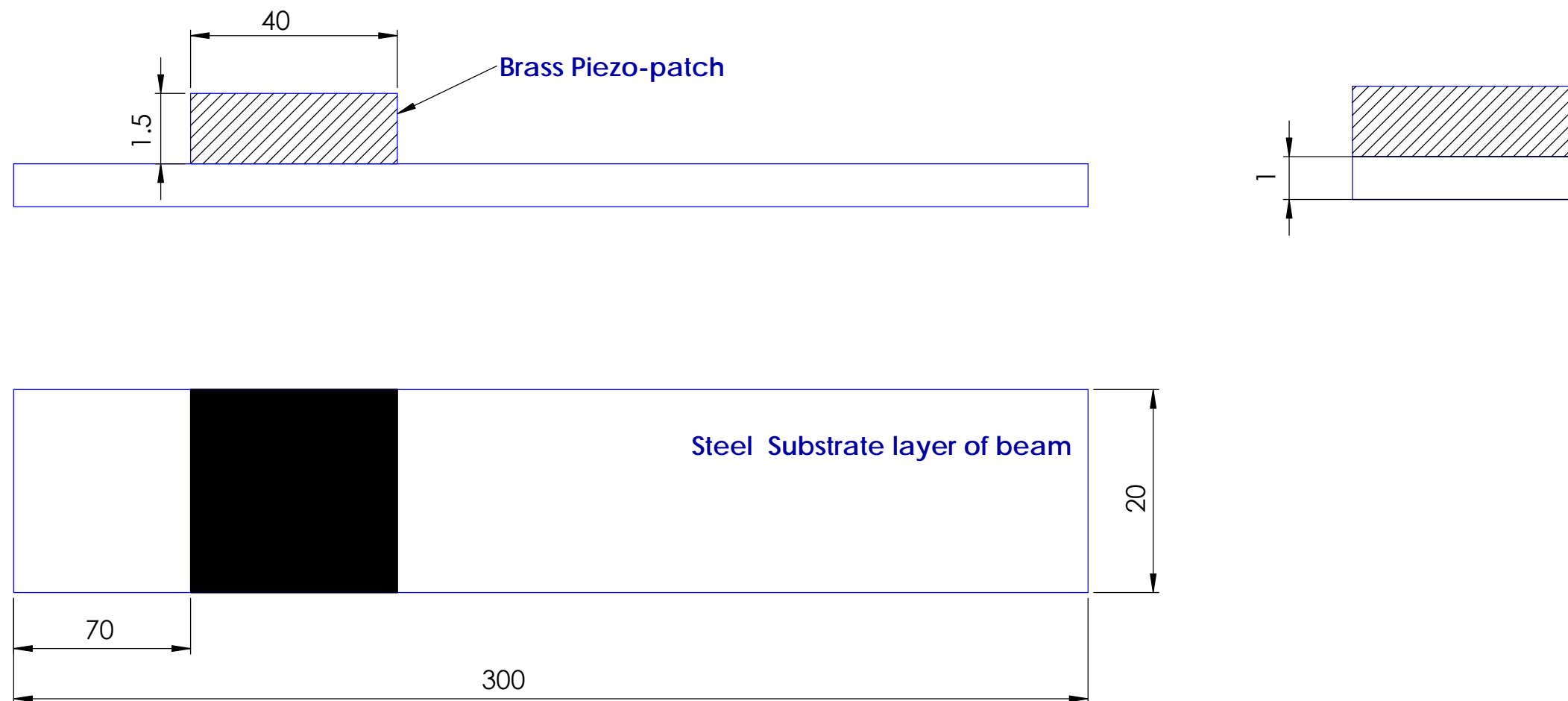
*Transfer-Matrix Method,  
Rayleigh-Ritz Approximation method, &  
ANSYS Modal analysis*



## Outline of Composite beam (not to scale).

All dimensions are in [mm]

We have been presented with a beam which has clamped-pinned boundary conditions.  
We need to find first three natural frequencies of this composite beam.



**The beam is discretized into three parts -**

- 1.) 7 cm from the left end is the first segment
- 2.) 4 cm from the previous 7 cm segment is the second segment
- 3.) 19 cm from the right end is the third one.

- The beam material known as substrate has been made of Steel, while the Piezo-patch attached on the surface is composed of Brass. The following are the engineering properties of the two material:

	<i>Young's Modulus [GPa]</i>	<i>Density [kg/m<sup>3</sup>]</i>
<i>Brass</i>	120	8600
<i>Steel</i>	200	7800

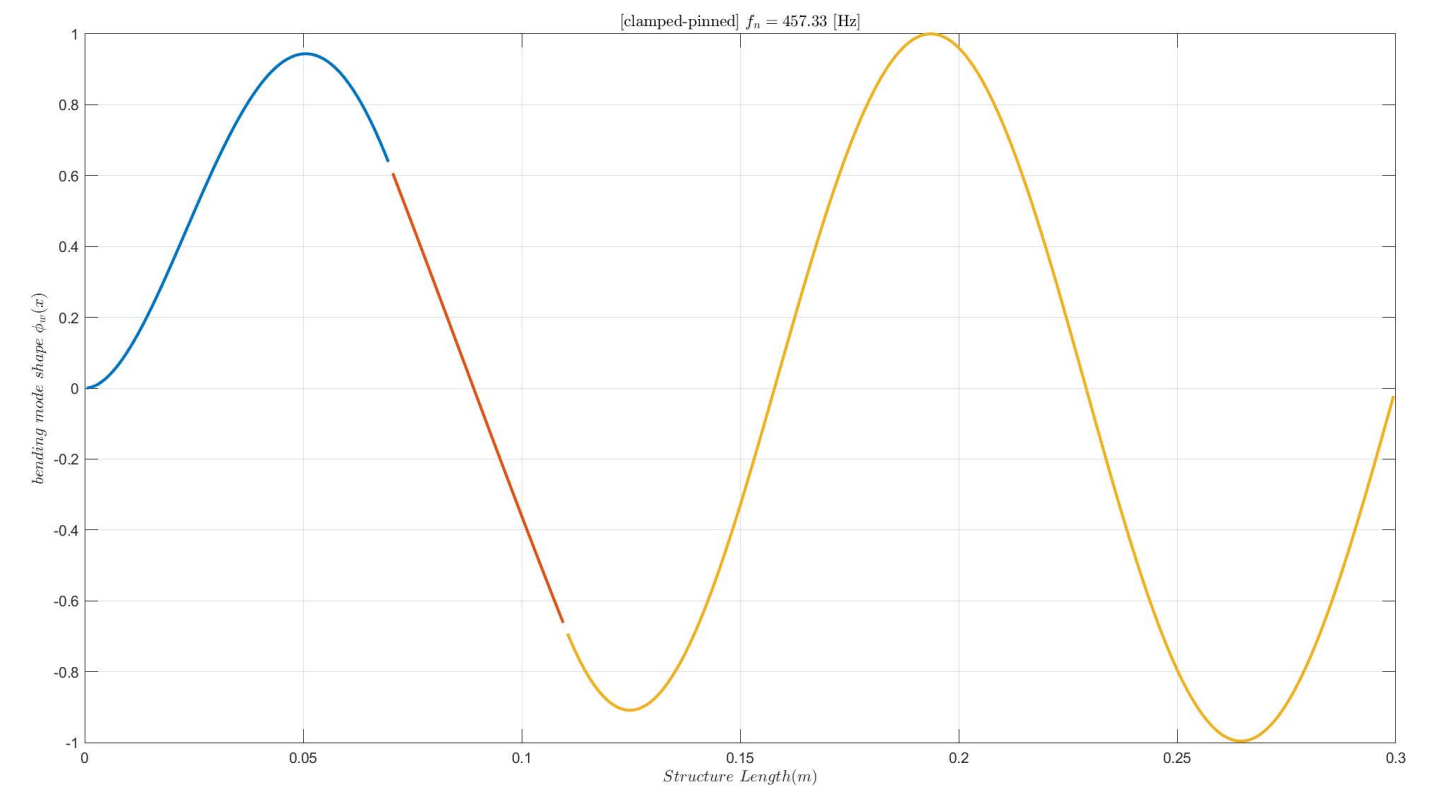
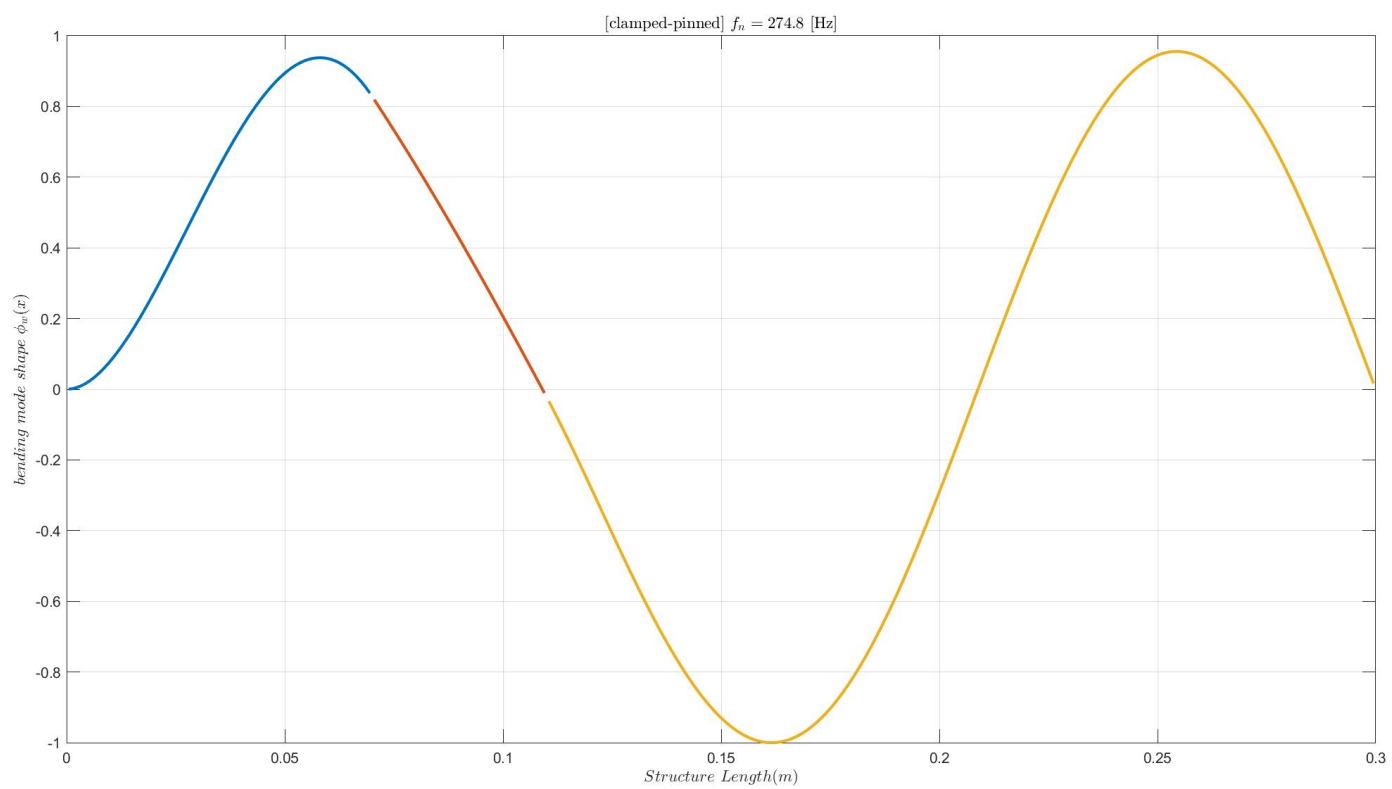
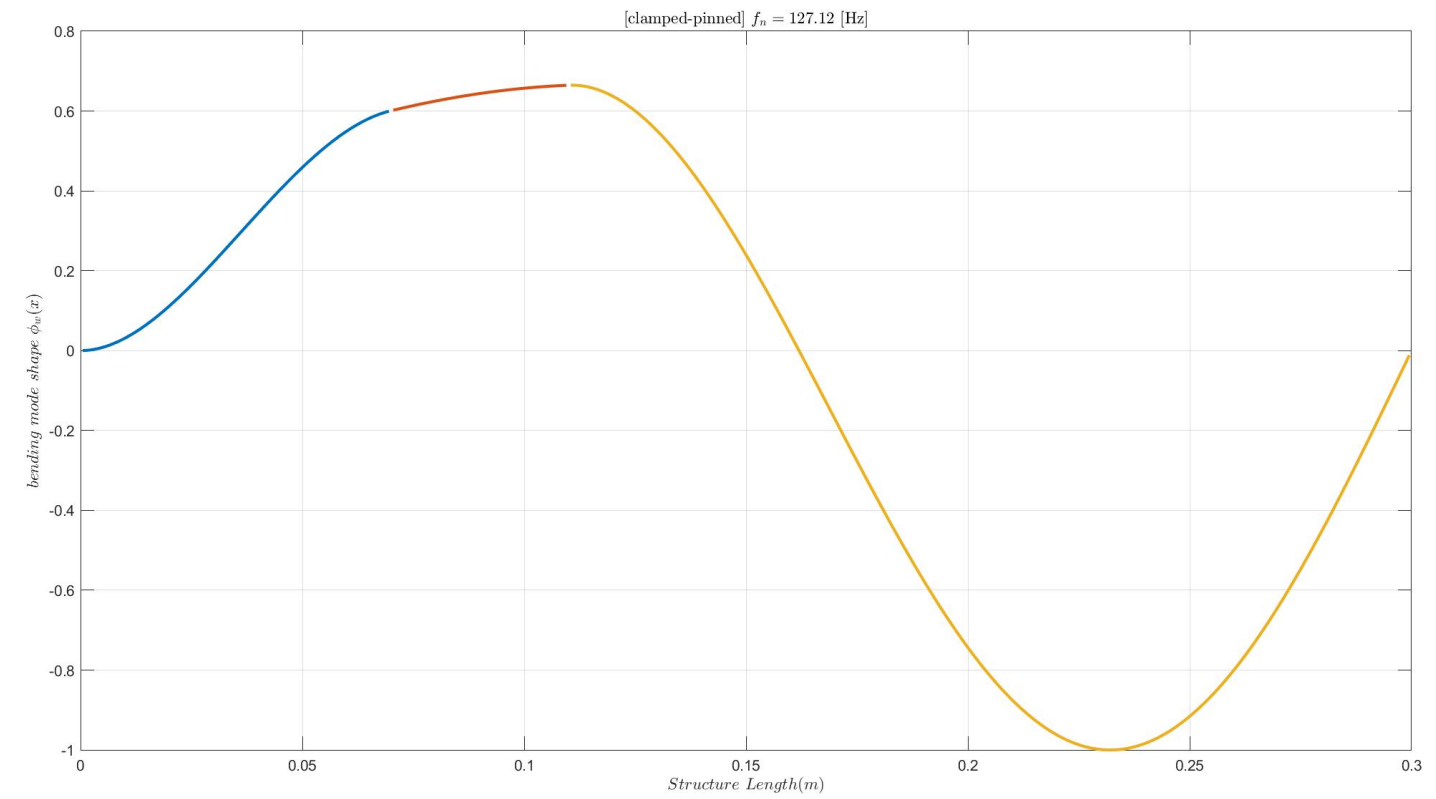
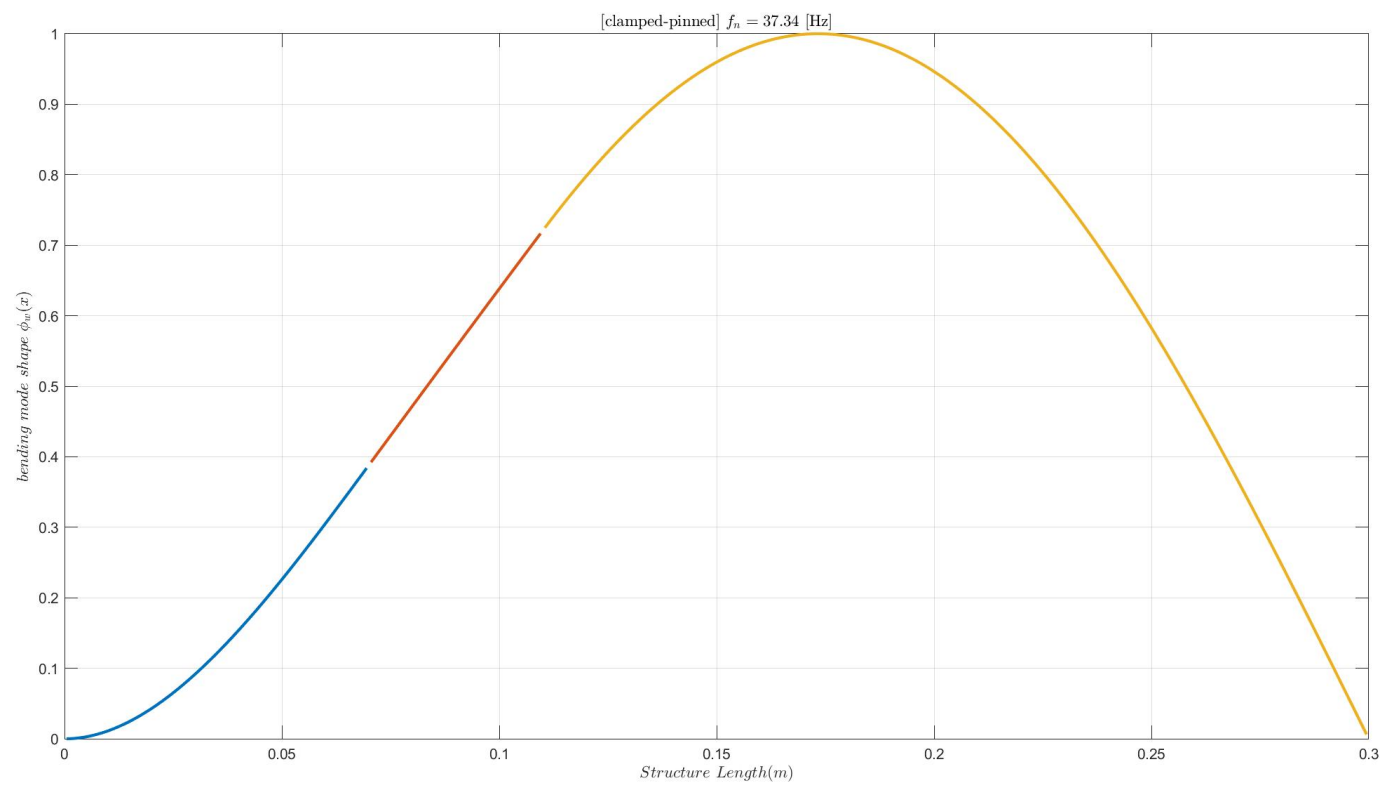
- Effective Properties for each segment;*

<i>Segment</i>	$\rho A$ [Kg/m]	$EI$ [Nm <sup>2</sup> ]
1 <sup>st</sup>	0.1560	0.3333
2 <sup>nd</sup>	0.4140	4.1582
3 <sup>rd</sup>	0.1560	0.3333

- The following table lists down first three frequencies that are obtained from these three different methods.
- For Rayleigh-Ritz approximation method, we are evaluating the values of frequencies for 9 different cases. This is done in order to study what number of Admissible shape functions would be required for the values of natural frequencies to converge close to the results from Transfer-matrix method.
- Purely base on extended run, it can be commented that the Number of shape function terms that must be included should be between 30 & 35.

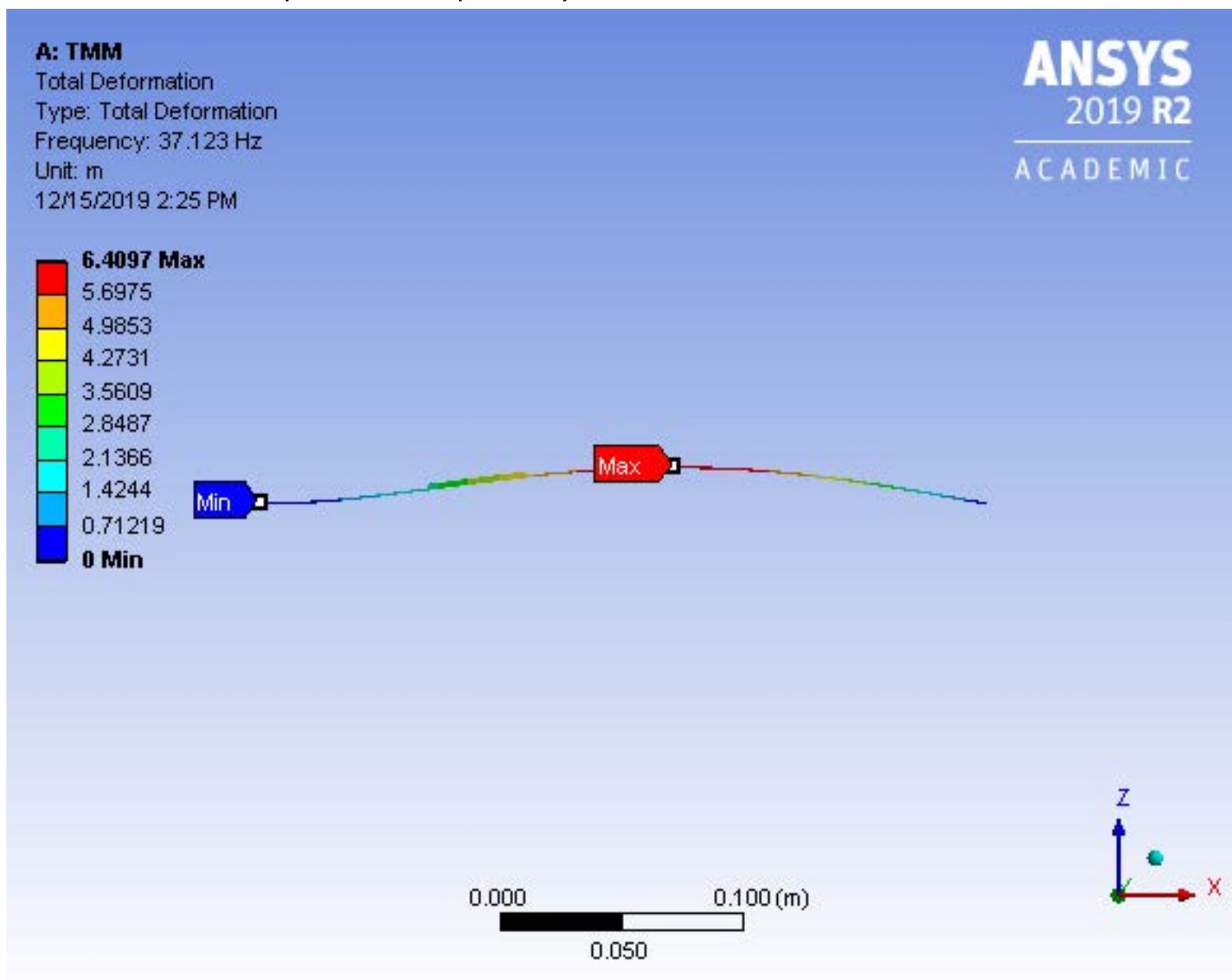
Method		First frequency $f_1$ [Hz]	Second frequency $f_2$ [Hz]	Third frequency $f_3$ [Hz]
TMM		37.34	127.12	274.8
ANSYS Modal analysis		37.123	109.64	250.35
Rayleigh-Ritz Approximation  With different cases of Number of Admissible shape function $N$ included in the analysis	<b>1</b>	41.7796	-	-
	<b>2</b>	40.1182	233.6618	-
	<b>3</b>	40.0296	173.7059	457.9501
	<b>4</b>	38.8449	163.8290	456.4827
	<b>5</b>	38.2748	145.2576	349.2241
	<b>10</b>	37.7827	138.6602	299.3835
	<b>15</b>	37.6458	136.0964	293.0809
	<b>20</b>	37.5608	132.6028	284.7574
	<b>30</b>	<b>37.3985</b>	<b>130.2814</b>	<b>281.1560</b>

# First 4 mode shapes plotted through TMM code using MATLAB

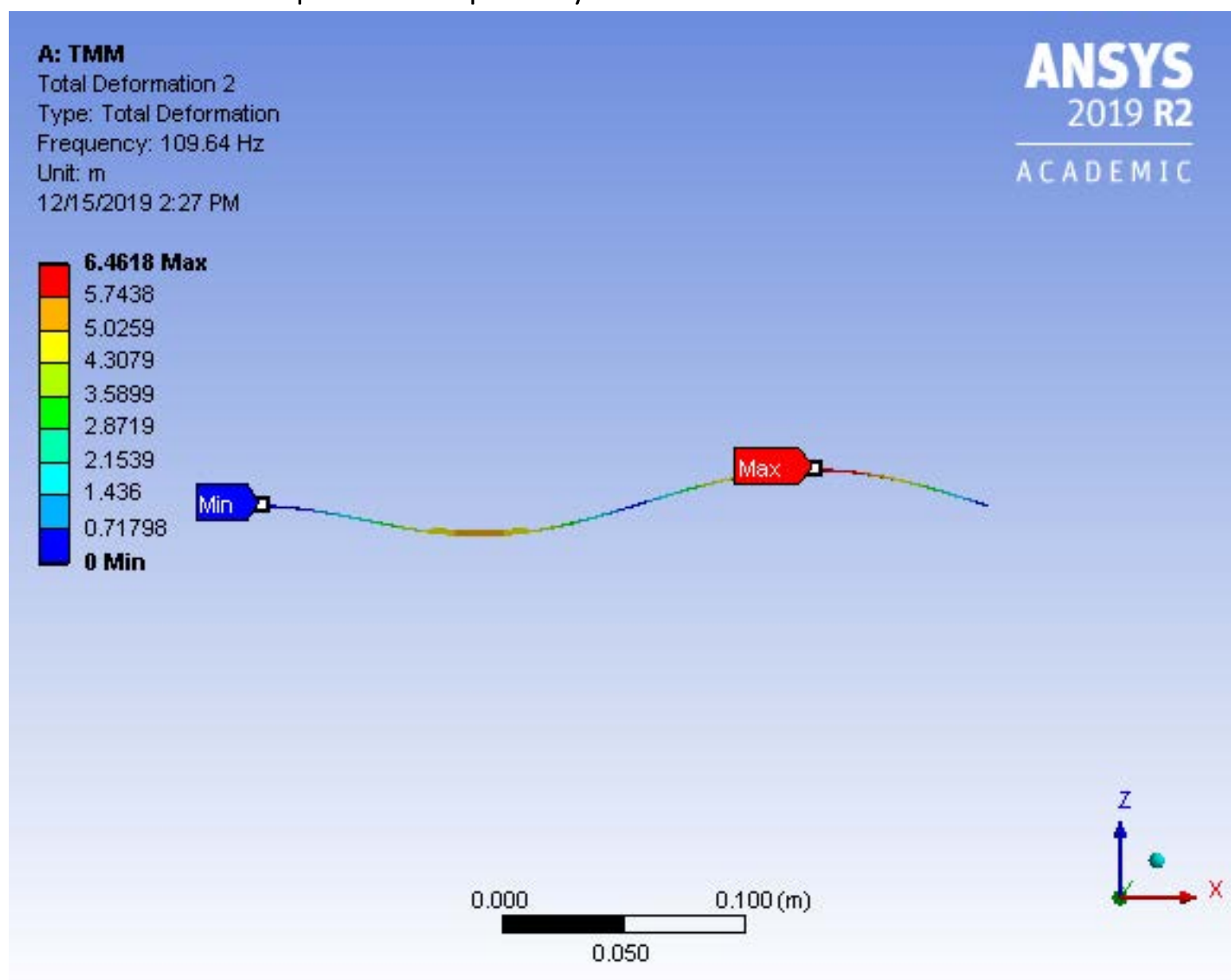


# Modal analysis done through ANSYS

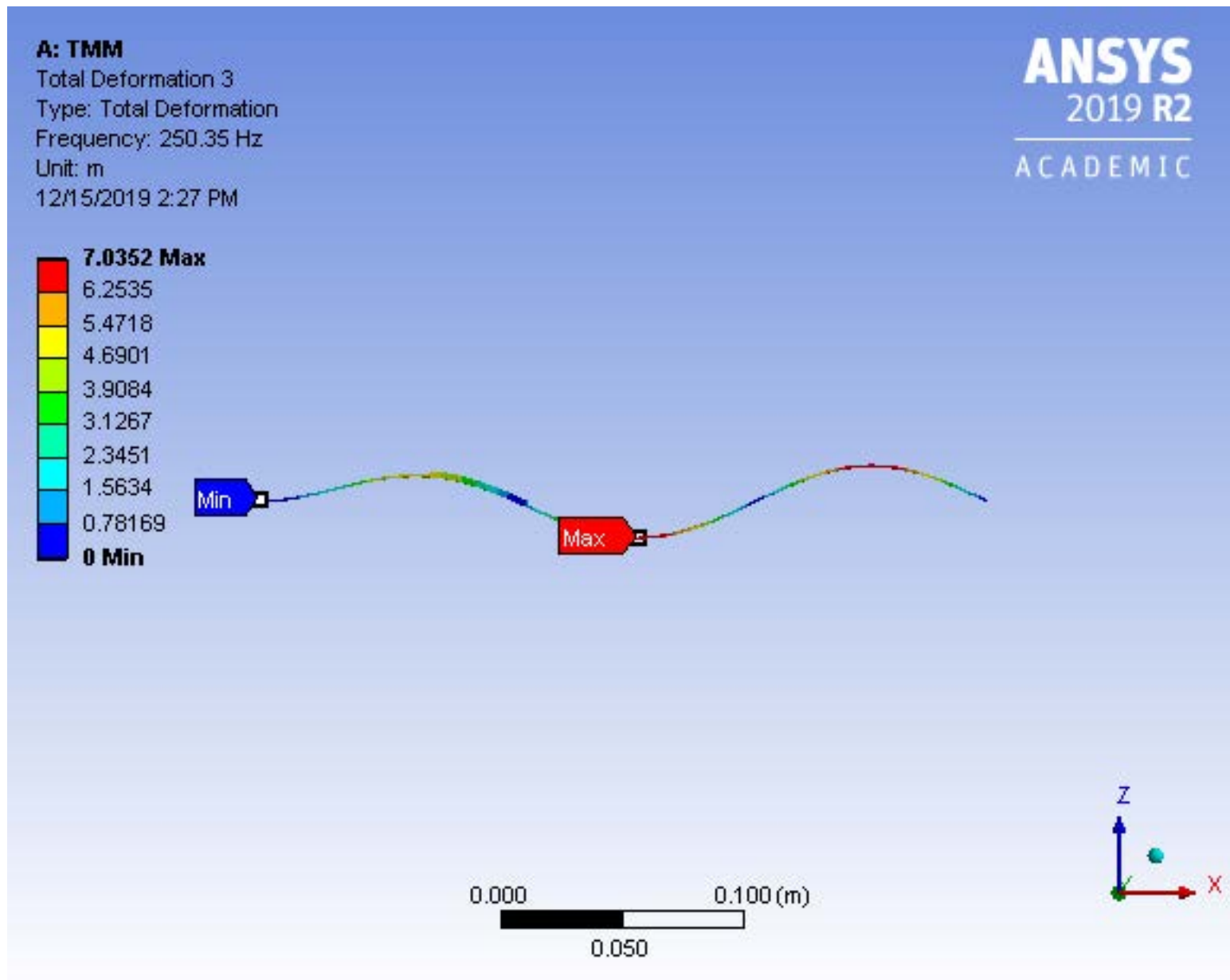
1st mode shape at frequency of  $37.123\text{ Hz}$



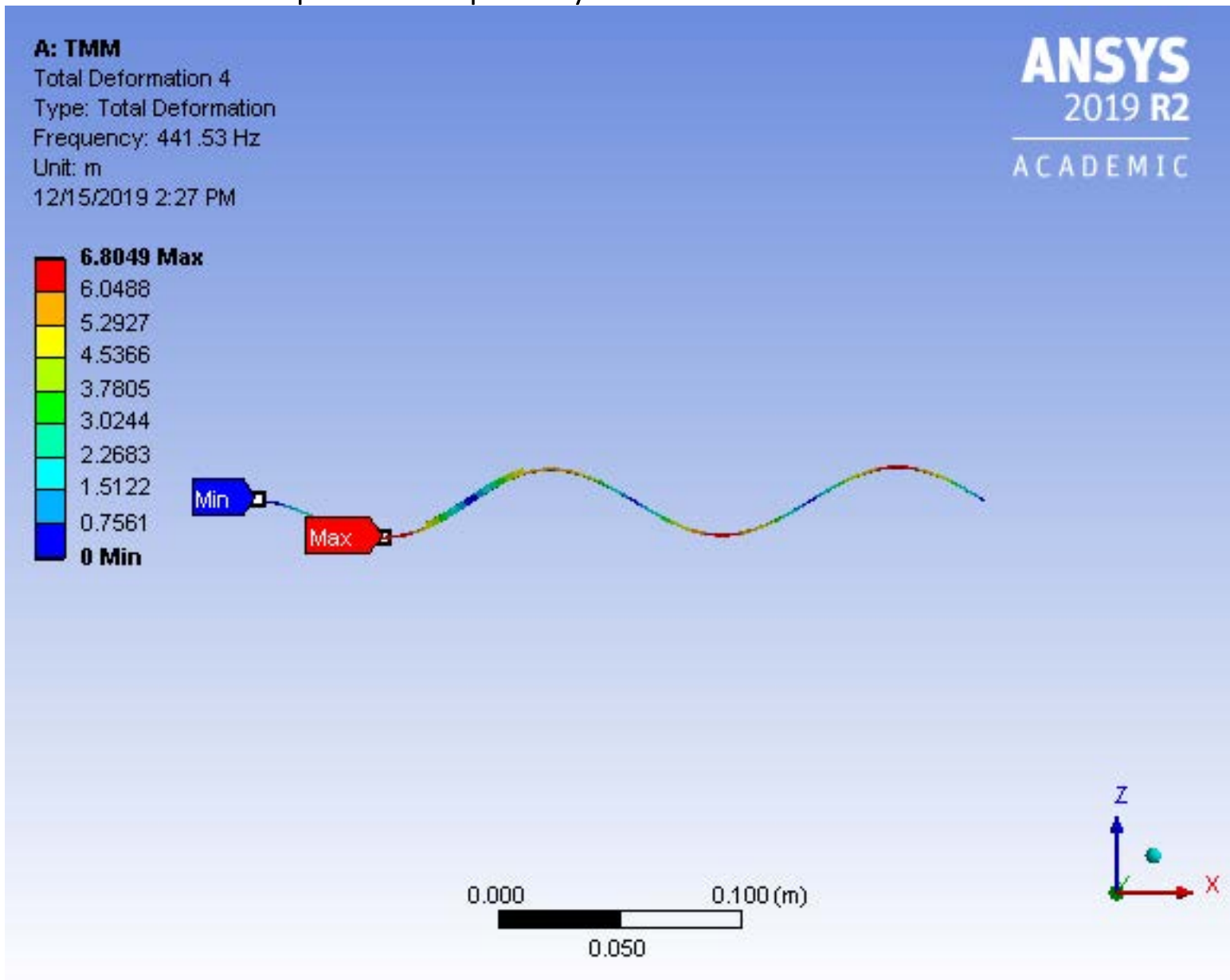
2nd mode shape at frequency of  $109.64\text{ Hz}$



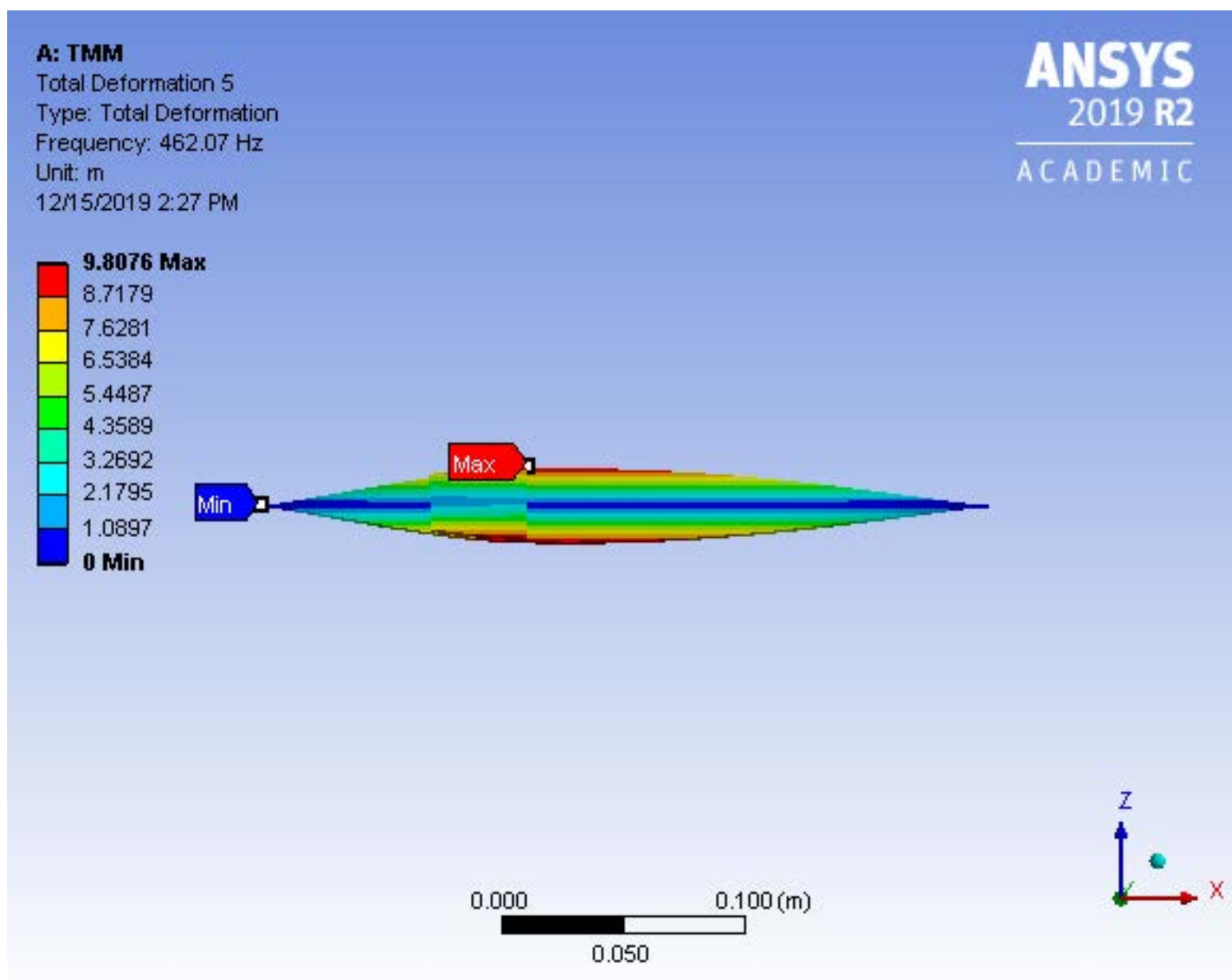
### 3rd mode shape at frequency of $250.35\text{ Hz}$



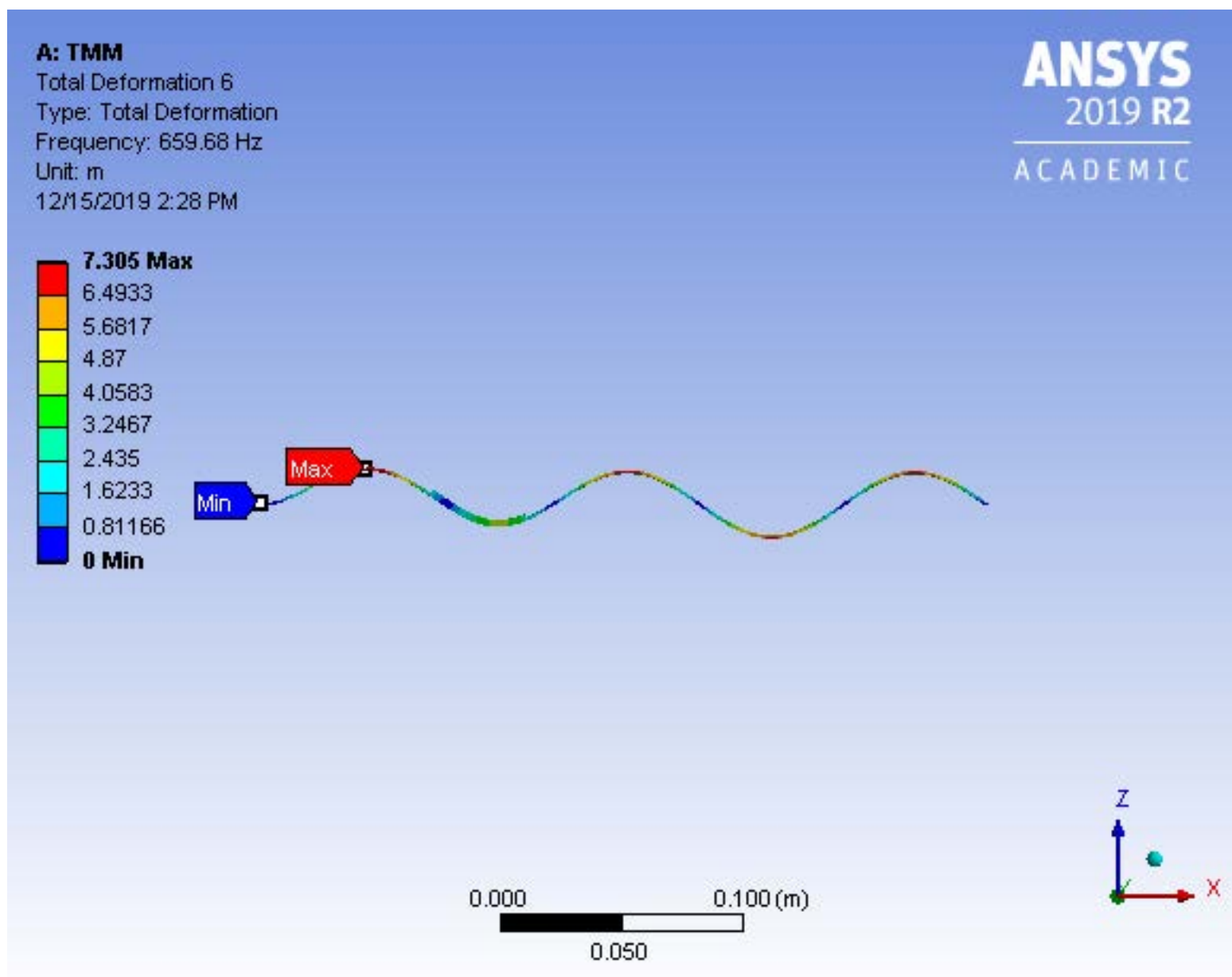
### 4th mode shape at frequency of $441.53\text{ Hz}$



## Torsional mode shape at frequency of *462.07 Hz*



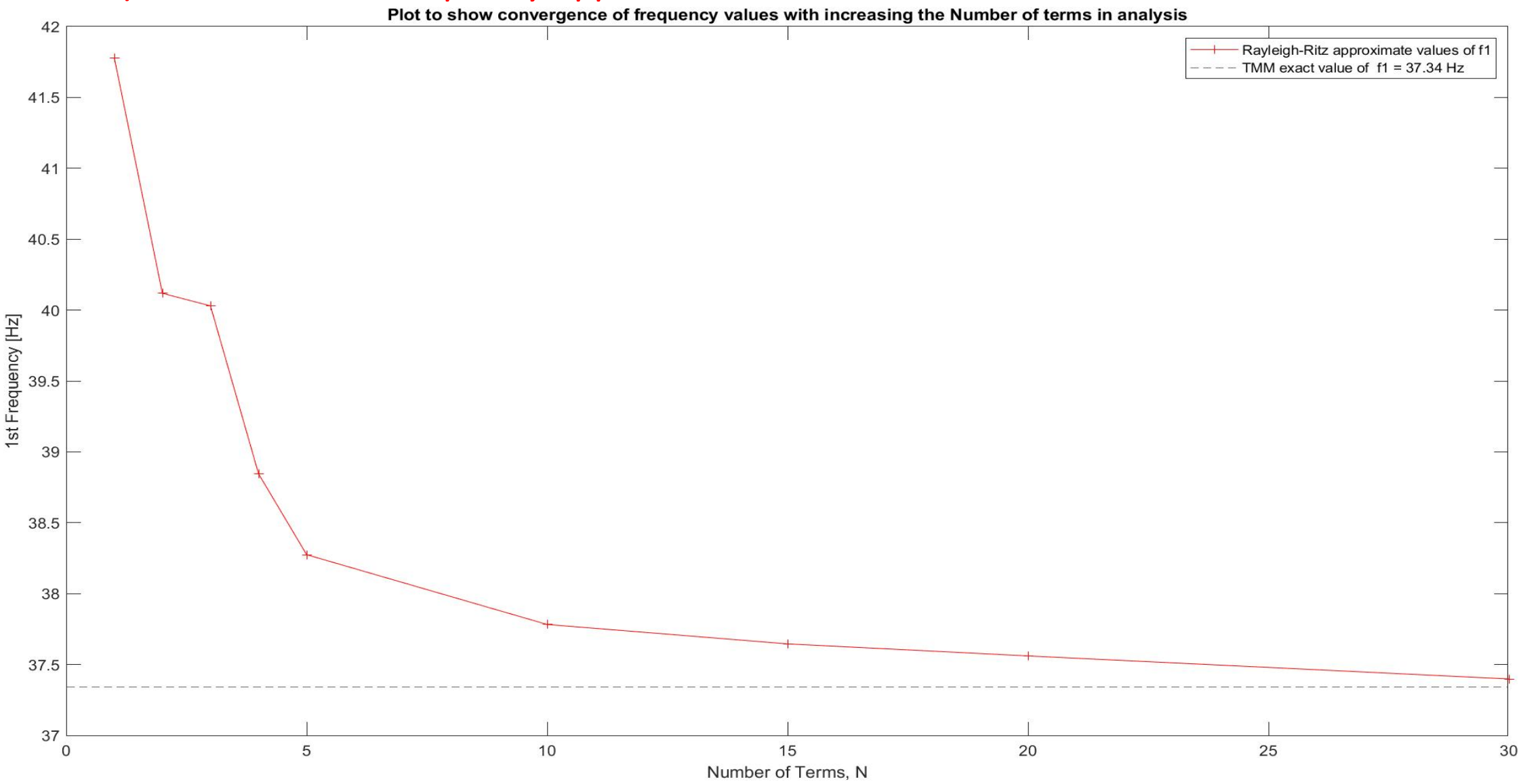
## 5th mode shape at frequency of *659.58 Hz*



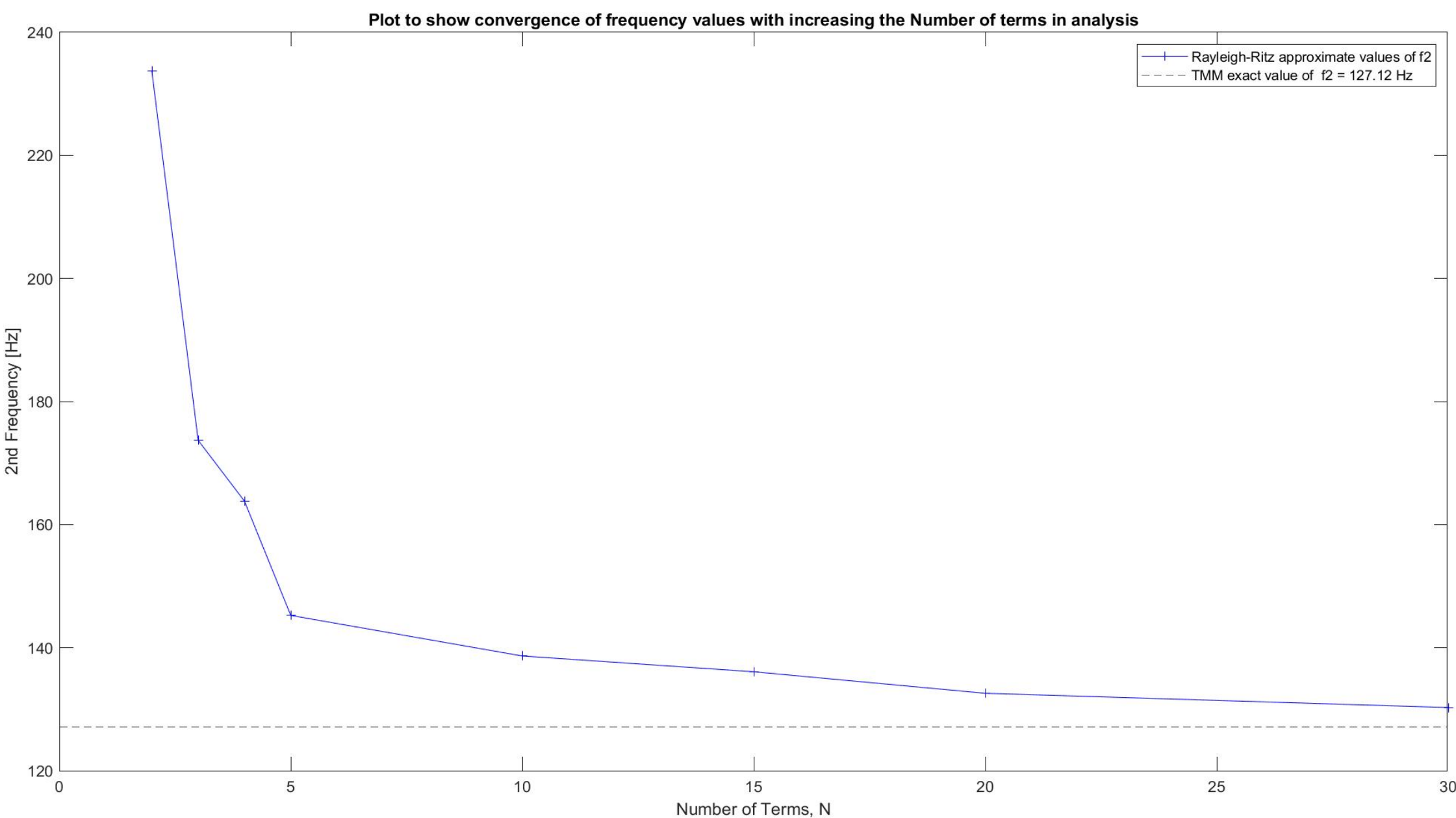


The following set of graphs shows how the natural frequency values converges to the exact value from TMM when Rayleigh-Ritz analysis includes more number of Admissible Shape functions.

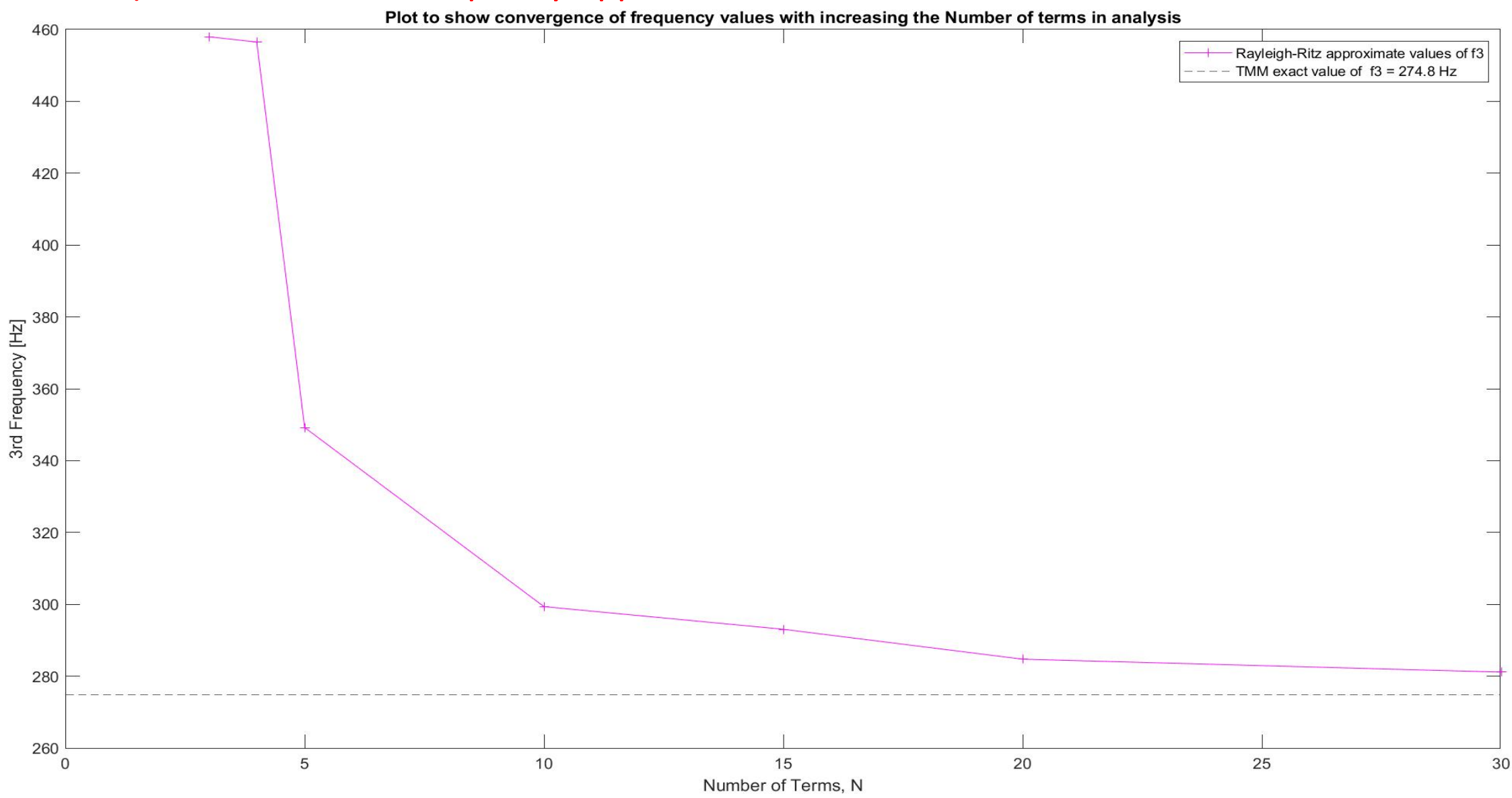
### 1.) For 1<sup>st</sup> Natural Frequency approximation



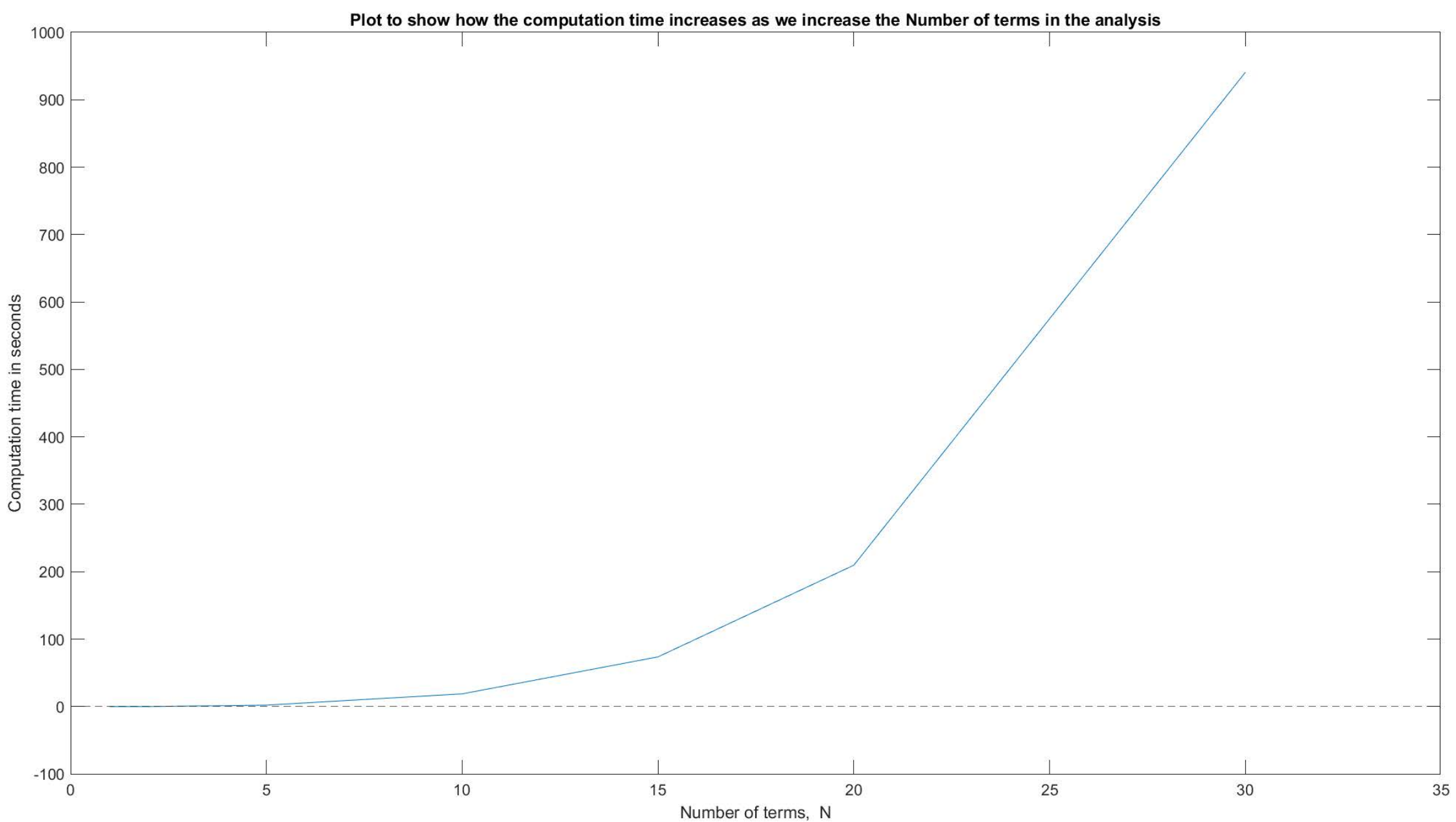
### 2.) For 2<sup>nd</sup> Natural Frequency approximation



### 3.) For 3rd Natural Frequency approximation



### 4.) Plot showing computation time -vs- Number of terms taken in the analysis



# APPENDIX

## Transfer Matrix Method code

```
clc;clear all %% INPUT parameters
% parameters listed are for "3 segmented beam" example from lecture
n = 3; %number of beam segments
EI = [1/3,4.1582,1/3]; % effective EI for [segment 1, segment 2, segment 3] in N.m^2
rhoA = [0.1560,0.4140,0.1560]; % effective rhoA for [segment 1, segment 2, segment 3] in kg/m
l = [.07, 0.04, .19]; % length for [segment 1, segment 2, segment 3] in m
BC = char('clamped-pinned'); % Given Clamped - pinned Boundary condition

w_guess = [1310]; % Keep changing this value. 1st Natural Freq b/w 10 & 550, 2nd Nat. Freq b/w 550 & 1300, 3rd Nat. Freq beyond 1300.

%% Internal calculations
globalx = [];
for i = 1:n %sweeping through segments
% Npaneli = [80, 45, 150]*1e3 + 1;
Npaneli = 1*1e3 + 1;
r_s = linspace(0,l(i),Npaneli(i)).*1e0; %panels within a segment
dr_s = (r_s(2) - r_s(1));
rmid_s = 0.5*dr_s + r_s(1:end - 1);
rmid{i} = double(rmid_s');
x_vec = rmid_s;
if i > 1
x_vec = rmid_s + sum(l(1:i - 1))*1e0;
end
% globalx = [globalx; x_vec']; %global r
globalx{i} = x_vec'; %g
end

%% Step 1. Calculating Field Transfer Matrix (FTM) for a Euler Bernoulli Beam
syms w x
assume(w > 0)
assume(x >= 0)
FTM_matrix_size = 4;
FTM_x_w = sym(NaN(FTM_matrix_size,FTM_matrix_size,n));
FTM_w = FTM_x_w;

for seg = 1:n %sweeping through beam elements
FTM_x_w(:, :, seg) = FTM_bending_class(w,x,EI(seg),rhoA(seg)); %
```

```

    FTM_w(:,:,seg) = subs(FTM_x_w(:,:,seg),x,l(seg));
end
disp('FTMs evaluated')

%% Step 2. Calculating Global Transfer Matrix (GTM) for any system
%GTM_w is a 2D Global Transfer matrix for the entire structure as a
%function of 'w'

% [ PHI(L)      ;          [ PHI(0)      ;
% d(PHI(L))/dx; = [GTM_w]* d(PHI(0))/dx ;
%   M(L)        ;          M(0)        ;
%   V(L)]       ;          V(0) ]

GTM_w = eye(FTM_matrix_size);
for seg = 1:n
    GTM_w = FTM_w(:,:,seg)*GTM_w;
end
disp('GTM evaluated')

%% Step 3. Applying B/C and finding Characteristic matrix

Char_Matrix_w = TMM_characteristic_equation_class(BC, GTM_w);
Char_Matrix_w = vpa(Char_Matrix_w,5);
% max_val = max(max(double(subs(Char_Matrix_w,w,1))));
disp(['[Bending]Characteristic matrix generated'])

%% Step 4: Finding natural frequencies
[w_h, roots] = TMM_root_finding_class(w,Char_Matrix_w,w_guess);
disp(['[Bending]Natural freq = ',num2str(0.5*w_h/pi,5),' [Hz]'])

%% Step 5: Finding Mode Shape

[PHI,PHIs,PHI_shape] = mode_shape_class(w_h,w,FTM_x_w,x,l,BC,rmid);
plot_modeS = mode_shape_plot_panel_class(1,PHI,PHIs,n,'bending',globalx,BC,w_h);

function FTM = FTM_bending_class(w,x,EI,rhoA)
%% CALCULATES THE INDIVIDUAL FIELD TRANSFER MATRIXES OF SECTIONS
%Uses Cayley-Hamilton's theory (reduces numerical error associated with expm command)
%Elements of FTM from WickenHeiser Paper
%Title: "Eigensolution of piezoelectric energy harvesters with

```

```

%geometric discontinuities: Analytical modeling and validation"
%-----INPUT-----
% w      - symbolic variable for circular natural frequency
% x      - symbolic variable for local beam length function
% EI     - scalar, effective bending rigidity value for segment
% rhoA   - scalar, effective mass/length value for segment
%-----OUTPUT-----
% FTM    - [4x4] OR [6x6] Field transfer matrix for the beam segment
%-----

Bet = (rhoA*w^2/EI)^0.25; %Beta for section

%% Mode Shape Constants
coshBx = cosh(Bet*x);
sinhBx = sinh(Bet*x);
cosBx  = cos(Bet*x);
sinBx  = sin(Bet*x);

c0      = (Bet^2*coshBx + Bet^2*cosBx)/(Bet^2 + Bet^2);
c1      = ((Bet^2/(Bet^2*Bet + Bet^3))*sinhBx + (Bet^2/(Bet^3 + Bet*Bet^2))*sinBx)/x;
c2      = (1/(x^2*(Bet^2 + Bet^2)))*(coshBx - cosBx);
c3      = ((1/(Bet^2*Bet + Bet^3))*sinhBx - (1/(Bet^3 + Bet*Bet^2))*sinBx) /x^3;

c0 = vpa(c0,5);
c1 = vpa(c1,5);
c2 = vpa(c2,5);
c3 = vpa(c3,5);

%% Elements of FTM Matrix
F33 = c0;
F34 = c1*x;
F35 = (c2*x^2)/EI;
F36 = -(c3*x^3)/EI;

F43 = (c3*x^3*rhoA*w^2)/EI;
F44 = c0 ;
F45 = (c1*x)/EI ;
F46 = -(c2*x^2)/EI;

F53 = c2*x^2*rhoA*w^2;

```

```

F54 = c3*rhoA*x^3*w^2;
F55 = c0;
F56 = - c1*x;

F63 = -c1*x*rhoA*w^2;
F64 = -c2*x^2*rhoA*w^2;
F65 = -(c3*x^3*rhoA*w^2)/EI;
F66 = c0;

FTM = [F33, F34, F35, F36;
        F43, F44, F45, F46;
        F53, F54, F55, F56;
        F63, F64, F65, F66];

```

```

% keyboard
FTM = vpa(FTM,10);
end

```

```

function [Char_Matrix_w,BC_sym,root_cols] = TMM_characteristic_equation_class(BC,GTM_w)
% Finds the symbolic characteristic equation for TMM formulation
%-----INPUT-----
% BC      - string, geometric BC on either side ('clamped','pinned','free')
%          example: 'clamped-clamped' OR 'clamped-free'
% GTM_w   - global transfer matrix of system (usually a 4x4 matrix)
% symbolic in 'w'
%-----OUTPUT-----
% Char_Matrix_w - characteristic matrix symbolic in 'w'
% BC_sym       - [4x2] matrix of symbolic B/C for root (column 1) and tip
%                (column 2)
%                Sequence is: [   PHIx ; %mode shape
%                               d(PHIX)/dx; %slope
%                               M      ; %moment
%                               V      ] %shear force
% root_cols    - Non-zero states at the root
%-----

```

```

%% Actual evaluations start from here

```

```

idx      = strfind(BC,'-');

```

```

BC_tip      = BC(idx + 1:end);
BC_root     = BC(1 : idx-1);
BC_vec{1}   = BC_tip;
BC_vec{2}   = BC_root;

```

```

syms Phi PHI_x M V

```

```

BC_sym = sym(zeros(4,2));
states_sym = [Phi;PHI_x;M;V];
for i = 1:2
    BCi = BC_vec{i};
    if strcmp(BCi,'clamped') == 1
        states = [0;0;1;1];
    elseif strcmp(BCi,'free') == 1
        states = [1;1;0;0];
    elseif strcmp(BCi,'pinned') == 1
        states = [0;1;0;1];
    else
        error('Either BC is pinned/clamped/free. Pls input only these')
    end
    BC_sym(:,i) = states.*states_sym;
end
tip_rows      = find(BC_sym(:,1) == 0); % Zero states at the tip
root_cols     = find(BC_sym(:,2) ~= 0); % Non-zero states at the root
Char_Matrix_w = GTM_w(tip_rows,root_cols);

```

```

end

```

```

function [w_h, roots] = TMM_root_finding_class(w,Char_Matrix_w, w_guess)
assume(w > 0)

```

```

% Char_Matrix_w = Char_Matrix_check(Char_Matrix_w,w);
Det_F          = Char_Matrix_w(1,1)*Char_Matrix_w(2,2) - Char_Matrix_w(1,2)*Char_Matrix_w(2,1);
Det_F          = vpa(Det_F,5);
disp('Characteristic determinant evaluated')
w_h           = [];

```

```

[w_h_temp,fval] = fzero(@(w0) double(subs(Det_F,w,w0)),w_guess);

w_h_temp      = double(abs(w_h_temp));
w_h           = uniquetol(sort([w_h,w_h_temp]),1e-3);
w_h           = nonzeros(w_h);
w_h           = transpose(nonzeros(w_h));

if isempty(w_h) == 1
    error('No frequencies found within search range !')
end
disp('Finished solving Eigenvalue problem.')
nroots = numel(w_h);

end

function [PHI,PHIs,PHI_symbolic] = mode_shape_class(w_h,w,FTM_x_w,x,l_s,BC,r)
%% Mode shape
% Finding the maximum value of a plotted mode shape and then normalizing
% with the max value
syms x
[PHI_shape_temp] = get_sym_bending_modeshape(w_h,w,FTM_x_w,x,l_s,BC);%FTM_w,(w_a(i),w,FTM_x_w,x,FTM_w,l_s);
PHI_shape      = transpose(simplify(expand(PHI_shape_temp)));
PHI_shape      = vpa(PHI_shape,8);

disp('Finished evaluating symbolic mode shape.')

%% Mode shape normalization
PHI = [];
[~,~,n] = size(FTM_x_w);          % number of TMM beam segments
PHI_value = [];
for i = 1:n
    PHI_value_s = double(subs(PHI_shape(i,:), x, r{i})); %segmental mode shape
    PHI_value    = vertcat(PHI_value, PHI_value_s);
    PHI          = vertcat(PHI, PHI_value_s);
    PHIs{i}     = PHI_value_s;
end

max_mode_value = max(abs(PHI_value));
PHI_symbolic   = PHI_shape/max_mode_value;
PHI            = PHI/max_mode_value;

```



```

for i = 1:n
    PHIs{i} = PHIs{i}./max_mode_value;
end

```

```

disp('Finished normalizing symbolic mode shape.')

```

```

end

```

```

function PHI_shape = get_sym_bending_modeshape(w_eval,w,FTM_x_w,x,l_s,BC)
%% EVALUATES THE SYMBOLIC MODE SHAPE
%This function plots the mode shape corresponding to a natural frequency
%-----INPUT-----
% w_eval - scalar, current natural freq. (rad/s) for which mode shape is reqd.
% w - scalar, symbolic natural freq.
% FTM_x_w - Field transfer matrix (symbolic in 'x' and 'w')
% x - symbolic beam local coordinate along length
% l_s - [1xns] length of each segment
% BC - string, geometric BC on either side ('clamped','pinned','free')
% example: 'clamped-clamped' OR 'clamped-free'
% FTM_size - whether to perform TMM in 4x4 or more general 6x6 format
%-----OUTPUT-----
% PHI_shape - [1xns] segmental mode shape corresponding to 'w_eval' natural freq.
%-----
% keyboard
[nrow, ncol, n] = size(FTM_x_w);
FTM_w = sym(NaN(nrow,ncol,n));
% [N,~] = size(FTM_x_w);
mul = 1;
for j = 1:n
    FTM_w(:, :, j) = subs(FTM_x_w(:, :, j), x, l_s(j));
    termj = subs(FTM_w(:, :, j), w, w_eval);%( FTM_w,freq_w,w,ns,BC)
    mul = termj*mul;
end
FTM = subs(FTM_w,w,w_eval); %3D matrix 'FTM' is just numbers
GTM = double(mul);
% GTM = vpa(GTM,5);

[U, ~,root_cols] = TMM_characteristic_equation_class(BC,GTM);
PHI_shape = sym(NaN(1,n));

```

```

if U(1,2) ~= 0
    k = - U(1,1)/U(1,2);
elseif U(2,2) ~= 0
    k = - U(2,1)/U(2,2);
else
    k = 0;
end
k = double(k);

%% Current segment is just the beam and all previous segments are beam + point mass FTM's
% keyboard
for seg = 1:n
    GTM_seg_x = eye(nrow); %4x4 matrix
    if seg > 1
        for i = 1: seg - 1 %segmental sweep before current segment
            GTM_seg_x = FTM(:, :, i)*GTM_seg_x; %at this point this matrix 'GTM_seg_x' is just numbers
        end
        GTM_seg_x = subs(FTM_x_w(:, :, seg), w, w_eval)*GTM_seg_x; %at this point this matrix 'GTM_seg_x' is just a function of 'x'
    else
        GTM_seg_x = subs(FTM_x_w(:, :, seg), w, w_eval); %for a structure with just 1 segment
        GTM_seg_x = vpa(GTM_seg_x); %4x4 matrix OR 6x6 matrix
    end

    PHI_shape(seg) = GTM_seg_x(1,root_cols)*[1; k];

end

end

function plot_modeS = mode_shape_plot_panel_class(figno, PHI, PHIs, n, mode_type, x_vec, BC, omega)

legend_type = 'n_h';
mode_symbol = '\phi_w(x)';

plot_modeS = figure(figno);
set(plot_modeS, 'defaulttextinterpreter', 'latex')

for i = 1:n

```

```

        plot(x_vec{i}, PHIs{i}, 'LineWidth', 2)
        hold on
end

% plot(x_vec, PHI, 'LineWidth', 2);
% legend_stringh = ['f_h', ' = ', num2str(0.5*omega/pi, 5), ' [Hz]'] ;

grid on
% legend(legend_stringh);

title(['[', BC, ']', ' $f_n$, ' = ', num2str(0.5*omega/pi, 5), '$ [Hz]'])
xlabel('$Structure\,\,Length(m)$')
ylabel(['$', char(mode_type), '\,\,mode\,\,shape\,\,', mode_symbol, '$']) %'
% set(legend, 'Location', 'Best', 'LineWidth', 1, 'FontSize', 9)

end

```

## Rayleigh-Ritz code

```
function Rayleigh_Ritz_20
tic
clc
format short

N_modes = 20; % Different cases N = [1 2 3 4 5 10 15 20 30]

% Material Properties

E_s = 200e9;      % Pa, substrate elastic modulus
rho_s = 7800;    % kg/m^3, substrate density
E_p1 = 120e9;    % Pa, patch 1 elastic modulus
rho_p1 = 8600;  % kg/m^3, patch 1 density

% Beam Substrate Geometry

w = 0.02;       % m
ts = 0.001;    % m
L = 0.3;       % m

% Patch Geometry

Lp1 = 0.04;     % m, patch 1 length
tp1 = 0.0015;  % m, patch 1 thickness
x1p1 = 0.07;   % m, patch 1 start position
x2p1 = x1p1+Lp1; % m, patch 1 end position

% Specify Applied Force

f = 0;

% Define mode shape coefficients for cantilevered beam
lam_L = [3.92660231 7.06858275 10.21017612 13.35176878 16.49336143];
sig = [1.0008 1.0000 1.0000 1.0000 1.0000];
if N_modes > 5
    for i = 5:N_modes
        lam_L = [lam_L ((4*i+1)*pi/4)];
        sig = [sig 1];
    end
end
end
```

```

lam = lam_L/L;

% Compute System Stiffness, Mass, and Force Input Matrices
Ks = zeros(N_modes,N_modes);
Kp1 = zeros(N_modes,N_modes); % Preallocate matrixes to speed up matlab

Ms = zeros(N_modes,N_modes);
Mp1 = zeros(N_modes,N_modes);

for i = 1:N_modes
    for j = 1:N_modes
        Ks(i,j) = E_s*triplequad(@(x,y,z)K_integrnd(x,y,z,i,j,sig,lam,L),0,L,-w/2,w/2,-ts/2,ts/2);
        Kp1(i,j) = E_p1*triplequad(@(x,y,z)K_integrnd(x,y,z,i,j,sig,lam,L),x1p1,x2p1,-w/2,w/2,ts/2,ts/2+tp1);

        Ms(i,j) = rho_s*triplequad(@(x,y,z)M_integrnd(x,y,z,i,j,sig,lam,L),0,L,-w/2,w/2,-ts/2,ts/2);
        Mp1(i,j) = rho_p1*triplequad(@(x,y,z)M_integrnd(x,y,z,i,j,sig,lam,L),x1p1,x2p1,-w/2,w/2,ts/2,ts/2+tp1);
    end
end
K = Ks+Kp1; % system stiffness matrix
M = Ms+Mp1; % system mass matrix

% compute natural frequencies
K_til2 = M\K;
wn = sort(eig(K_til2).^(1/2));
fn = wn/(2*pi);
f1 = fn(1)
f2 = fn(2)
f3 = fn(3)
toc
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Integrand Functions %%%%%%%%%
function int_K = K_integrnd(x,y,z,i,j,sig,lam,L)
xi = x;
d2phidx_d2phidx = (lam(i)^2*(cosh(lam(i)*xi)+cos(lam(i)*xi)-sig(i)*(sinh(lam(i)*xi)+sin(lam(i)*xi)))).*(lam(j)^2*(cosh(lam(j)*xi)
+cos(lam(j)*xi)-sig(j)*(sinh(lam(j)*xi)+sin(lam(j)*xi)));
int_K = z.^2.*d2phidx_d2phidx+0.*y; % Integrand is not a function of y, but matlab needs y to be included
end

function int_M = M_integrnd(x,y,z,i,j,sig,lam,L)

xi = x;
phi_phi = (cosh(lam(i)*xi)-cos(lam(i)*xi)-sig(i)*(sinh(lam(i)*xi)-sin(lam(i)*xi))).*(cosh(lam(j)*xi)-cos(lam(j)*xi)-sig(j)*(sinh
(lam(j)*xi)-sin(lam(j)*xi))); % modes for cantilevered beam
int_M = phi_phi+0.*y+0.*z; % Integrand is not a function of y or z, but matlab needs them to be included
end

```