International Journal of Trend in Research and Development, Volume 4(1), ISSN: 2394-9333
www.ijtrd.com

MATLAB Code for Structural Analysis of 2-D Structures Subjected to Static and Self-Weight Loading Conditions

Naman Jain
Department Of Mechanical Engineering, G. B. Pant University of Agriculture and Technology, Pantnagar, India

Abstract: In present investigation MATLAB code for structural analysis of 2-dimension linear elastic isotropic structures subjected to static and self-weight loading conditions is been presented. In this paper implementation of MATLAB code on two structures i.e. Cantilever beam and Bolkow Blohm Beam is done to show the application of code. The meshing of the 2-dimension structure is done with quadrilateral 4-node element, for analysis of self-weight loading condition the weight of an element is equally transferred on each node in downward direction and results obtained through MATLAB code is been compared with ANSYS software.

Keywords— *Finite element method; self-weight; MATLAB; 4-node element*

I. INTRODUCTION

The finite element method **(Hutton 2004; Chandrupatla and Belegunda 2004; Khennane 2013)** represent is one of the most significant achievements in the field of computational methods in the last century. Historically, it has its roots in the analysis of weight-critical framed aerospace structures. These framed structures were treated as an assemblage of one-dimensional members, for which the exact solutions to the differential equations for each member were well known. These solutions were cast in the form of a matrix relationship between the forces and displacements at the ends of the member. Hence, the method was initially termed matrix analysis of structures. Later, it was extended to include the analysis of continuum structures. Since continuum structures have complex geometries, they had to be subdivided into simple components or "elements" interconnected at nodes. It was at this stage in the development of the method that the term "finite element" appeared. However, unlike framed structures, closed form solutions to the differential equations governing the behavior of continuum elements were not available. Energy principles such as the theorem of virtual work or the principle of minimum potential energy, which were well known, combined with a piece-wise polynomial interpolation of the unknown displacement, were used to establish the matrix relationship between the forces and the interpolated displacements at the nodes numerically. In the late 1960s, when the method was recognized as being equivalent to a minimization process, it was reformulated in the form of weighted residuals and variational calculus and expanded to the simulation of nonstructural problems in fluids, thermomechanics and electromagnetics.

II. LINEAR QUADRILATERAL 4-NODE ELEMENT

In the quadrilateral family of elements, except for the square or the rectangle, it is impossible to construct the shape functions directly in terms of x and y. The only way to construct these functions is to use a reference element, which is a square of side 2 (units) as represented in Figure 1.

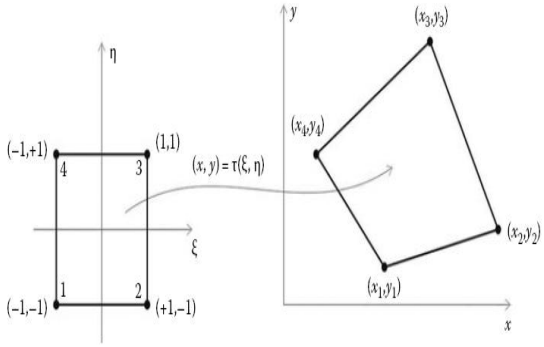


Figure 1: Geometrical transformation of 4-node element (Khennane 2013)

To define the geometrical transformation, we will assume that the coordinates (x, y) of an arbitrary point of the parent element are the unknown functions defined over the domain represented by the reference element in its local coordinate system (ξ, η). Notice that both the variables x and y belong to the linear class of functions since they are continuous and their first derivatives are constant equal to 1. Therefore, we start by constructing a general approximation for x in terms of ξ and η.

$$x = \alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\xi\eta \quad (1)$$

$$x = [1 \quad \xi \quad \eta \quad \xi\eta] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} \quad (2)$$

Then, we will transform the general approximation, equation (2), to a nodal approximation by using the nodal values x_1, x_2, x_3, and x_4 respectively at nodes 1, 2, 3, and 4. Notice also that the couple (ξ, η) takes on the values of (−1, −1), (1, −1), (1, 1), and (−1, 1) respectively at nodes 1, 2, 3, and 4. It follows

$$x_1 = \alpha_1 - \alpha_2 - \alpha_3 + \alpha_4$$
$$x_2 = \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4$$
$$x_3 = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$$
$$x_4 = \alpha_1 - \alpha_2 + \alpha_3 - \alpha_4$$

which, when rewritten in a matrix form, yields

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} \quad (3)$$

or in a more compact form as

$$\{X\} = [A]\{\alpha\} \quad (4)$$

The parameters α_i can be obtained easily by solving the equation (3). It can be noticed that the columns of the matrix $[A]$ are actually orthogonal vectors of norm 4. Hence, the inverse of the matrix $[A]$ is obtained as

$$[A]^{-1} = 1/4[A]^T = 1/4\begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \quad (5)$$

and the parameters α_i as

$$\begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} = 1/4\begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix}\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} \quad (6)$$

Substituting for the parameters α_i in equation (2) yields

$$x(\xi,\eta) = 1/4[1 \quad \xi \quad \eta \quad \xi\eta]\begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix}\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} \quad (7)$$

Expanding and rearranging equation (7) leads to

$$x(\xi, \eta) = N_1(\xi, \eta)x_1 + N_2(\xi, \eta)x_2 + N_3(\xi, \eta)x_3 + N_4(\xi, \eta)x_4 \quad (8)$$

$$N_1(\xi, \eta) = 0.25(1 - \xi - \eta + \xi\eta)$$
$$N_2(\xi, \eta) = 0.25(1 + \xi - \eta - \xi\eta)$$
$$N_3(\xi, \eta) = 0.25(1 + \xi + \eta + \xi\eta)$$
$$N_4(\xi, \eta) = 0.25(1 - \xi + \eta - \xi\eta)$$

Following exactly the same process for the variable y, we obtain

$$y(\xi, \eta) = N_1(\xi, \eta)y_1 + N_2(\xi, \eta)y_2 + N_3(\xi, \eta)y_3 + N_4(\xi, \eta)y_4 \quad (9)$$

Expressions (3) and (5) represent well and truly a linear geometrical transformation. The center of the reference square is given by $(\xi, \eta) = (0, 0)$.

III. FINITE ELEMENT FORMULATION FOR PLANE STRESS PROBLEMS

The stress–strain relationships for plane stress given as

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{Bmatrix} = \frac{E}{1-\mu^2}\begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \frac{(1-\mu)}{2} \end{bmatrix}\begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{xy} \end{Bmatrix} \quad (10)$$

$$\{\sigma\} = [D]\{\varepsilon\} \quad (11)$$

Whether it is a state of plane stress or plane strain, a material point can only move in the directions x and y. Therefore, the two displacement variables that play a role are u(x, y) and v(x,y). The infinitesimal strain displacements relations for both theories are the same and they are given as

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} \quad (12)$$

$$\varepsilon_{yy} = \frac{\partial v}{\partial y} \quad (13)$$

$$\varepsilon_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \quad (14)$$

These relations can be written in a matrix form as

$$\begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{xy} \end{Bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix}\begin{Bmatrix} u \\ v \end{Bmatrix} \quad (15)$$

or in a more compact form as

$$\{\varepsilon\} = [L]U \quad (16)$$

Where $[L]$ is a linear differential operator. Let us consider a finite element approximation for the unknown functions u and v. For an element having n nodes, the unknown displacements are interpolated using nodal approximations as

$$u = N_1u_1 + N_2u_2 + \cdots + N_nu_n$$
$$v = N_1v_1 + N_2v_2 + \cdots + N_nv_n$$

which, when written in a matrix form, yields

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & \cdots & N_n & 0 \\ 0 & N_1 & 0 & N_2 & \cdots & 0 & N_n \end{bmatrix}\begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_n \\ v_n \end{Bmatrix} \quad (17)$$

or simply as

$$\{U\} = [N]\{a\} \quad (18)$$

with $\{a\} = \{u_1, v_1, u_2, v_2, \ldots, u_n, v_n\}$ being the vector of nodal displacements. The number and the form of the shape functions depend on the element used.

Substituting for $\{U\}$ using equation (15), the strain displacement equation (16) become

$$\{\varepsilon\} = [B]\{a\} \quad (19)$$

where

$$[B] = \begin{bmatrix} \frac{\partial N_1}{\partial} & 0 & \frac{\partial N_2}{\partial} & 0 & \cdots & \frac{\partial N_n}{\partial} & 0 \\ 0 & \frac{\partial N_1}{\partial} & 0 & \frac{\partial N_2}{\partial} & \cdots & 0 & \frac{\partial N_n}{\partial} \\ \frac{\partial N_1}{\partial} & \frac{\partial N_1}{\partial} & \frac{\partial N_2}{\partial} & \frac{\partial N_2}{\partial} & \cdots & \frac{\partial N_n}{\partial} & \frac{\partial N_n}{\partial} \end{bmatrix} \quad (20)$$

The matrix $[B]$ is called the strain matrix; it relates the nodal displacements to the strains. It is formed by the partial derivatives of the shape functions $N_i(x, y)$.

IV. DISPLACEMENT FIELD

The displacement field over the element is approximated as

$$u = N_1u_1 + N_2u_2 + N_3u_3 + N_4u_4$$
$$v = N_1v_1 + N_2v_2 + N_3v_3 + N_4v_4$$

or in a matrix form as

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix}\begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} \quad (21)$$

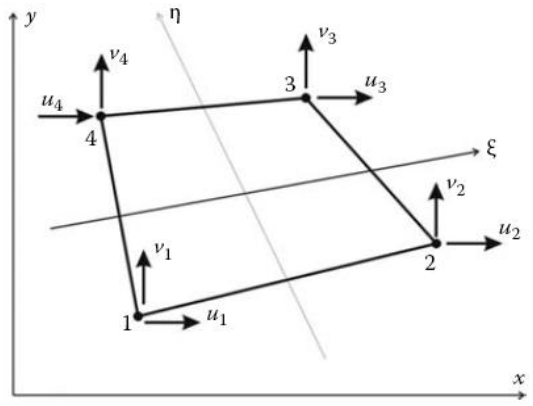


Figure 2: Linear 4-node quadrilateral element (Khennane 2013)

or more compactly as

$$\{U\} = [N]\{a\} \tag{22}$$

The element is isoparametric, therefore the shape functions $N_i(\xi, \eta)$ also define the geometrical transformation between the reference and the parent element. The coordinates *x* and *y* of any point of the parent element are given as

$$x = N_1x_1 + N_2x_2 + N_3x_3 + N_4x_4 \tag{23}$$
$$y = N_1y_1 + N_2y_2 + N_3y_3 + N_4y_4 \tag{24}$$

Subsequently, we will need to express the derivatives of a function in x and y coordinates in terms of its derivatives in ξ and η coordinates. This is done as follows

$$\frac{\partial N}{\partial \xi} = \frac{\partial N}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial N}{\partial y}\frac{\partial y}{\partial \xi} \tag{25}$$

$$\frac{\partial N}{\partial \eta} = \frac{\partial N}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial N}{\partial y}\frac{\partial y}{\partial \eta} \tag{26}$$

$$\begin{Bmatrix} \frac{\partial N}{\partial \xi} \\ \frac{\partial N}{\partial \eta} \end{Bmatrix} = J \begin{Bmatrix} \frac{\partial N}{\partial x} \\ \frac{\partial N}{\partial y} \end{Bmatrix} \tag{27}$$

Where J is Jacobian transformation matrix which is given as

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{4} \frac{\partial N_i}{\partial \xi} x_i & \sum_{i=1}^{4} \frac{\partial N_i}{\partial \xi} y_i \\ \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} x_i & \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} y_i \end{bmatrix} \tag{28a}$$

After deriving and rearranging, the Jacobian is written in the form of a product of two matrices:

$$[J] = 1/4 \begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \tag{28b}$$

A. Strain Matrix

Substituting for the displacements *u* and *v* in equation (16) using equation (22), the strain vector is obtained as

$$\{\varepsilon\} = [B]\{a\} \tag{29}$$

$$[B] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{bmatrix}$$

To evaluate the matrix [*B*], it is necessary to relate the partial derivatives in the (x, y) coordinates to the local coordinates (ξ, η). The derivative of the shape functions can be written as follows using the chain rule:

$$\frac{\partial N_i}{\partial \xi} = \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \xi} \tag{30}$$
$$\frac{\partial N_i}{\partial \eta} = \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \eta} \tag{31}$$

The derivatives of the shape functions in the (x, y) system is obtained by inversing the previous equation:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J]^{-1} \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{32}$$

B. Stiffness Matrix

The stiffness matrix for the quadrilateral element can be derived from the strain energy in the body, given by

$$U = \frac{1}{2}\int \sigma^T \varepsilon dv \tag{33}$$
$$U = \sum \frac{1}{2} t_e \int \sigma^T \varepsilon dA \tag{34}$$
$$\{\sigma\} = [D][B]\{a\} \tag{35}$$
$$\{\varepsilon\} = [B]\{a\} \tag{36}$$

where t_e is the thickness of element e

$$U = \sum \frac{1}{2} t_e \int_{-1}^{1}\int_{-1}^{1} \{a\}[B]^T[D][B]\{a\} dxdy \tag{37}$$
$$U = \sum \frac{1}{2} t_e \int_{-1}^{1}\int_{-1}^{1} \{a\}[B]^T[D][B]\{a\} detJ d\xi d\eta \tag{38}$$
$$= \frac{1}{2}\sum a^T k a \tag{39}$$

where

$$k = t_e \int_{-1}^{1}\int_{-1}^{1} [B]^T[D][B] detJ d\xi d\eta \tag{40}$$

is the element stiffness matrix of dimension (8x8).

C. Numerical Integration

The Gaussian quadrature approach is used for integration of quadrilateral element. Consider the n-point approximation

$$I = \int_{-1}^{1} f(\xi) d\xi \approx w_1 f(\xi_1) + w_2 f(\xi_2) + \cdots . + w_n f(\xi_n) \tag{41}$$

Where $w_1, w_2, \ldots$, and w_n are the weights and ξ_1, ξ_2, …., and ξ_n are the sampling points or gauss points. The idea behind Gaussian quadrature is to select the n Gauss points and n weight such that equation 3.51 provides an exact answer or polynomials f(ξ) of as large a degree as possible. In other words, the idea is that if the n-point integration formula is exact for all polynomial. In our work we have two variables so we used two point formula such as

$$\int_{-1}^{1} f(\xi) d\xi \approx w_1 f(\xi_1) + w_2 f(\xi_2) \tag{42}$$

We have four parameters to choose: w1, w2, ξ1 and ξ2. We can therefore accept the formula in equation (42) to be exact for a cubic polynomial. Thus choosing $f(\xi)=a_o+a_1\xi+a_2\xi^2+a_3\xi^3$ yields

$$Error = \left[\int_{-1}^{1}(a_o + a_1\xi + a_2\xi^2 + a_3\xi^3)\right] - \left[w_1 f(\xi_1) + w2f\xi2\right] \tag{42}$$

Requiring zero error yields

$$w_1+w_2=2$$
$$w_1\xi_1+w_2\xi_2=0$$
$$w_1\xi_1^2+w_2\xi_2^2=2/3$$
$$w_1\xi_1^3+w_2\xi_2^3=0$$

These nonlinear equations have the unique solution

$w_1 = w_2 = 1$ $\quad -\xi_1 = \xi_2 = 1/\sqrt{3} = 0.57735022691\ldots..$

These values are directly used in the finite element analysis for 4-nodes quadrilateral element.

V. MATLAB CODE IMPLEMENTATION

In present investigation a compact MATLAB code for structural analysis of cantilever beam and Messerschmitt Bolkow Blohm Beam subjected to static and self-weight loading has been performed. The Matlab code (see the Appendix), is built up as a standard topology optimization code. The main program is called from the Matlab prompt by the line

function FEM (Nx,Ny,CANTILEVER,MMB,GRAVITY)

Where *nelx* and *nely* are number of elements in the horizontal and vertical directions, respectively. *CANTILEVER* and *MBB* is used for analysis of cantilever beam and Messerschmitt Bolkow Blohm beam respectively. *GRAVITY* is used when structure is analyzed under self weight. Other variables as well as boundary conditions are defined in the MATLAB code itself and can be edited if needed. The Compete MATLAB code for static and self-weight loading conditions are described in appendix. The description of MATLAB code (cantilever beam under static loading meshing by 4-node elements) written is as follow:

- Nodal coordinate generation (line 4-14)
- Nodal connectivity generation (line 15-28)
- Input value used in FEM (line 29-42)
- Calculating stiffness matrix of body (line 43-74)
- Defining boundary condition force applied and calculating displacement of body (line 75-119)
- Plotting final result obtained (line 120-169)

1. For structure analysis of cantilever beam under static loading condition as shown in Figure 4 (b) input

FEM(32,20,1,0,0)

2. For structure analysis of cantilever beam under self-weight loading condition as shown in Figure 5 (b) input

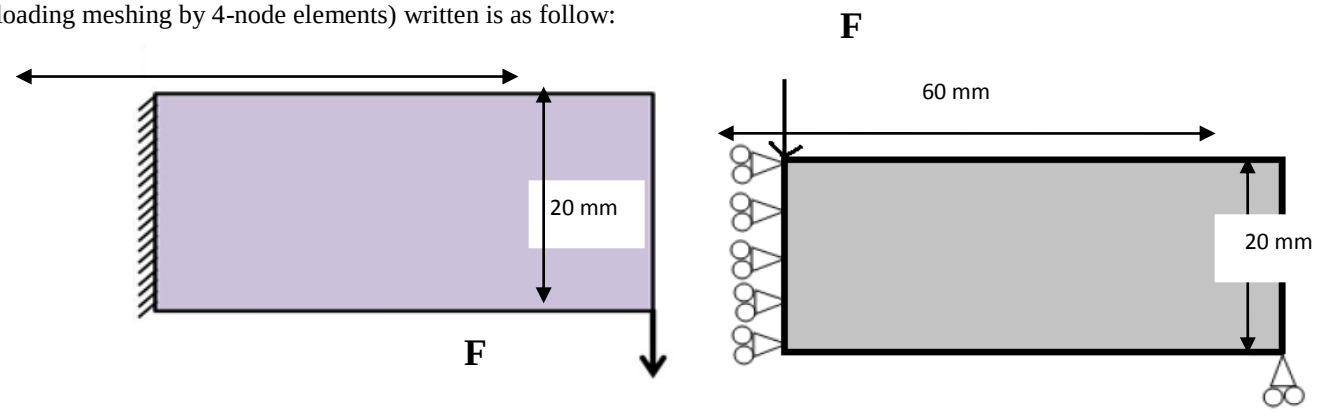
FEM(32,20,0,0,11)

3. For structure analysis of MBB beam under static loading condition as shown in Figure 6 (b) input

FEM(60,20,0,1,0)

4. For structure analysis of MBB beam under self-weight loading condition as shown in Figure 7 (b) input

FEM(60,20,0,0,12)

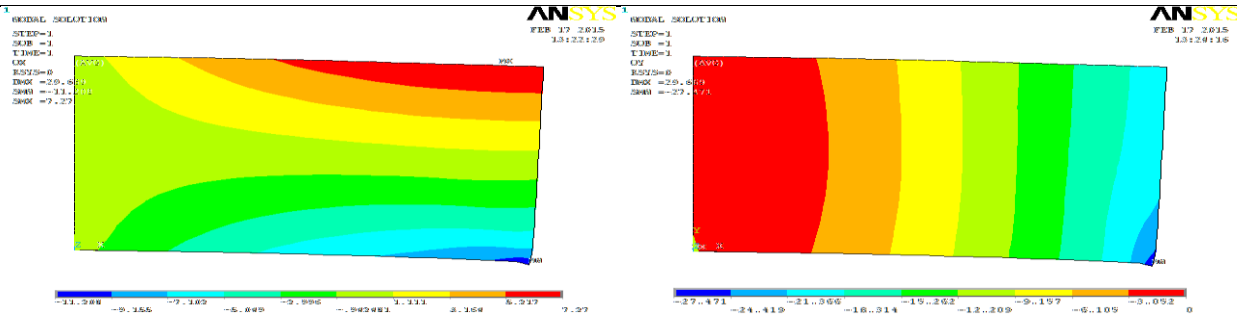


(a) **(b)**

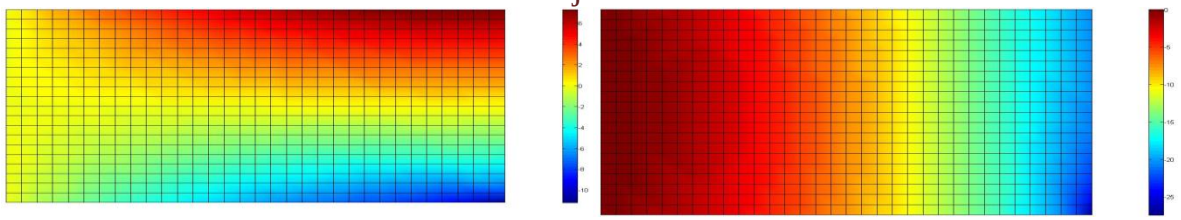
Figure 3 Geometry and Boundary Conditions (a) cantilever beam (b) MBB beam

Table 1: Maximum deformation in x and y directions under point load

CANTILEVER BEAM				MBB BEAM			
ANSYS CANTILEVER BEAM		MATLAB CANTILEVER BEAM		ANSYS MBB BEAM		MATLAB MBB BEAM	
UX	11.208 mm	**UX**	30.545 mm	**UX**	30.545 mm	**UX**	30.545 mm
UY	27.471 mm	**UY**	125.88 mm	**UY**	125.88 mm	**UY**	125.88 mm
MATERIAL PROPERTIES				E =1		ν =0.3	

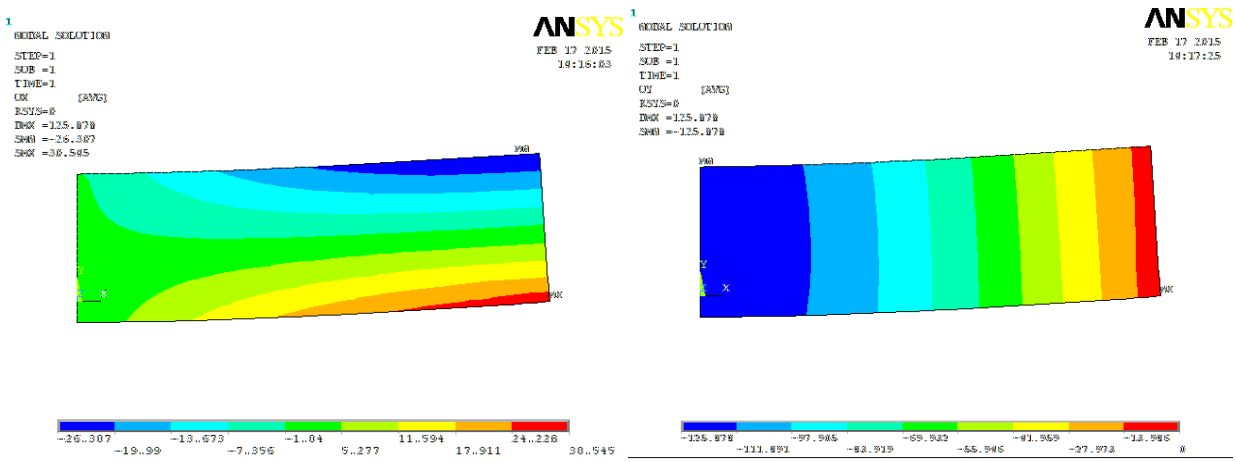


(a) DISPLACEMENT UX DISPLACEMENT UY

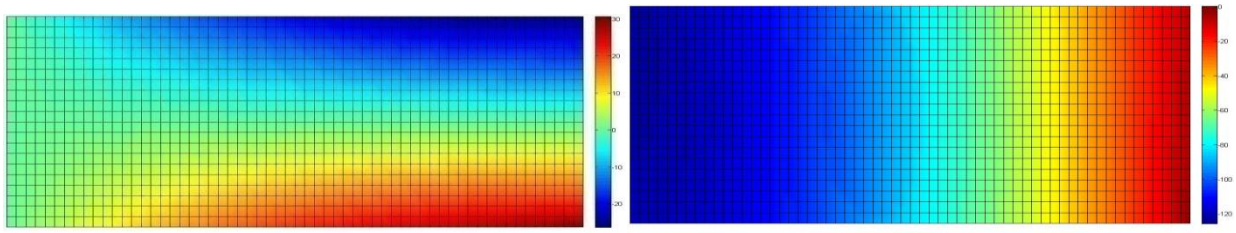


(b) DISPLACEMENT UX DISPLACEMENT UY

Figure 4: Deformation of catiliver beam under point load (a) ANSYS (b) MATLAB



(a) DISPLACEMENT UX DISPLACEMENT UY

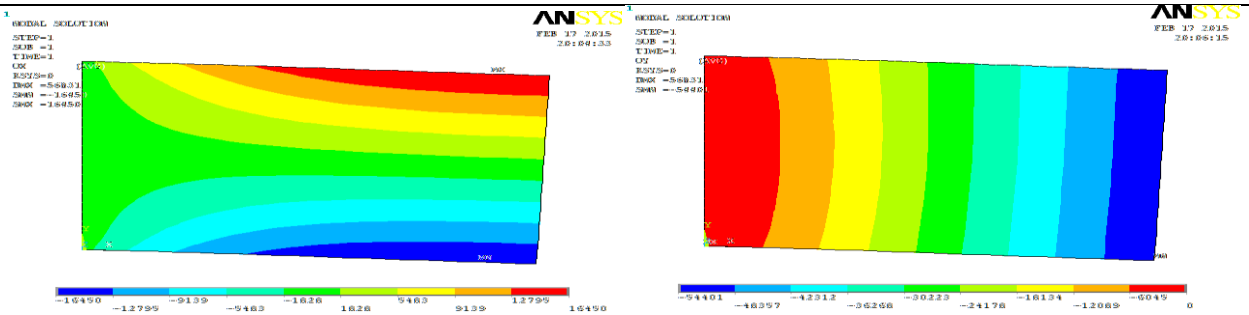


(b) DISPLACEMENT UX DISPLACEMENT UY

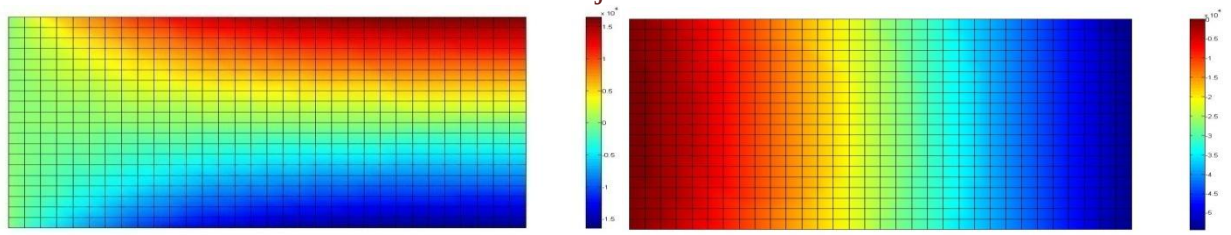
Figure 5: Deformation of MBB beam under point load (a) ANSYS (b) MATLAB

Table 1: Maximum deformation in x and y directions self-weight load

CANTILEVER BEAM				MBB BEAM			
ANSYS		MATLAB		ANSYS		MATLAB	
CANTILEVER BEAM		CANTILEVER BEAM		MBB BEAM		MBB BEAM	
UX	16450	**UX**	16450	UX	2.5849×10^5	**UX**	2.5849×10^5
UY	54401	**UY**	54401	UY	9.4256×10^5	**UY**	9.4256×10^5
MATERIAL PROPERTIES		$\rho=1$ $g=10$ $E=1$ $\nu=0.3$					

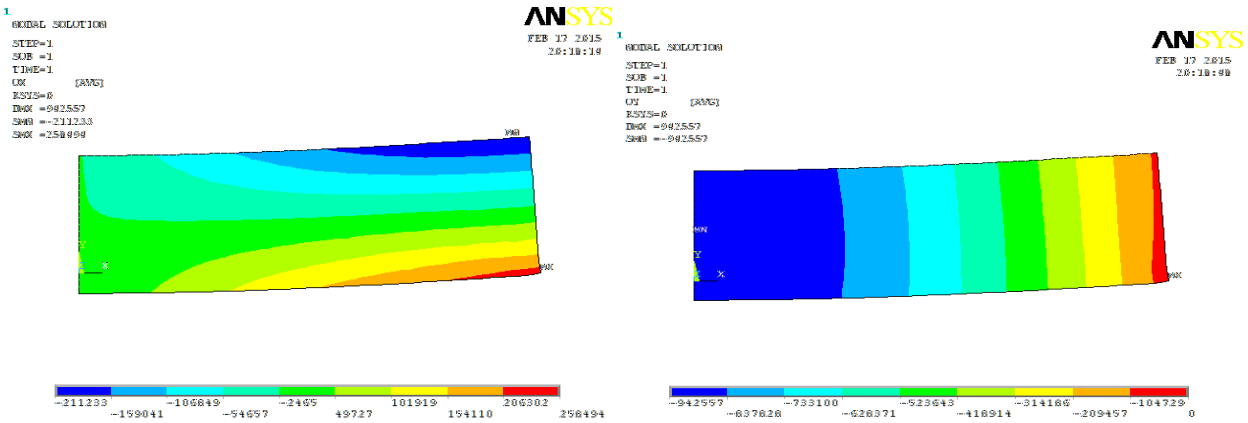


(a) DISPLACEMENT UX DISPLACEMENT UY

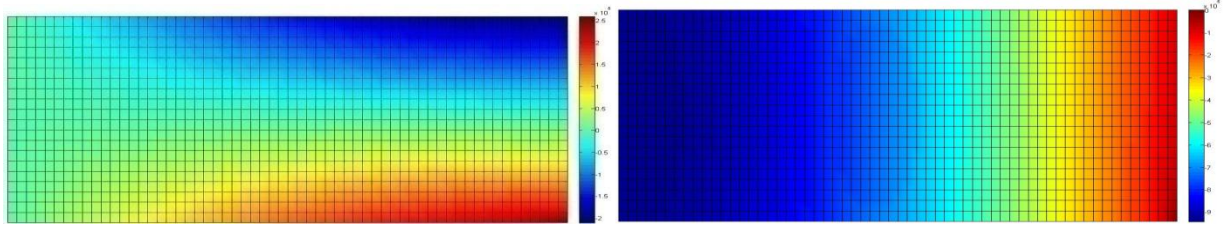


(b) DISPLACEMENT UX DISPLACEMENT UY

Figure 6: Deformation of cantiliver beam under self-weight load (a) ANSYS (b) MATLAB



(a) DISPLACEMENT UX DISPLACEMENT UY



(b) DISPLACEMENT UX DISPLACEMENT UY

Figure 7: Deformation of MBB beam under self-weight load (a) ANSYS (b) MATLAB

CONCLUSION

In present investigation MATLAB code for structural analysis of 2-D linear elastic isotropic structures i.e. cantilever beam and MBB beam is presented. For both the beam maximum displacement obtained in x & y directions with MATLAB code are been compared with the ANSYS and from above result we conclude that the accuracy of MATLAB code is same as that of ANSYS software for structural analysis of above numerical examples (cantilever beam and MBB beam). Another application of FEM MATLAB code is for self-weight analysis in which weight of the structure also include in structure analysis of structures which also equally efficient to ANSYS software.

APPENDIX

```
function FEM (Nx,Ny,CANTILEVER,MMB,GRAVITY)
nelx=Nx;
nely=Ny;
% Input data for nodal coordinate values
coordinates=zeros((nely+1)*(nelx+1),2);
y=[nely:-1:0];
x=[0:1:nelx];
n=0;
for i=1:(nelx+1)
for j=1:(nely+1)
coordinates((n+j),1)=x(i);
coordinates((n+j),2)=y(j);
end
n=n+(nely+1);
end
% Input data for nodal connectivity for each element
nodes=zeros(nelx*nely,4);
B=[1:nely];
t=[2:(nely+1)];
m=[0:nely:nely*(nelx-1)];
n=0;
for j=1:nelx
for i=1:nely
nodes((i+m(j)),1)=B(i)+n;
nodes((i+m(j)),2)=B(i)+n+(nely+1);
nodes((i+m(j)),3)=t(i)+n+(nely+1);
nodes((i+m(j)),4)=t(i)+n;
end
n=n+(nely+1);
end
% Input data
nel = length(nodes);
nnel=4;
ndof=2;
nnode = length(coordinates) ;
 sdof=nnode*ndof;
```

```
edof=nnel*ndof;
stiffness = zeros(sdof,sdof);
displacement=zeros(sdof,1);
force = zeros(sdof,1);
E = 2.1*10^9;
density=7850;
gravity=9.81;
F=(density*gravity);
nu = 0.3;                                   % Poisson's ratio
%  Computation of element matrices and vectors and their assembly
D = E/(1-nu^2)*[1 nu 0 ; nu 1 0; 0 0 (1-nu)/2] ;
Gausspointx=[-1 1 1 -1]/sqrt(3);
Gausspointy=[1 1 -1 -1]/sqrt(3);
Gaussweight=[1 1 1 1];
for iel=1:nel
for i=1:nnel
nd(i)=nodes(iel,i);
xx(i)=coordinates(nd(i),1);
yy(i)=coordinates(nd(i),2);
end
K = zeros(edof,edof);
for int=1:4
xi = Gausspointx(int);
wtx = Gaussweight(int);
eta = Gausspointy(int);
wty = Gaussweight(int) ;
% Compute isoparametric four-node Quadrilateral shape   functions and
their derivatives
j=0.25*[-(1-eta)  (1-eta)   (1+eta)  -(1+eta);...
                (1-xi)    (1+xi)   -(1+xi)   -(1-xi)];
J=j*[xx;yy]';
A=[J(2,2) -J(1,2) 0 0;
      0 0 -J(2,1) J(1,1);
    -J(2,1) J(1,1) J(2,2) -J(1,2)]*(1/det(J));
G=(1/4)*[-(1-eta)    0   (1-eta)   0  (1+eta)   0 -(1+eta)   0;
                 (1-xi)    0    (1+xi)   0   -(1+xi)   0    -(1-xi)   0;
                0      -(1-eta)     0    (1-eta)  0 (1+eta) 0 -(1+eta);
                0       (1-xi)       0    (1+xi)   0 -(1+xi)  0 -(1-xi)];
B=A*G;
K=K+B'*D*B*wtx*wty*det(J);
end
EDOF   =   [2*nd(1)-1;   2*nd(1);   2*nd(2)-1;   2*nd(2);   2*nd(3)-1;
   2*nd(3); 2*nd(4)-1; 2*nd(4)];
stiffness(EDOF,EDOF)=stiffness(EDOF,EDOF)+K;
end
% constrain, boundary condition and force vector
%-----------------------external force-----------------------------
  if (CANTILEVER==1)
  force(2*(nelx+1)*(nely+1)) = -1 ;
  displacement((2*(nely+1)+1):sdof)                                    =
   stiffness((2*(nely+1)+1):sdof,(2*(nely+1)+1):sdof)\force((2*(nely
   +1)+1):sdof);
end
%-----------------------gravity force-------------------------------
  if (MMB==1)
  force(2,1) = -1;
  fixeddofs   = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
  alldofs     = [1:2*(nely+1)*(nelx+1)];
  freedofs    = setdiff(alldofs,fixeddofs);
  displacement(freedofs,:)      =      stiffness(freedofs,freedofs)      \
   force(freedofs,:);
  displacement(fixeddofs,:)= 0;
  end
%-----------------------gravity force-------------------------------
  if (GRAVITY>0)
  W=[1 (nely+1) ((nely+1)*(nelx+1)-nely) (nely+1)*(nelx+1)];
  w1=[2:nely];
  w2=[(nely+2):(nely+1):(nely+1)*(nelx)];
  w3=[(2*(nely+1)):(nely+1):(nely+1)*(nelx)];
  w4=[((nely+1)*(nelx)+2):((nely+1)*(nelx+1)-1)];
  w5=[1:(nely+1)*(nelx+1)];
  W1=union(w1,w2);
  W2=union(w3,w4);
  W3=union(W1,W2);
  W4=setdiff(w5,W3);
  W5=setdiff(W4,W);
  force(2*W')=-F/4;
  force(2*W3')=-F/2;
  force(2*W5')=-F;
  if (GRAVITY==11)
  displacement((2*(nely+1)+1):sdof) =
   stiffness((2*(nely+1)+1):sdof,(2*(nely+1)+1):sdof)\force((2*(nely
   +1)+1):sdof);
  end
  if (GRAVITY==12)
  fixeddofs   = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
  alldofs     = [1:2*(nely+1)*(nelx+1)];
  freedofs    = setdiff(alldofs,fixeddofs);
  displacement(freedofs,:) = stiffness(freedofs,freedofs) \
   force(freedofs,:);
  displacement(fixeddofs,:)= 0;
  end
  end
% solution
  UX = displacement(1:2:sdof) ;
  UY = displacement(2:2:sdof)
  disp('The maximum displacement UX')
  max(abs(UX))
  disp('The maximum displacement UY')
  max(abs(UY))
  for iel=1:nel
     for i=1:nnel

     nd(i)=nodes(iel,i);          X(i,iel)=coordinates(nd(i),1);
   Y(i,iel)=coordinates(nd(i),2);
     end
     profile1(:,iel) = -UX(nd') ;
     profile2(:,iel) = -UY(nd') ;
  end
  % Plotting the FEM mesh and profile of the given component
     f3 = figure ;
     set(f3,'name','Postprocessing','numbertitle','off') ;
     plot(X,Y,'k')
     fill(X,Y,profile1)
     axis off ;
      cbar = colorbar;
     set(figure,'name','Postprocessing','numbertitle','off') ;
     plot(X,Y,'k')
     fill(X,Y,profile2)
     axis off ;
     cbar = colorbar;
```

References

[1] Chandrupatla, T. R. and Belegunda, A. D. 2004. Introduction to finite element in engineering, Prentice hall upper Inc.

[2] Hutton, D. V. 2004. Fundamental of finite element analysis, The McGraw-Hill companies

[3] Khennane, A. 2013. Introduction to finite element analysis using MATLAB and Abaqus, Taylor & Francis Group, LLC

[4] Ain, J.E 1982. Application and implementation of finite element methods, London:Academic.

[5] Baker, A.J. and Pepper D.W., 1988. Finite element analysis on microcomputers. New York:McGraw-Hill.

[6] Lepi, S.M., 1998. Practical guide to finite elements: a solid mechanics approach, Marcel Dekker.
[7] Smith I.M. and Griffiths D.V. Programming the Finite Element Method, 2nd edn. Wiley, Chichester, U.K., 1988.
[8] Timoshenko S. and Goodier J. Theory of Elasticity, 3rd edn. McGraw-Hill, New York, 1970.
[9] Timoshenko S. and Woinowsky-Krieger S. Theory of Plates and Shells. McGraw-Hill, New York, 1959.
[10] Zienkiewicz O.C. The Finite Element Method, 3rd edn. McGraw-Hill, York, London, 1977.
[11] Cook R.D. Finite Element Modeling for Stress Analysis. Wiley, New York, 1995.
[12] Kwon Y.W. and Bang H. The Finite Element Method Using Matlab, 2nd edn. CRC Press, London, U.K., 2000.
[13] Logan D.L. A First Course in the Finite Element Method Using Algor, 2nd edn. Brooks/Cole Thompson Learning, Pacific Groove, CA, 2001.
[14] Mase G.E. Schaum's Outline Series: Theory and Problems of Continuum Mechanics. McGraw-Hill, New York, 1970.
[15] McGuire M., Gallagher G.H., and Ziemian R.D. Matrix Structural Analysis, 2nd edn. Wiley, New York, 2000.
[16] Meek J.L. Computer Methods in Structural Analysis. E & FN SPON, London, U.K., 1991.
[17] Reddy J.N. An Introduction to the Finite Element Method, 3rd edn. McGraw-Hill, New York, 2006.
[18] Saada A.S. Elasticity: Theory and Applications, 2nd edn. Krieger Publishing, Melbourne, FL, 1993.