*This registration is subject to approval(s) from faculty advisor/Course Instructor/Academic office.

Year/Semester: 2024-25/Autumn

Course Code	Course Name	Tag	Credits	Grade	Credit/Audit
AE 616	Gas Dynamics	Additional Learning	6.0		C
GC 101	Gender in the workplace	Core course	0.0		N
ME 601	Stress Analysis	Department elective	6.0		C

Year/Semester: 2024-25/Project

Course Code	Course Name	Tag	Credits	Grade	Credit/Audit
AE 796	I Stage Project	Core course	42.0		C

Year/Semester: 2023-24/Spring

Course Code	Course Name	Tag	Credits	Grade	Credit/Audit
AE 673	Fiber Reinforced Composites	Core course	6.0	AB	C
AE 678	Aeroelasticity	Core course	6.0	BB	C
AE 694	Seminar	Core course	4.0	AB	C
AE 714	Aircraft Design	Department elective	6.0	BB	C
AE 899	Communication Skills	Core course	6.0	PP	N
GC 101	Gender in the workplace	Core course	0.0	NP	N
TD 626	Technology, Society and Development	Institute elective	6.0	BB	C

Year/Semester: 2023-24/Autumn

Course Code	Course Name	Tag	Credits	Grade	Credit/Audit
AE 649	Finite Element Method	Core course	6.0	BC	C
AE 705	Introduction to Flight	Core course	6.0	BB	C
AE 709	Aerospace Structures	Core course	6.0	BB	C
AE 715	Structural Dynamics	Core course	6.0	BC	C
AE 727	Aircraft Structural Mechanics Lab.	Core course	4.0	BB	C
GC 101	Gender in the workplace	Core course	0.0	NP	N
TA 101	Teaching Assistant Skill Enhancement & Training (TASET)	Core course	0.0	PP	N

Personal Info

Krishna Teja Yarranagula Room : H12-C-012
Rollno : **23M0036**
Mobile :7995875345
M.Tech.,Aerospace Engineering

Finite Element Analysis

Project report (AE-649)

Master in Technology

in

Aerospace Engineering

By

Yarranagula Krishna Teja

(23M0036)

Under the Guidance of

Prof. P J Guruprasad

Department of Aerospace Engineering

Indian Institute of Technology Bombay

Mumbai-400076, India

```
%% Rayliegh Ritz Method %%
clear all                                                  %Clear all
syms x  w1 w2                                                 % defining variable as symbols
syms  C1 C2                                                  % Variable for trail polynomial soln
L=1
x_=[0:L]
u__x= C1*x*(1-x) + C2*x*x*(1-x)                            %Trial soln
du__dx= diff(u__x)
du___dx=diff(du__dx)
%% rayleigh ritz
disp('rayleigh ritz METHOD');
R__x= -(du___dx)-u__x+x^2                                              %Residue EQN **
w1=x*(1-x)
w2=x*x*(1-x)
G_K_1 = (R__x)*(w1)
G_K_2 = (R__x)*(w2)
expand(G_K_1)
expand(G_K_2)
G_K_1 = int(G_K_1, x, 0, L)                                              %integrating over domain
G_K_2 = int(G_K_2, x, 0, L)
[c1,c2] = solve(G_K_1,G_K_2)
```

```
% Tapered beam
%2 elements
B=[1 0 0 0; 0 1 1 0;0 0 0 1]
A=B.'
k1=(2.5*10*200000/(100/2))
k2=(1.5*10*200000/(100/2))
Ke=[k1 -k1 0 0;-k1 k1 0 0;0 0 k2 -k2;0 0 -k2 k2]
K=B*Ke*A
error = (100*(-0.00267+0.00275))/0.00275
%3 elements
B_=[1 0 0 0 0 0; 0 1 1 0 0 0;0 0 0 1 1 0;0 0 0 0 0 1]
A_=B_.'
k_1=(2.67*10*200000/(100/3))
k_2=(2*10*200000/(100/3))
k_3=(1.33*10*200000/(100/3))
Ke=[k_1 -k_1 0 0 0 0;-k_1 k_1 0 0 0 0;0 0 k_2 -k_2 0 0;0 0 -k_2 k_2 0 0;0 0 0 0 k_3 -k_3;0 0 0 0 -k_3 
k_3]
K_=B_*Ke*A_
C=[280200 -120000 0;-120000 199800 -79800;0 -79800 79800]
D=[0; 0; 100]
Y=linsolve(C,D)
error=(100*(-0.0027+0.00275))/0.00275
```

```
## petroGlaerkin_sample.......##
clear all                                                          %Clear all
syms x  w1                                                          % defining variable as symbols
syms  C1                                                           % Variable for trail polynomial soln
L=1
x_=[0:L]
u__x= 1-x + C1*x*(1-x)                                     %Trial soln
du__dx= diff(u__x)
du___dx=diff(du__dx)
%% petrov Galerkin Method
disp('petrov GALERKIN METHOD');
R__x= -2*u__x*du___dx+du__dx*du__dx-4                                                     %Residue EQN **
w1=1
G_K_1 = (R__x)*(w1)
expand(G_K_1)
G_K_1 = int(G_K_1, x, 0, L)                                                       %integrating over domain
G_K = solve(G_K_1)
C_1= double(G_K)
```

```
## Least Sqaure Method ......###
clear all                                                     %Clear all
syms x  w1                                                     % defining variable as symbols
syms  C1                                                      % Variable for trail polynomial soln
L=1
x_=[0:L]
u__x= 1-x + C1*x*(1-x)                               %Trial soln
du__dx= diff(u__x)
du___dx=diff(du__dx)
%% least square method
disp('least square METHOD');
R__x= -2*u__x*du___dx+du__dx*du__dx-4 %Residue EQN **
w1=2*C1+2
G_K_1 = (R__x)*(w1)
expand(G_K_1)
G_K_1 = int(G_K_1, x, 0, L)                                                   %integrating over domain
G_K = solve(G_K_1)
C_1= double(G_K)
```

```
clear all                                                       %Clear all
syms x  w1                                                       % defining variable as symbols
syms  C1                                                        % Variable for trail polynomial soln
L=1
x_=[0:L]
u__x= 1-x + C1*x*(1-x)                                 %Trial soln
du__dx= diff(u__x)
du___dx=diff(du__dx)
%% Galerkin Method
disp('GALERKIN METHOD');
R__x= -2*u__x*du___dx+du__dx*du__dx-4                                              %Residue EQN **
w1=x*(1-x)
G_K_1 = (R__x)*(w1)
expand(G_K_1)
G_K_1 = int(G_K_1, x, 0, L)                                                 %integrating over domain
G_K = solve(G_K_1)
C_1= double(G_K)
```

```
%.................................................................
% MATLAB codes for Finite Element Analysis
% Discrete_Systems.m

% clear memory

clear all

% elementNodes: connections at elements

elementNodes=[1 2;2 3;2 4];

% numberElements: number of Elements

numberElements=size(elementNodes,1);

% By using the MATLAB function size, that returns the number
%of lines and columns of a rectangular matrix, we can detect
%the number of elements by inspecting the number of lines of
%matrix elementNodes.

% numberNodes: number of nodes

numberNodes=4;

% for structure:
% displacements: displacement vector
% force : force vector
```

```
% stiffness: stiffness matrix

% initializing the matrices

displacements=zeros(numberNodes,1);
force=zeros(numberNodes,1);
stiffness=zeros(numberNodes);

% applied load at node 2
force(2)=10.0;

% computation of the system stiffness matrix
% We compute now the stiffness matrix for each element in turn and then assemble
% it in the global stiffness matrix.

for e=1:numberElements;
% elementDof: element degrees of freedom (Dof)
elementDof=elementNodes(e,:) ;

stiffness(elementDof,elementDof)=...
stiffness(elementDof,elementDof)+[1 -1;-1 1];
end


% boundary conditions and solution
% prescribed dofs
prescribedDof=[1;3;4];
% free Dof : activeDof
```

```
activeDof=setdiff([1:numberNodes],[prescribedDof]);

% solution
displacements=stiffness(activeDof,activeDof)\force(activeDof);
% positioning all displacements
displacements1=zeros(numberNodes,1);
displacements1(activeDof)=displacements;
% output displacements/reactions
outputDisplacementsReactions(displacements1,stiffness,numberNodes,prescribedDof)


%.............................................................
function outputDisplacementsReactions(displacements1,stiffness,GDof,prescribedDof)
% output of displacements and reactions in
% tabular form
% GDof: total number of degrees of freedom of
% the problem
% displacements
%displacements=displacements1;
GDof=4;
jj=[1:GDof];
disp('displacements')
 [ jj.' displacements1]

% reactions
F=stiffness*displacements1;
reactions=F(prescribedDof);
disp('reactions')
```

```
[prescribedDof reactions]
end
%.............................................................
```