MARMARA UNIVERSITY
FACULTY OF ENGINEERING

DEVELOPLMENT OF A COMPUTER PROGRAM FOR THE FINITE ELEMENT ANALYSIS AND DESIGN OF PLANAR FRAME STRUCTURES

EMRE BAŞAR

GRADUATION PROJECT REPORT
Department of Mechanical Engineering

Supervisor
Prof. Dr. Mustafa Özdemir

ISTANBUL, 2024

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

DEVELOPMENT OF A COMPUTER PROGRAM FOR THE FINITE ELEMENT ANALYSIS AND DESIGN OF PLANAR FRAME STRUCTURES

by

Emre BAŞAR

May 28, 2024, Istanbul

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERINGIN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

Signature of Author(s) .....Emre BAŞAR

Department of Mechanical Engineering

Certified By ..... Prof. Dr. Mustafa ÖZDEMİR.

Project Supervisor, Department of Mechanical Engineering

Accepted By.....Prof. Dr. Bülent EKİCİ

Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENT

First of all, I would like to thank my supervisor Prof. Dr. Mustafa ÖZDEMİR, for the valuable guidance and advice on preparing this thesis and giving me moral and material support.

May, 2024 Emre BAŞAR

Contents

ACKNOWLEDGEMENT i
ÖZET iii
ABSTRACT iv
SYMBOLS v
LIST OF FIGURES vi
1. HISTORICAL BACKGROUND 1
2. INTRODUCTION TO FINITE ELEMENT ANALYSIS 2
2.1. Spring 2
2.2. Bar Element 3
2.3. Planar Truss 5
2.4. Spatial Truss 7
2.5. Beam Element 8
2.6. Plane Frame 11
2.7. Spatial Frame 13
3. PROGRAM DESIGN PROCEDURE 18
3.1 MATLAB 18
3.2 Getting Number of Nodes and Elements 18
3.3 Getting Coordinates of Nodes and Plot the Frame 19
3.4 Getting Element Properties and Calculating Angle and Length 19
3.5 Element Stiffness Matrix Function 21
3.6 Assembly Function for Global Stiffness Matrix 22
3.7 Boundary Conditions for Nodes 25
3.8 Storing Boundary Conditions, Global Stiffness Matrix 27
3.9 Solving Unknown Displacements and Forces 28
3.10 Obtaining Element Forces with respect to Displacements 30
3.11 Functions for Plot Diagrams 31
3.12 Getting Diagrams 32
4. SOLVING AN EXAMPLE 33
5. CONCLUSION 48
REFERENCES 49

ÖZET

Sonlu elemanlar analizi ve düzlemsel çerçeve yapıların tasarımı için bir bilgisayar programının geliştirilmesi

Sistemleri analiz ederken, matematiksel bir denklem türetmeye çalışırız ve denklemleri yazarken bazı varsayımlarda bulunuruz. Böylelikle işimiz kolaylaşmış olur. Ancak kurulan bu denklemler, içerisinde diferansiyel denklemleri de barındırır ve sistemlerin davranışlarını açıklayan bu denklemleri çözmek zordur. Sonlu elemanlar analizi bu denklemleri çözmek üzere geliştirilmiş bir yoldur. Sonlu elemanlar analizi denklemi türetilecek sistemi küçük bileşenlere ayırarak çözmeye yarar. En büyük avantajlarından biri de bu tür problemleri çözebilecek bir bilgisayar programı geliştirebilmesidir. Ben de bu projede sonlu elemanlar analizi ve düzlemsel çerçeve yapıların tasarımı için bir bilgisayar programı geliştirmeye çalıştım.

ABSTRACT

Development of a computer program for finite element analysis and design of planar frame structures

When analyzing systems, we try to derive a mathematical equation and make some assumptions when writing the equations. This makes our job easier. However, these established equations also include differential equations, and it is difficult to solve these equations that explain the behavior of the systems. Finite element analysis is a way to solve these equations. Finite element analysis is used to solve the equation by dividing the system to be derived into small components. One of its biggest advantages is the ability to develop a computer program that can solve such problems. In this project, I tried to develop a computer program for finite element analysis and design of planar frame structures.

SYMBOLS

A: Cross sectional area, m^2

E: Modulus of elasticity, Pa

$f_n^{(k)}$: Nodal force at nth node of kth element, N

F_n: Global nodal force, N

G: Shear modulus, Pa

I_n: Area moment of inertia around n axis, m^4

J: Polar moment of inertia, m^4

$[K_e]$: Global stiffness matrix, N/m

k_e: Stiffness matrix in element coordinate system, N/m

k_n: Stiffness of nth element, N/m

L: Length, m

M_n: Bending moment, Nm

N: Interpolation function

P: Axial Load, N

[R]: Transformation matrix

T: Torque, Nm

$u_n^{(k)}$: Nodal displacement at nth node of kth element, m

U_n: Global nodal displacement, m

v_n: Transverse displacement at nth node, m

w_n: Nodal displacement at z axis at nth node, m

ε: Strain, m/m

σ: Stress, N/m^2

δ: Deflection, m

ξ: Dimensionless length coordinate

ρ: Radius of curvature, rad

ψ: Orientation angle, rad

θ_n: Rotation at nth element (Nodal slope), rad

LIST OF FIGURES

Figure 1: Circumference of circle [1] 1
Figure 2: Springs in series 2
Figure 3:Spring subsystem 2
Figure 4:Bar element with nodal displacement notation 4
Figure 5: Two-Dimensional Truss Element 5
Figure 6: Bar element in a 3-D global coordinate system [3] 7
Figure 7: Deflected Beam Element [3] 8
Figure 8: Beam element nodal displacements 9
Figure 9:Nodal Displacements in Local Coordinate System 11
Figure 10:Nodal Displacements in Global Coordinate System 12
Figure 11:Three-dimensional beam element 13
Figure 12:Nodal displacements in x-z plane 14
Figure 13: Circular cylinder subjected to torsion 14
Figure 14:Example of element stiffness matrix 22
Figure 15: Plane Frame with Three Elements for Example 8.1 [5] 33
Figure 16: Global stiffness matrix according to answer of book [5] 33
Figure 17: Global nodal displacement vector U according to answer of book [5] 34
Figure 18:Global nodal force vector F according to answer of book [5] 34
Figure 19: Number of nodes 34
Figure 20:Number of Elements 34
Figure 21: Asks Coordinates 35
Figure 22: Drawing of Frame 35
Figure 23:Element Properties 35
Figure 24: Asks Inclined Support 36
Figure 25: Boundary Condition Selection 36
Figure 26: Displacement/Rotation 36
Figure 27: Results 37
Figure 28: Axial force diagram of element 1 38
Figure 29: Axial force diagram of element 1 [5] 38
Figure 30: Shear force diagram of element 1 39
Figure 31: Shear force diagram of element 1 [5] 39
Figure 32: Bending moment diagram of element 1 40
Figure 33: Bending moment diagram of element 1 [5] 40
Figure 34: Axial force diagram of element 2 41
Figure 35: Axial force diagram of element 2 [5] 41
Figure 36: Shear force diagram of element 2 42
Figure 37: Shear force diagram of element 2 [5] 42
Figure 38: Bending moment diagram of element 2 43
Figure 39: Bending moment diagram of element 2 [5] 43
Figure 40: Axial force diagram of element 3 44
Figure 41: Axial force diagram of element 3 [5] 44
Figure 42: Shear force diagram of element 3 45
Figure 43: Shear force diagram of element 3 [5] 45
Figure 44: Bending moment diagram of element 3 46
Figure 45: Bending moment diagram of element 3 [5] 46
Figure 46:Global stiffness matrix 47

1. HISTORICAL BACKGROUND

Although finite element theory is very famous nowadays, it actually dates back to centuries. For instance, the circumference of a circle was approximated by perimeter of polygon.[1]

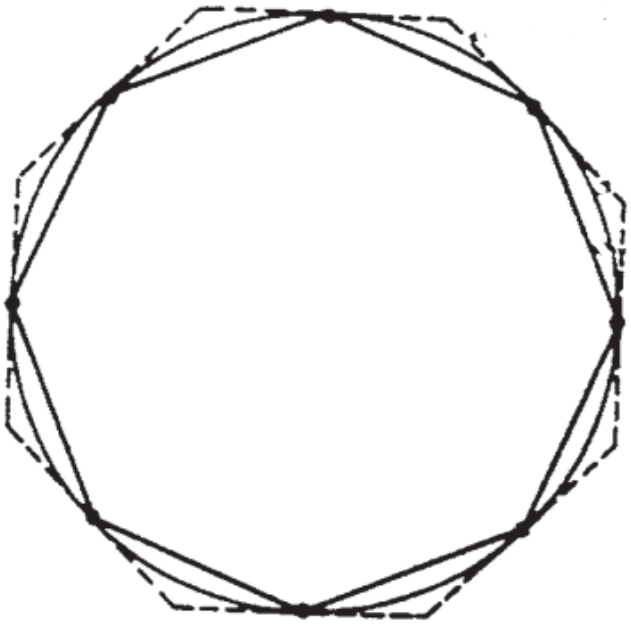

Figure 1: Circumference of circle [1]

In 1943, Courant has proposed the finite element method to literature of applied mathematics. But this suggestion did not consider until the other engineers work on it.[2]

In 1960, finite element term used in a presentation by Clough. He used this method for explain a certain plain stress. After that presentation, many scientists understood that Clough look at the problem from different view. They were also working on same problem but they were following a different path. This process provides concrete stair step to develop finite element method by bringing closer the analogue based approach and pure mathematical approach. Simulating and analyzing a wide variety of boundary value problems has been done by finite element method which thought as numerical, and mathematically well defined, discretization method.[2]

Finally, finite element method is very wide used method in many engineering problems.

2. INTRODUCTION TO FINITE ELEMENT ANALYSIS

2.1. Spring

Springs in series are very good way of introducing finite element method. So, in this part of report, matrix analysis of springs in series which is early finite element method and then assembly will be shown.

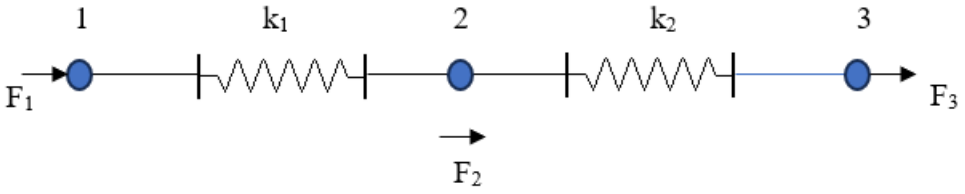

Figure 2: Springs in series

Divide this system to 2 subsystems:

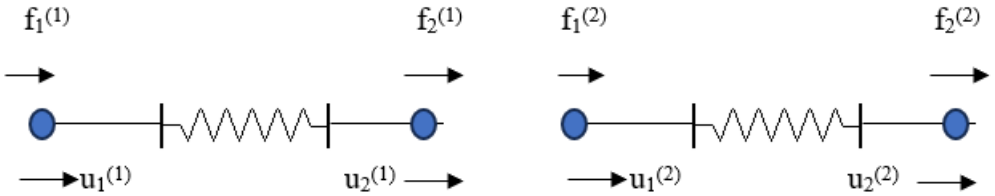

Figure 3:Spring subsystem

Then element stiffness matrix for first spring [3]:

$$\begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \begin{bmatrix} {u_1}^{(1)} \\ {u_2}^{(1)} \end{bmatrix} = \begin{bmatrix} {f_1}^{(1)} \\ {f_2}^{(1)} \end{bmatrix} \quad (1)$$

Element stiffness matrix for second spring [3]:

$$\begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{bmatrix} {u_1}^{(2)} \\ {u_2}^{(2)} \end{bmatrix} = \begin{bmatrix} {f_1}^{(2)} \\ {f_2}^{(2)} \end{bmatrix} \quad (2)$$

These founded values belong to local matrices. We need to assemble global matrix which includes all of them. First, make assumptions for assemble to global matrix:

$$F_1 = {f_1}^{(1)} \quad (3)$$

$$F_2 = {f_2}^{(1)} + {f_1}^{(2)} \quad (4)$$

$$F_3 = {f_2}^{(2)} \tag{5}$$

$$U_1 = {u_1}^{(1)} \tag{6}$$

$$U_2 = {u_2}^{(1)} = {u_1}^{(2)} \tag{7}$$

$$U_3 = {u_2}^{(2)} \tag{8}$$

According to equations (3), (4), (5), (6), (7), (8):

$$F_1 = k_1 U_1 - k_1 U_2 \tag{9}$$

$$F_2 = -k_1 U_1 + k_1 U_2 + k_2 U_2 - k_2 U_3 \tag{10}$$

$$F_3 = -k_2 U_2 + k_2 U_3 \tag{11}$$

Then global matrix becomes:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \tag{12}$$

There is also second way to find global matrix after we found element stiffness matrix. We give the position number to the element stiffness matrices and then get all of them together. First matrix:

$$\begin{bmatrix} (1,1) & (1,2) \\ (2,1) & (2,2) \end{bmatrix} \tag{13}$$

Second matrix:

$$\begin{bmatrix} (2,2) & (2,3) \\ (3,2) & (3,3) \end{bmatrix} \tag{14}$$

Then combining these 2 element matrices gives global matrix:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \tag{15}$$

2.2. Bar Element

Usage of spring element is limited. So, I will examine bar element. Because logic of bar element is similar to spring element. While I am examining the bar element, there is also assumptions which are

- Bar is geometrically straight.[3]

- The material obeys Hooke's law.[3]
- Forces are applied at the end of bar.[3]
- The bar supports axial loading only; bending, torsion, and shear are not transmitted to the element via the nature of its connections to other elements.[3]

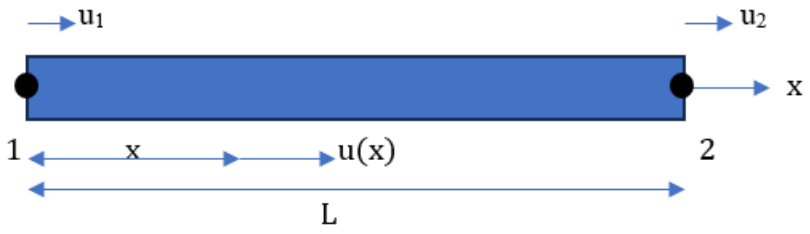

Figure 4:Bar element with nodal displacement notation

the deflection, δ, of an elastic bar of length L and uniform cross-sectional area A when subjected to axial load P is given by [4]: δ

$$\delta = \frac{PL}{AE} \tag{16}$$

where the E is modulus of elasticity.

When we substitute spring constant to (16):

$$k = \frac{P}{\delta} = \frac{AE}{L} \tag{17}$$

Strain [3]:

$$\varepsilon_x = \frac{(u_2 - u_1)}{L} \tag{18}$$

The axial stress by Hooke's law [3]:

$$\sigma_x = E * \varepsilon_x = E * \frac{(u_2 - u_1)}{L} \tag{19}$$

Equation (19) helps us to find force:

$$P = \sigma_x * A = \frac{AE}{L} * (u_2 - u_1) \tag{20}$$

We have found force equation. Now we can consider sign convention. Equation (20) states that f_2 is positive so f_2 will be compression. In that case, f_1 will be tension. So, f_1 is negative and f_2 is positive.

$$f_1 = -f_2 \tag{21}$$

$$f_1 = -\frac{AE}{L} * (u_2 - u_1) \tag{22}$$

$$f_2 = \frac{AE}{L} * (u_2 - u_1) \tag{23}$$

Expression in matrix form:

$$\frac{AE}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \tag{24}$$

Then stiffness matrix for bar element:

$$[k_e] = \frac{AE}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{25}$$

2.3. Planar Truss

Now, I will examine two-dimensional truss which is subjected axial forces only.

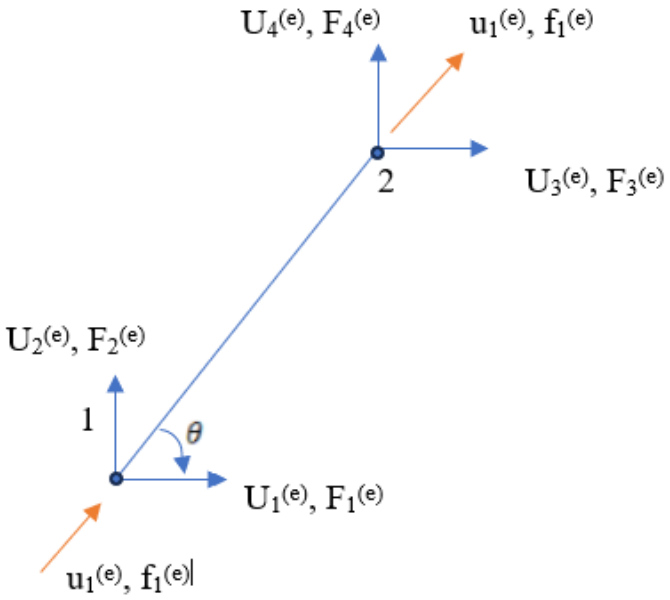

Figure 5: Two-Dimensional Truss Element

Then the relation between the global and element coordinate displacements [3]:

$$u_1{}^{(e)} = U_1{}^{(e)} \cos\theta + U_2{}^{(e)} \sin\theta \tag{26}$$

$$u_2{}^{(e)} = U_3{}^{(e)} \cos\theta + U_4{}^{(e)} \sin\theta \tag{27}$$

In matrix form:

$$\begin{bmatrix} u_1^{(e)} \\ u_2^{(e)} \end{bmatrix} = \begin{bmatrix} cos\,\theta & sin\,\theta & 0 & 0 \\ 0 & 0 & cos\,\theta & sin\,\theta \end{bmatrix} \begin{bmatrix} U_1^{(e)} \\ U_2^{(e)} \\ U_3^{(e)} \\ U_4^{(e)} \end{bmatrix} = [R] \begin{bmatrix} U_1^{(e)} \\ U_2^{(e)} \\ U_3^{(e)} \\ U_4^{(e)} \end{bmatrix} \quad (28)$$

[R] is transformation matrix

We can reach to force matrix from displacement matrix:

$$\begin{bmatrix} k_e & -k_e \\ -k_e & k_e \end{bmatrix} [R] \begin{bmatrix} U_1^{(e)} \\ U_2^{(e)} \\ U_3^{(e)} \\ U_4^{(e)} \end{bmatrix} = \begin{bmatrix} f_1^{(e)} \\ f_2^{(e)} \end{bmatrix} \quad (29)$$

Then combine it with transpose of transformation matrix for finding nodal force matrix in global frame [3]:

$$[R]^T \begin{bmatrix} k_e & -k_e \\ -k_e & k_e \end{bmatrix} [R] \begin{bmatrix} U_1^{(e)} \\ U_2^{(e)} \\ U_3^{(e)} \\ U_4^{(e)} \end{bmatrix} = \begin{bmatrix} f_1^{(e)}\,cos\,\theta \\ f_1^{(e)}\,sin\,\theta \\ f_2^{(e)}\,cos\,\theta \\ f_2^{(e)}\,sin\,\theta \end{bmatrix} = \begin{bmatrix} F_1^{(e)} \\ F_2^{(e)} \\ F_3^{(e)} \\ F_4^{(e)} \end{bmatrix} \quad (30)$$

Element stiffness matrix in global coordinate frame will be [3]:

$$[K^{(e)}] = [R]^T \begin{bmatrix} k_e & -k_e \\ -k_e & k_e \end{bmatrix} [R] \quad (31)$$

$$\left[K^{(e)}\right] = k_e \begin{bmatrix} (cos\,\theta)^2 & cos\,\theta\,sin\,\theta & -(cos\,\theta)^2 & -cos\,\theta\,sin\,\theta \\ cos\,\theta\,sin\,\theta & (sin\,\theta)^2 & -cos\,\theta\,sin\,\theta & -(sin\,\theta)^2 \\ (cos\,\theta)^2 & -cos\,\theta\,sin\,\theta & (cos\,\theta)^2 & cos\,\theta\,sin\,\theta \\ -cos\,\theta\,sin\,\theta & -(sin\,\theta)^2 & cos\,\theta\,sin\,\theta & (sin\,\theta)^2 \end{bmatrix} \quad (32)$$

2.4. Spatial Truss

In this section three-dimensional truss will be examined and while we examine it we can model it as bar element.

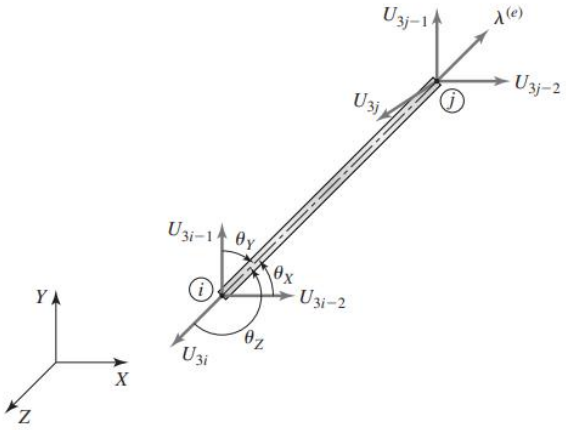

Figure 6: Bar element in a 3-D global coordinate system [3]

Similar to planar truss we can also write element displacements as:

$$u_1{}^{(e)} = U_1{}^{(e)} \cos\theta_x + U_2{}^{(e)} \sin\theta_y + U_3{}^{(e)} \cos\theta_z \tag{33}$$

$$u_2{}^{(e)} = U_4{}^{(e)} \cos\theta_x + U_5{}^{(e)} \sin\theta_y + U_6{}^{(e)} \cos\theta_z \tag{34}$$

We use displacements 1 and 4 are for global X direction, 2 and 5 are for global Y direction, 3 and 6 are for global Z direction. In matrix form:

$$\begin{bmatrix} u_1{}^{(e)} \\ u_2{}^{(e)} \end{bmatrix} = \begin{bmatrix} \cos\theta_x & \cos\theta_y & \cos\theta_z & 0 & 0 & 0 \\ 0 & 0 & 0 & \sin\theta_x & \sin\theta_y & \sin\theta_z \end{bmatrix} \begin{bmatrix} U_1{}^{(e)} \\ U_2{}^{(e)} \\ U_3{}^{(e)} \\ U_4{}^{(e)} \\ U_5{}^{(e)} \\ U_6{}^{(e)} \end{bmatrix} \tag{35}$$

According to Equation (31) we find 3-D global stiffness matrix:

$$\left[K^{(e)}\right] = \begin{bmatrix} c_x{}^2 & c_x c_y & c_x c_z & -c_x{}^2 & -c_x c_y & -c_x c_z \\ c_x c_y & c_y{}^2 & c_y c_z & -c_x c_y & -c_y{}^2 & -c_y c_z \\ c_x c_z & c_y c_z & c_z{}^2 & -c_x c_z & -c_y c_z & -c_z{}^2 \\ -c_x{}^2 & -c_x c_y & -c_x c_z & c_x{}^2 & c_x c_y & c_x c_z \\ -c_x c_y & -c_y{}^2 & -c_y c_z & c_x c_y & c_y{}^2 & c_y c_z \\ -c_x c_z & -c_y c_z & -c_z{}^2 & c_x c_z & c_y c_z & c_z{}^2 \end{bmatrix} \tag{36}$$

where $c_x = \cos\theta_x$, $c_y = \cos\theta_y$, $c_z = \cos\theta_z$.

2.5. Beam Element

Now, I will examine the beam element which is capable of exhibiting transverse bending effect. Deflected beam is shown in Figure 5:

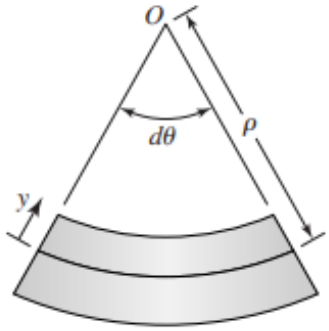

Figure 7: Deflected Beam Element [3]

Figure 5 shows us differential length dx after bending. ρ is the distance from centroid of curvature O to neutral axis. After bending dx becomes:

$$ds = (\rho - y)d\theta \tag{37}$$

Bending strain is then:

$$\varepsilon_x = \frac{ds-dx}{dx} = \frac{(\rho-y)d\theta-\rho d\theta}{\rho d\theta} = -\frac{y}{\rho} \tag{38}$$

Radius of curvature of planar curve, approximated by [3]:

$$\rho = \frac{1}{\frac{d^2v}{dx^2}} \tag{39}$$

Normal strain becomes:

$$\varepsilon_x = -y\frac{d^2v}{dx^2} \tag{40}$$

Corresponding normal stress:

$$\sigma_x = E\varepsilon_x = -Ey\frac{d^2v}{dx^2} \tag{41}$$

Figure 8: Beam element nodal displacements

We can derive a displacement function as:

$$v_0 = a_0 + a_1x + a_2x^2 + a_3x^3 \tag{42}$$

Substituting the boundary conditions:

$$v(x=0) = v_1 = a_0 \tag{43}$$

$$v(x=L) = v_2 = a_0 + a_1L + a_2L^2 + a_3L^3 \tag{44}$$

$$\left.\frac{dv}{dx}\right|_{x=0} = \theta_1 = a_1 \tag{45}$$

$$\left.\frac{dv}{dx}\right|_{x=L} = \theta_2 = a_1 + 2a_2L + 3a_3L^2 \tag{46}$$

By solving the equations (31)-(35), we get the collection of coefficients of nodal variables expression [3]:

$$v(x) = \underbrace{\left(1 - \frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)}_{N_1} v_1 + \underbrace{\left(x - \frac{2x^2}{L} - \frac{x^3}{L^2}\right)}_{N_2}\theta_1 + \underbrace{\left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)}_{N_3} v_2 + \underbrace{\left(\frac{x^3}{L^2} - \frac{x^2}{L}\right)}_{N_4}\theta_2 \tag{47}$$

Now, we may start to find beam element stiffness matrix by deriving from total strain energy [3]:

$$U_e = \frac{1}{2}\int_V \sigma_x\varepsilon_x dV \tag{48}$$

$$U_e = \frac{E}{2}\int_V \ y^2 \left(\frac{d^2v}{dx^2}\right)^2 dV \tag{49}$$

$$U_e = \frac{E}{2}\int_0^L \left(\frac{d^2v}{dx^2}\right)^2 \left(\int_A \ y^2 dA\right) dx = \frac{EI_z}{2}\int_0^L \left(\frac{d^2v}{dx^2}\right)^2 dx \tag{50}$$

$$U_e = \frac{EI_z}{2}\int_o^L \left(\frac{d^2N_1}{dx^2}v_1 + \frac{d^2N_2}{dx^2}\theta_1 + \frac{d^2N_3}{dx^2}v_2 + \frac{d^2N_4}{dx^2}\theta_2\right)^2 dx \tag{51}$$

Now use Castigliano's first theorem for finding F_1, M_1, F_2, M_2:

$$\frac{\partial U_e}{\partial v_1} = F_1 = EI_z\int_o^L (\frac{d^2N_1}{dx^2}v_1 + \frac{d^2N_2}{dx^2}\theta_1 + \frac{d^2N_3}{dx^2}v_2 + \frac{d^2N_4}{dx^2}\theta_2)\frac{d^2N_1}{dx^2}dx \tag{52}$$

$$\frac{\partial U_e}{\partial \theta_1} = M_1 = EI_z\int_o^L (\frac{d^2N_1}{dx^2}v_1 + \frac{d^2N_2}{dx^2}\theta_1 + \frac{d^2N_3}{dx^2}v_2 + \frac{d^2N_4}{dx^2}\theta_2)\frac{d^2N_2}{dx^2}dx \tag{53}$$

$$\frac{\partial U_e}{\partial v_2} = F_2 = EI_z\int_o^L (\frac{d^2N_1}{dx^2}v_1 + \frac{d^2N_2}{dx^2}\theta_1 + \frac{d^2N_3}{dx^2}v_2 + \frac{d^2N_4}{dx^2}\theta_2)\frac{d^2N_3}{dx^2}dx \tag{54}$$

$$\frac{\partial U_e}{\partial \theta_2} = M_2 = EI_z\int_o^L (\frac{d^2N_1}{dx^2}v_1 + \frac{d^2N_2}{dx^2}\theta_1 + \frac{d^2N_3}{dx^2}v_2 + \frac{d^2N_4}{dx^2}\theta_2)\frac{d^2N_4}{dx^2}dx \tag{55}$$

Then we put equations (41)-(44) into matrix form we get:

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix} \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ M_1 \\ F_2 \\ M_2 \end{bmatrix} \tag{56}$$

When we compare the Equations (41)-(44) we can get another equation [3]:

$$k_{mn} = k_{nm} = EI_z\int_0^L \frac{d^2N_m}{dx^2}\frac{d^2N_n}{dx^2}dx \quad m,n = 1,4 \tag{57}$$

After we make equation (46) as dimensionless, we can calculate k_{11} value [3]:

$$\int_0^L f(x)dx = \int_0^1 f(\xi)Ld\xi \tag{58}$$

$$\frac{d}{dx} = \frac{1}{L}\frac{d}{d\xi} \tag{59}$$

$$k_{mn} = k_{nm} = EI_z\int_0^L \frac{d^2N_m}{dx^2}\frac{d^2N_n}{dx^2}dx = \frac{EI_z}{L^3}\int_0^1 \frac{d^2N_m}{d\xi^2}\frac{d^2N_n}{d\xi^2}d\xi \quad m,n = 1,4 \tag{60}$$

Now we are ready to calculate k_{11}:

$$k_{11} = \frac{EI_z}{L^3}\int_0^1 (12\xi - 6)^2 d\xi = \frac{36EI_z}{L^3}\int_0^1 (4\xi^2 - 4\xi + 1)d\xi = \frac{12EI_z}{L^3} \tag{61}$$

In a similar manner, we can find other stiffness coefficients, too. So, our complete stiffness

matrix for beam element will be [3]:

$$[k_e] = \frac{EI_z}{L^3}\begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \tag{62}$$

2.6. Plane Frame

Until plane frame topic, I have examined spring, bar element, beam element. Now, if we think about bar element, there is only two degrees of freedom, one at each node. This gave us 2x2 matrix. When we examine beam element, we have found four degrees of freedom, two at each node. And this gave us 4x4 matrix. We examined these elements while introducing finite element method. Because Plane Frame topic is related with these two elements. Plane Frame is kind of combination of bar and beam elements. So, plane frame will have 6 degrees of freedom, 3 at each node (two displacement and a rotation) [5].

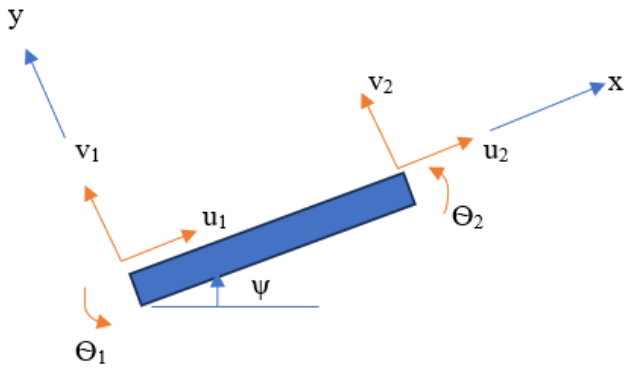

Figure 9:Nodal Displacements in Local Coordinate System

Element displacement vector in the element reference frame is:

$$[\delta] = \begin{Bmatrix} u_1 \\ v_1 \\ \theta_1 \\ u_2 \\ v_2 \\ \theta_2 \end{Bmatrix} \tag{63}$$

And the element stiffness matrix becomes:

$$[k_e] = \begin{bmatrix} \frac{AE}{L} & 0 & 0 & -\frac{AE}{L} & 0 & 0 \\ 0 & \frac{12EI_Z}{L^3} & \frac{6EI_Z}{L^2} & 0 & -\frac{12EI_Z}{L^3} & \frac{6EI_Z}{L^2} \\ 0 & \frac{6EI_Z}{L^2} & \frac{4EI_Z}{L} & 0 & -\frac{6EI_Z}{L^2} & \frac{2EI_Z}{L} \\ -\frac{AE}{L} & 0 & 0 & \frac{AE}{L} & 0 & 0 \\ 0 & -\frac{12EI_Z}{L^3} & -\frac{6EI_Z}{L^2} & 0 & \frac{12EI_Z}{L^3} & -\frac{6EI_Z}{L^2} \\ 0 & \frac{6EI_Z}{L^2} & \frac{2EI_Z}{L} & 0 & -\frac{6EI_Z}{L^2} & \frac{4EI_Z}{L} \end{bmatrix} \quad (64)$$

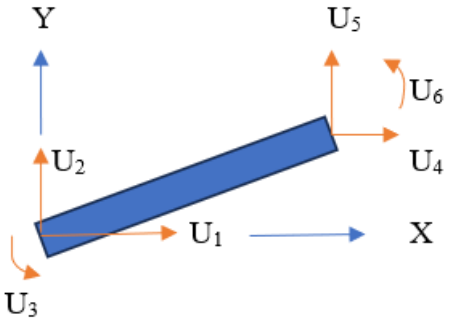

Figure 10:Nodal Displacements in Global Coordinate System

The element displacements can be written in terms of global displacements:

$$u_1 = U_1 \cos\psi + U_2 \sin\psi \quad (65)$$

$$v_1 = -U_1 \sin\psi + U_2 \cos\psi \quad (66)$$

$$\theta_1 = U_3 \quad (67)$$

$$u_2 = U_4 \cos\psi + U_5 \sin\psi \quad (68)$$

$$v_2 = -U_4 \sin\psi + U_5 \cos\psi \quad (69)$$

$$\theta_2 = U_6 \quad (70)$$

In matrix form:

$$\begin{Bmatrix} u_1 \\ v_1 \\ \theta_1 \\ u_2 \\ v_2 \\ \theta_2 \end{Bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\psi & \sin\psi & 0 \\ 0 & 0 & 0 & -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \end{Bmatrix} \quad (71)$$

6x6 element stiffness matrix in the global system is given by [3]:

$$[K_e] = [R]^T [k_e][R] \tag{72}$$

$$[K_e] = \frac{E}{L}\begin{bmatrix} AC^2 + \frac{12I}{L^2}S^2 & (A - \frac{12I}{L^2})CS & -\frac{6I}{L}S & -(AC^2 + \frac{12I}{L^2}S^2) & -(A - \frac{12I}{L^2})CS & -\frac{6I}{L}S \\ (A - \frac{12I}{L^2})CS & AS^2 + \frac{12I}{L^2}C^2 & \frac{6I}{L}C & -(A - \frac{12I}{L^2})CS & -(AS^2 + \frac{12I}{L^2}C^2) & \frac{6I}{L}C \\ -\frac{6I}{L}S & \frac{6I}{L}C & 4I & \frac{6I}{L}S & -\frac{6I}{L}C & 2I \\ -(AC^2 + \frac{12I}{L^2}S^2) & -(A - \frac{12I}{L^2})CS & \frac{6I}{L}S & AC^2 + \frac{12I}{L^2}S^2 & (A - \frac{12I}{L^2})CS & \frac{6I}{L}S \\ -(A - \frac{12I}{L^2})CS & -(AS^2 + \frac{12I}{L^2}C^2) & -\frac{6I}{L}C & (A - \frac{12I}{L^2})CS & (AS^2 + \frac{12I}{L^2}C^2) & -\frac{6I}{L}C \\ -\frac{6I}{L}S & \frac{6I}{L}C & 2I & \frac{6I}{L}S & -\frac{6I}{L}C & 4I \end{bmatrix} \tag{73}$$

2.7. Spatial Frame

Spatial frame is a kind of three-dimensional beam element. It has axial, torsional, and two-plane bending deflections. For finding the element stiffness matrix we will examine two plane bending, torsion and add axial deflections.

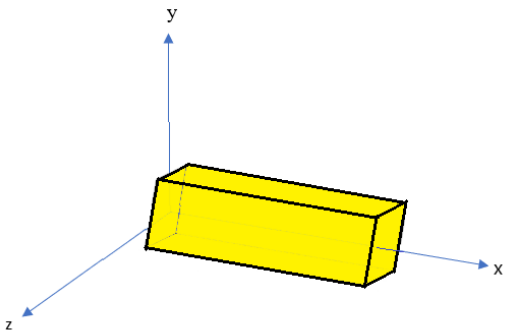

Figure 11:Three-dimensional beam element

The axes y and z chosen as principal axes. We will use these axes for area moments of inertia of cross section [3].

Element stiffness matrix of axial deflection is equal to Equation (25).

Bending about z axis:

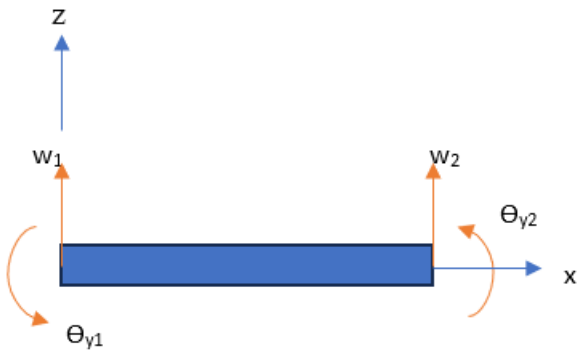

Figure 12:Nodal displacements in x-z plane

Then element stiffness matrix for x-z plane [3]:

$$[k_e] = \frac{EI_y}{L^3}\begin{bmatrix} 12 & -6L & -12 & -6L \\ -6L & 4L^2 & 6L & 2L^2 \\ -12 & 6L & 12 & 6L \\ -6L & 2L^2 & 6L & 4L^2 \end{bmatrix} \tag{74}$$

The element stiffness matrix of x-y plane will be same as Equation (51).

Now, examine the torsion on circular cylinder:

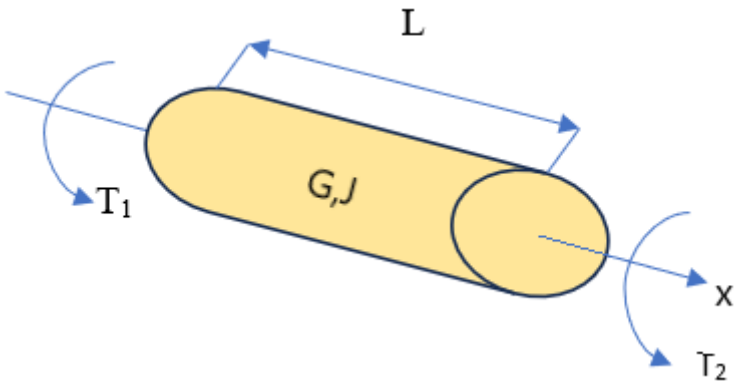

Figure 13: Circular cylinder subjected to torsion

Elastic circular cylinder under torque [4]:

$$\phi = \frac{T}{JG} \tag{75}$$

$$T = \frac{JG}{L}(\theta_{x2} - \theta_{x1}) = k_T(\theta_{x2} - \theta_{x1}) \tag{76}$$

The equilibrium equation:

$$\frac{JG}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} \emptyset_{x1} \\ \emptyset_{x2} \end{Bmatrix} = \begin{Bmatrix} M_{x1} \\ M_{x2} \end{Bmatrix} \tag{77}$$

So, the torsional stiffness matrix is:

$$[k_{torsion}] = \frac{JG}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{78}$$

As a result, when we assemble Equations (25), (51), (62), (66):

$$[k_e]\begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ \theta_{x1} \\ \theta_{y1} \\ \theta_{z1} \\ u_2 \\ v_2 \\ w_2 \\ \theta_{x2} \\ \theta_{y2} \\ \theta_{z2} \end{Bmatrix} = \begin{Bmatrix} f_{x1} \\ f_{y1} \\ f_{z1} \\ M_{x1} \\ M_{y1} \\ M_{z1} \\ f_{x2} \\ f_{y2} \\ f_{z2} \\ M_{x2} \\ M_{y2} \\ M_{z2} \end{Bmatrix} \tag{79}$$

$$[k_e] = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} & 0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\ 0 & 0 & \frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & 0 & -\frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 \\ 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} \\ -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} & 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} \\ 0 & 0 & -\frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & \frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 \\ 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 \\ 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L} \end{bmatrix} \quad (80)$$

Element stiffness matrix will be found as in Equation (72).

$$[R] = \begin{bmatrix} [r] & 0 & 0 & 0 \\ 0 & [r] & 0 & 0 \\ 0 & 0 & [r] & 0 \\ 0 & 0 & 0 & [r] \end{bmatrix} \tag{81}$$

where r is 3x3 matrix:

$$[r] = \begin{bmatrix} C_{Xx} & C_{Yx} & C_{Zx} \\ C_{Xy} & C_{Yy} & C_{Zy} \\ C_{Xz} & C_{Yz} & C_{Zz} \end{bmatrix} \tag{82}$$

$C_{Xx} = \cos\theta_{Xx}$, … etc.

3. PROGRAM DESIGN PROCEDURE

3.1 MATLAB

MATLAB is a programming platform created for help to the scientist and engineers by analyzing data's, developing algorithms, creating models and applications. The MATLAB language, a matrix-based language that enables the most natural expressions of computer mathematics.

3.2 Getting Number of Nodes and Elements

First, we need to ask to user how many nodes and elements do we have. Our program will do calculations according to this basic information.

```
prompt = {'Enter the number of nodes: '};
dlgtitle = 'Number of Nodes';
dims = [1 50];
answer = inputdlg(prompt,dlgtitle,dims);
N= str2double(answer{1});
while N<=0
   disp('Invalid Entry. Please enter a positive integer.');
   answer = inputdlg(prompt,dlgtitle,dims);
   N= str2double(answer{1});
end
```

In this part, first 4 lines belong to design of the program. Prepares pop-up window. Creates dialog box to gather user input. Stores input as character. And in the 5th line we convert it to numerical answer. Then attain it to N which is created to store number of nodes.

Line 6 and rest of it prevents user to enter negative and zero node.

Now a similar procedure will be done for getting number of elements from user.

```
prompt={'Enter the number of elements: '};
dlgtitle='Number of Elements';
dims=[1 50];
```

```
answer=inputdlg(prompt,dlgtitle,dims);
Element=str2double(answer{1});
while Element<=0
   disp('Invalid Entry. Please enter a positive integer.');
   answer=inputdlg(prompt,dlgtitle,dims);
   Element=str2double(answer{1});
end
```

3.3 Getting Coordinates of Nodes and Plot the Frame

For the calculations will be done later and visualize our frame we need the coordinates of nodes.

```
for i=1:N
   prompt={sprintf('Enter the x coordinate element %d: ',i),sprintf('Enter the y coordinate of element %d: ',i)};
   dlgtitle='Coordinates';
   dims=[1 50];
   answer=inputdlg(prompt,dlgtitle,dims);
   x(i)=str2num(answer{1});
   y(i)=str2num(answer{2});
end
plot(x,y,'-o',LineWidth=2)
```

Programs asks coordinates of nodes and then stores them in x and y matrices. After that visualizes the frame to user.

3.4 Getting Element Properties and Calculating Angle and Length

For calculations that will be done later, we need properties of every element. And then we also can calculate length and angle by using inputs requested until now.

```
for i=1:Element
   prompt={sprintf('Enter modulus of elasticity for element %d (E) [kPa]: ',i),sprintf('Enter cross sectional area for element %d (A) [m^2]:',i),sprintf('Enter moment of inertia for element %d (I) [m^4]: ',i)};
   dlgtitle='Element Properties';
   dims=[1 50];
   answer=inputdlg(prompt,dlgtitle,dims);
   E(i)=str2num(answer{1});
   while E(i)<=0
      disp('Invalid Entry. Please enter a positive integer.');
      answer=inputdlg(prompt,dlgtitle,dims);
      E(i)=str2num(answer{1});
   end
   A(i)=str2num(answer{2});
   while A(i)<=0
      disp('Invalid Entry. Please enter a positive integer.');
      answer=inputdlg(prompt,dlgtitle,dims);
      A(i)=str2num(answer{2});
   end
   I(i)=(str2num(answer{3}));
   while I(i)<=0
      disp('Invalid Entry. Please enter a positive integer.');
      answer=inputdlg(prompt,dlgtitle,dims);
      I(i)=(str2num(answer{3}));
   end

   %Element length calculated by element length function:
```

```
    Lengths(i)=ElementLength(x(i),y(i),x(i+1),y(i+1)); %m


    %Angle of elements are calculated:
    Theta(i)=atand((y(i+1)-y(i))/(x(i+1)-x(i))); %degree
end
```

In these codes, program asks user 3 different element properties which are modulus of elasticity, cross sectional area, and moment of inertia for each element. Units of these element properties already determined in questions. All questions asked in one box for each element. Also, negative and zero values forbidden to enter. After that, lengths and angles are determined and stored in Lengths, Theta matrices. Angles are determined in degree. Element lengths are calculated by ElementLength function which is:

```
function length=ElementLength(x1,y1,x2,y2)
%This function calculates the length of planar frame element that first
%node is at (x1,y1) and second node is at (x2,y2)
length=((x2-x1)^2+(y2-y1)^2)^(1/2);
end
```

3.5 Element Stiffness Matrix Function

After getting element properties and calculating lengths, element stiffness matrix can be found. I wrote element stiffness matrix function for this purpose. Codes:

```
function ElementStiffnessMatrix=ElementStiffness(E,A,I,L,theta)
% This function calculates element stiffness matrix
C=cosd(theta);
S=sind(theta);
a1=A*C^2+((12*I)/L^2)*S^2;
a2=(A-((12*I)/L^2))*C*S;
a3=-(6*I/L)*S;
a4=A*S^2+((12*I)/L^2)*C^2;
```

```
a5=(6*I/L)*C;
a6=4*I;
ElementStiffnessMatrix=E/L*[a1,a2,a3,-a1,-a2,a3;
   a2,a4,a5,-a2,-a4,a5;
   a3,a5,a6,-a3,-a5,a6/2;
   -a1,-a2,-a3,a1,a2,-a3;
   -a2,-a4,-a5,a2,a4,-a5;
   a3,a5,a6/2,-a3,-a5,a6];
end
```

Then I created 3d matrix for store each element stiffness matrix and placed them:

```
for i=1:Element
   kArrays3D(:,:,i)=ElementStiffness(E(i),A(i),I(i),Lengths(i),Theta(i));
end
```

An example for an element that nodes (1,1) and (2,2):

```
kArrays3D(:,:,1) =

    2.4749   -1.7678   -2.1213   -2.4749    1.7678   -2.1213
   -1.7678    2.4749    2.1213    1.7678   -2.4749    2.1213
   -2.1213    2.1213    2.8284    2.1213   -2.1213    1.4142
   -2.4749    1.7678    2.1213    2.4749   -1.7678    2.1213
    1.7678   -2.4749   -2.1213   -1.7678    2.4749   -2.1213
   -2.1213    2.1213    1.4142    2.1213   -2.1213    2.8284
```

Figure 14:Example of element stiffness matrix

3.6 Assembly Function for Global Stiffness Matrix

By using direct stiffness method, I wrote a function in order to assemble each element stiffness matrix to global stiffness matrix. Codes:

```
function AssembleK=AssemblyFunction(K,k,i,j)


 K(3*i-2,3*i-2) = K(3*i-2,3*i-2) + k(1,1);
```

```
K(3*i-2,3*i-1) = K(3*i-2,3*i-1) + k(1,2);
K(3*i-2,3*i) = K(3*i-2,3*i) + k(1,3);
K(3*i-2,3*j-2) = K(3*i-2,3*j-2) + k(1,4);
K(3*i-2,3*j-1) = K(3*i-2,3*j-1) + k(1,5);
K(3*i-2,3*j) = K(3*i-2,3*j) + k(1,6);
K(3*i-1,3*i-2) = K(3*i-1,3*i-2) + k(2,1);
K(3*i-1,3*i-1) = K(3*i-1,3*i-1) + k(2,2);
K(3*i-1,3*i) = K(3*i-1,3*i) + k(2,3);
K(3*i-1,3*j-2) = K(3*i-1,3*j-2) + k(2,4);
K(3*i-1,3*j-1) = K(3*i-1,3*j-1) + k(2,5);
K(3*i-1,3*j) = K(3*i-1,3*j) + k(2,6);
K(3*i,3*i-2) = K(3*i,3*i-2) + k(3,1);
K(3*i,3*i-1) = K(3*i,3*i-1) + k(3,2);
K(3*i,3*i) = K(3*i,3*i) + k(3,3);
K(3*i,3*j-2) = K(3*i,3*j-2) + k(3,4);
K(3*i,3*j-1) = K(3*i,3*j-1) + k(3,5);
K(3*i,3*j) = K(3*i,3*j) + k(3,6);
K(3*j-2,3*i-2) = K(3*j-2,3*i-2) + k(4,1);
K(3*j-2,3*i-1) = K(3*j-2,3*i-1) + k(4,2);
K(3*j-2,3*i) = K(3*j-2,3*i) + k(4,3);
K(3*j-2,3*j-2) = K(3*j-2,3*j-2) + k(4,4);
K(3*j-2,3*j-1) = K(3*j-2,3*j-1) + k(4,5);
K(3*j-2,3*j) = K(3*j-2,3*j) + k(4,6);
K(3*j-1,3*i-2) = K(3*j-1,3*i-2) + k(5,1);
K(3*j-1,3*i-1) = K(3*j-1,3*i-1) + k(5,2);
K(3*j-1,3*i) = K(3*j-1,3*i) + k(5,3);
```

```
K(3*j-1,3*j-2) = K(3*j-1,3*j-2) + k(5,4);
K(3*j-1,3*j-1) = K(3*j-1,3*j-1) + k(5,5);
K(3*j-1,3*j) = K(3*j-1,3*j) + k(5,6);
K(3*j,3*i-2) = K(3*j,3*i-2) + k(6,1);
K(3*j,3*i-1) = K(3*j,3*i-1) + k(6,2);
K(3*j,3*i) = K(3*j,3*i) + k(6,3);
K(3*j,3*j-2) = K(3*j,3*j-2) + k(6,4);
K(3*j,3*j-1) = K(3*j,3*j-1) + k(6,5);
K(3*j,3*j) = K(3*j,3*j) + k(6,6);

AssembleK=K;
end
```

Now by using this function and the elements stiffness matrices which was stored in section 3.5, we assemble all element stiffness matrices into a global stiffness matrix. The codes for assembly:

```
GlobalMatrixNumber=3*N;
K=zeros(GlobalMatrixNumber);
WillIncrease=1;

while WillIncrease<=Element
  K=AssemblyFunction(K,kArrays3D(:,:,WillIncrease),WillIncrease,WillIncrease+1);
  WillIncrease=WillIncrease+1;
end
```

3.7 Boundary Conditions for Nodes

First, I created force and displacement matrices from zeros. We will store forces, moments, displacements and rotations into those matrices:

```
Forces1=zeros(GlobalMatrixNumber,1);
Displacemenst1=zeros(GlobalMatrixNumber,1);
```

Now it is time to ask inputs from user for each node.

```
for i=1:N
   prompt=sprintf('Is there any inclined support for node %d',i);
   AnyInclined=questdlg(prompt,'Inclined Support','Yes','No','No');

   if strcmpi(AnyInclined, 'Yes')
      T=eye(GlobalMatrixNumber,GlobalMatrixNumber);
      InclinedAngle=input('What is degree with horizontal:');
      T=PlaneFrameInclinedSupport(T,i,InclinedAngle);
      K=T*K*T';
   end

   prompt=sprintf('Which boundary condition you will enter for node %d: ',i);
BoundaryConditionSelection=questdlg(prompt,'Boundary Condition Selection','Force/Moment','Displacement/Rotation','Displacement/Rotation');
   if strcmp(BoundaryConditionSelection,'Force/Moment')
      BoundaryConditionOPT(i)=1;
   else
      BoundaryConditionOPT(i)=2;
   end

   if BoundaryConditionOPT(1,i)==1
```

```
    prompt={sprintf('Enter your force value on x axis for node %d [kN]: ',i),sprintf('Enter your force value on y axis for node %d [kN]: ',i),sprintf('Enter your moment value for node %d [kN]: ',i)};
    dlgtitle='Force/Moment Boundary Condition';
    dims = [1 50];
    answer = inputdlg(prompt,dlgtitle,dims);
    Forces1(3*i-2)=str2double(answer{1});
    Forces1(3*i-1)=str2double(answer{2});
    Forces1(3*i)=str2double(answer{3});
  elseif BoundaryConditionOPT(1,i)==2
    prompt={sprintf('Enter your displacement value on x axis for node %d [m]: ',i),sprintf('Enter your displacement value on y axis for node %d [m]: ',i),sprintf('Enter your rotation value for node %d [rad]: ',i)};
    dlgtitle='Displacement/Rotation Boundary Condition';
    dims = [1 50];
    answer = inputdlg(prompt,dlgtitle,dims);
    Displacemenst1(3*i-2)=str2double(answer{1});
    Displacemenst1(3*i-1)=str2double(answer{2});
    Displacemenst1(3*i)=str2double(answer{3});
  end

end
```

In second and third lines, program asks if there is any inclined support for aforementioned node. Next, if the answer is yes then we transform that support and do calculations according to that. We use PlaneFrameInclinedSupport function for this. This function calculates the transformation matrix of the inclined support at node “i” with angle of inclination alpha (in degrees).

In the lines between 11-16, we ask to user which boundary condition he or she want to enter. And the program stores boundary condition option in BoundaryConditionOPT matrix. In that matrix 1 means force/moment and 2 means displacement rotation. According to this selection, user enter his or her inputs and the program stores them in Forces1 and Displacements1 matrices which are created as zeros before.

The PlaneFrameInclinedSupport function is:

```
function y = PlaneFrameInclinedSupport(T,i,alpha)


 % Transformation matrix of inclined support


 T(3*i-2,3*i-2) = cosd(alpha);

 T(3*i-2,3*i-1) = sind(alpha);

 T(3*i-2,3*i) = 0;

 T(3*i-1,3*i-2) =-sind(alpha);

 T(3*i-1,3*i-1) = cosd(alpha);

 T(3*i-1,3*i) = 0;

 T(3*i,3*i-2) = 0;

 T(3*i,3*i-1) = 0;

 T(3*i,3*i) = 1;

 y=T;
```

3.8 Storing Boundary Conditions, Global Stiffness Matrix

First, I store the forces, displacements, and global stiffness matrix:

```
K_used1=K;

ForcesOrg=Forces1;

DispOrg=Displacemenst1;
```

Then, I wrote the codes for determining boundary condition decisions of user

```
for i=1:length(BoundaryConditionOPT)
   Known_Condition(3*i-2:3*i)=BoundaryConditionOPT(i);
end
```

All choices are stored in Known_Condition 3 by 3. Because when we ask the boundary condition choice to user, we get 3 boundary condition for a node. With the written code above, we can be aware of which boundary conditions are known in matrix, if it is force/moment or displacement/rotation.

3.9 Solving Unknown Displacements and Forces

```
for i=1:length(Known_Condition)
   if Known_Condition(i)==2
      K_used1(i,:)=0;
      K_used1(i,i)=1;
      Forces1(i)=Displacemenst1(i);
   end
end


Unknown_Displ=K_used1\Forces1;
```

In this part, the program finds displacement boundary conditions that are entered by user. Our aim is solving for unknown displacements by using known force boundary conditions. So, first we make 0 the line that we know displacement boundary condition in global stiffness matrix which we created. Then add 1 at i^{th} row i^{th} column and add the known displacements into force matrix. This last process for not changing the result. Then I solved for the unknown displacements by taking the inverse of the modified global stiffness matrix and multiplying it by the force vector.

For clarity, I will solve an example for this process. Think we have a system of matrix equation with two unknowns as:

$$\begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ U_2 \end{bmatrix} = \begin{bmatrix} R_1 \\ 10 \end{bmatrix} \tag{83}$$

In this case we know U_1 and R_2. So, we update U_1 row in global stiffness matrix as zero.

$$\begin{bmatrix}0 & 0\\3 & 5\end{bmatrix}\begin{bmatrix}1\\U_2\end{bmatrix}=\begin{bmatrix}R_1\\13\end{bmatrix} \tag{84}$$

Then we update the diagonals of rows of U_1 as 1:

$$\begin{bmatrix}1 & 0\\3 & 5\end{bmatrix}\begin{bmatrix}1\\U_2\end{bmatrix}=\begin{bmatrix}R_1\\13\end{bmatrix} \tag{85}$$

Which becomes

$$\begin{bmatrix}1 & 0\\3 & 5\end{bmatrix}\begin{bmatrix}1\\U_2\end{bmatrix}=\begin{bmatrix}1\\13\end{bmatrix} \tag{86}$$

Then we can find U_2:

$$\begin{bmatrix}1\\U_2\end{bmatrix}=\begin{bmatrix}1 & 0\\3 & 5\end{bmatrix}^{-1}\begin{bmatrix}1\\13\end{bmatrix} \tag{87}$$

$$\begin{bmatrix}1\\U_2\end{bmatrix}=\begin{bmatrix}1\\2\end{bmatrix} \tag{88}$$

As seen in Equation (88), we found U_2 and also result does not affected.

Now I wrote a code for store the displacement boundary conditions. After storing process, we find all force boundary conditions and show the result to user.

```
for i=1:length(Known_Condition)
   if Known_Condition(i)==1
      DispOrg(i)=Unknown_Displ(i);
   end
end
Reactions=K*DispOrg;
%DispOrg
%Reactions
Results={'Node Forces and Moments:',num2str(Reactions),'Node Displacements and Rotations',num2str(DispOrg)};
msgbox(Results,'Results')
```

3.10 Obtaining Element Forces with respect to Displacements

```
for i=1:Element
   ElementDisplacements(:,:,i)=DispOrg(3*i-2:3*i+3,1);
end
```

In that code we get the results from founded displacement boundary conditions and store them in a new 3D array which name is ElementDisplacements. These results are special for each element by separately. Because we will plot diagrams of each element.

```
function y = PlaneFrameElementForces(E,A,I,L,theta,u)
 x = theta;
 C = cosd(x);
 S = sind(x);
 b1 = E*A/L;
 b2 = 12*E*I/(L*L*L);
 b3 = 6*E*I/(L*L);
 b4 = 4*E*I/L;
 b5 = 2*E*I/L;
 kprime = [b1 0 0 -b1 0 0;0 b2 b3 0 -b2 b3;
 0 b3 b4 0 -b3 b5;-b1 0 0 b1 0 0;
 0 -b2 -b3 0 b2 -b3;0 b3 b5 0 -b3 b4];
 T= [C S 0 0 0 0 ; -S C 0 0 0 0; 0 0 1 0 0 0; 0 0 0 C S 0 ; 0 0 0 -S C 0; 0 0 0 0 0 1];
 y = kprime*T*u;
end
```

The function above calculates the element force vector using the modulus of elasticity E, cross-sectional area A, moment of inertia I, length L, and the element displacement vector u. It returns the 6×1 element force vector f. I will use these results when the program plot plane frame element force diagram.

```
for i=1:Element


ElementForces(:,:,i)=PlaneFrameElementForces(E(i),A(i),I(i),Lengths(i),Theta(i),ElementDis
placements(:,:,i));

end
```

In the code above we call PlaneFrameElementForces function and calculate element forces. Then store them in 3D array.

3.11 Functions for Plot Diagrams

```
function y = PlaneFrameElementAxialDiagram(f,L,i)
 figure
 x= [0 ; L];
 z = [-f(1) ; f(4)];
 hold on;
 title(['Axial Force Diagram of Element ',num2str(i)]);
 plot(x,z);
 y1= [0 ; 0];
 plot(x,y1,'k')
end
```

I wrote this function to plot the axial force diagram for the plane frame element with nodal force vector and length.

In second line there writes “figure”. This provides us new diagram windows for each element.

In the 5th line, name changes according to element. The line with y1 determines our zero axis.

```
function y = PlaneFrameElementShearDiagram(f,L,i)
figure
x= [0 ; L];
z = [f(2) ;-f(5)];
hold on;
```

```
title(['Shear Force Diagram of Element ',num2str(i)]);
plot(x,z);
y1= [0 ; 0];
plot(x,y1,'k')
end
```

This function plots the shear force diagram for the plane frame element with nodal force vector f and length L.

```
function y = PlaneFrameElementMomentDiagram(f,L,i)
figure
x= [0 ; L];
z = [-f(3) ; f(6)];
hold on;
title(['Bending Moment Diagram of Element '],num2str(i));
plot(x,z);
y1=[0;0];
plot(x,y1,'k')
end
```

This function plots the bending moment diagram for the plane frame element with nodal force vector f and length L.

3.12 Getting Diagrams

```
for i=1:Element
   PlaneFrameElementAxialDiagram(ElementForces(:,:,i),Lengths(i),i)
   PlaneFrameElementShearDiagram(ElementForces(:,:,i),Lengths(i),i)
   PlaneFrameElementMomentDiagram(ElementForces(:,:,i),Lengths(i),i)
end
```

Finally, we get the diagrams by calling the functions in section 4.11.

4. SOLVING AN EXAMPLE

In this section, I will examine a problem from the book which is MATLAB Guide to Finite Elements an Iterative Approach by Peter Kattan. [5]

Consider the plane frame shown in Figure 15. Given $E = 210\ GPa, A = 2 * 10^{-2} m^2$ and $I = 5 * 10^{-5}\ m^4$.

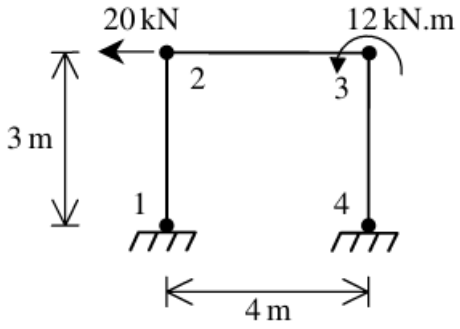

Figure 15: Plane Frame with Three Elements for Example 8.1 [5]

For this frame, I have 4 nodes and 3 elements. When we start from node 1, my coordinates will be:
1= (0,0)
2= (0,3)
3= (4,3)
4= (4,0)
First, I will show answers of the book for this plane frame. The answer says that global stiffness matrix is:

$$10^3 \begin{bmatrix} 4.7 & 0.0 & -7.0 & -4.7 & -0.0 & -7.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0 & 1400.0 & 0.0 & -0.0 & -1400.0 & 0.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7.0 & 0.0 & 14.0 & 7.0 & -0.0 & 7.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4.7 & -0.0 & 7.0 & 1054.7 & 0.0 & 7.0 & -1050.0 & 0 & 0 & 0 & 0 & 0 \\ -0.0 & -1400.0 & -0.0 & 0.0 & 1402.0 & 3.9 & 0 & -2.0 & 3.9 & 0 & 0 & 0 \\ -7.0 & 0.0 & 7.0 & 7.0 & 3.9 & 24.5 & 0 & -3.9 & 5.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1050.0 & 0 & 0 & 1054.7 & 0.0 & 7.0 & -4.7 & -0.0 & 7.0 \\ 0 & 0 & 0 & 0 & -2.0 & -3.9 & 0.0 & 1402.0 & -3.9 & -0.0 & -1400.0 & -0.0 \\ 0 & 0 & 0 & 0 & 3.9 & 5.3 & 7.0 & -3.9 & 24.5 & -7.0 & 0.0 & 7.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -4.7 & -0.0 & -7.0 & 4.7 & 0.0 & -7.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.0 & -1400.0 & 0.0 & 0.0 & 1400.0 & 0.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7.0 & -0.0 & 7.0 & -7.0 & 0.0 & 14.0 \end{bmatrix}$$

Figure 16: Global stiffness matrix according to answer of book [5]

Answer of global nodal displacement vector U (in meter):

```
U =

         0
         0
         0
   -0.0038
   -0.0000
    0.0008
   -0.0038
    0.0000
    0.0014
         0
         0
         0
```

Figure 17: Global nodal displacement vector U according to answer of book [5]

Answer of global nodal force vector F (in kN):

```
F =

   12.1897
    8.5865
  -21.0253
  -20.0000
   -0.0000
         0
   -0.0000
         0
   12.0000
    7.8103
   -8.5865
  -16.6286
```

Figure 18:Global nodal force vector F according to answer of book [5]

Of course there is answer for diagrams. But I will give them with the corresponding result of my program in order to compare them.
Now, I start my program. It asks me how many nodes do I have and I enter my node number:

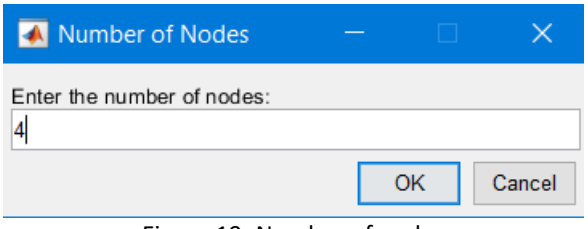

Figure 19: Number of nodes

Now asks me how many elements do I have and I enter 3:

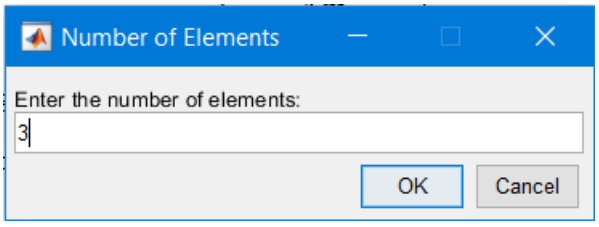

Figure 20:Number of Elements

Next, it asks the x and y coordinates of my nodes. This question same for all nodes so I put photo of only first node:

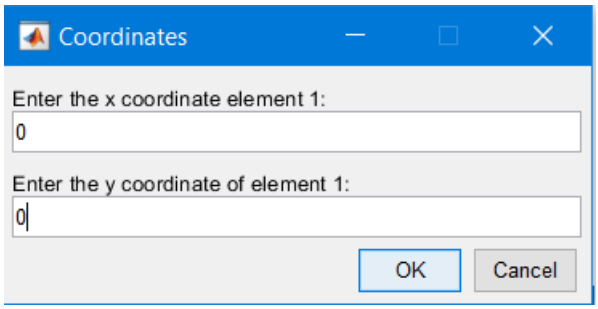

Figure 21: Asks Coordinates

After enter coordinates of nodes, the program gives us illustration of our frame so we can see our frame:

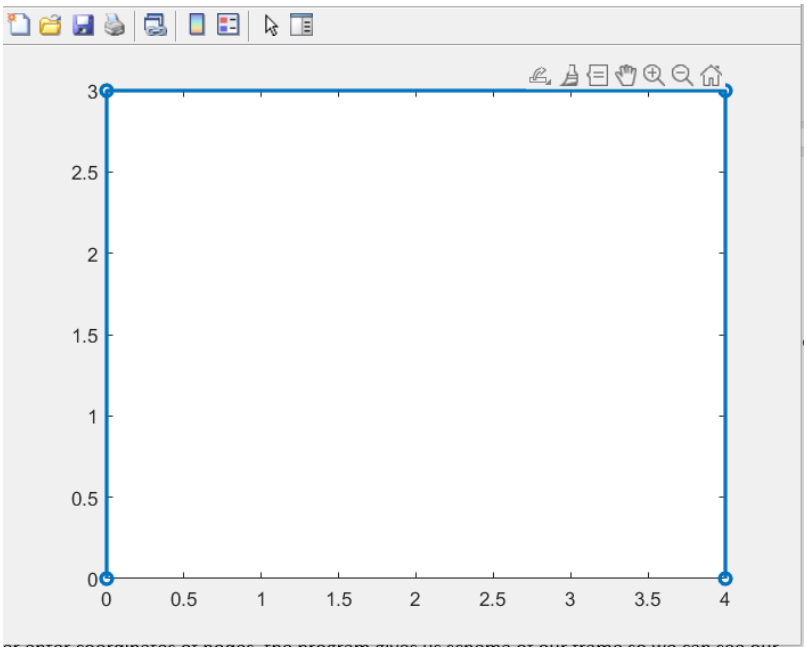

Figure 22: Drawing of Frame

Then asks for element properties:

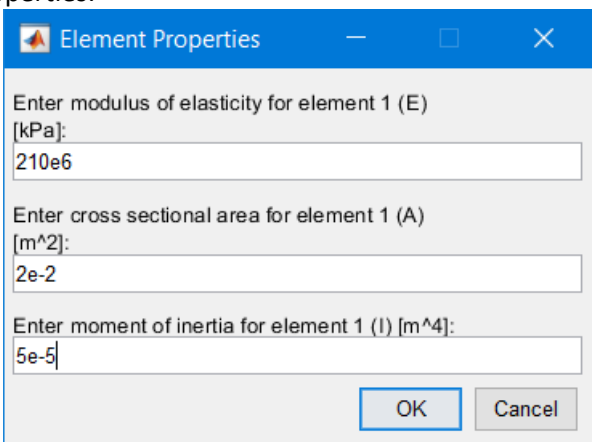

Figure 23:Element Properties

After I enter these values for element 1, I enter to “OK”. And it asks for other elements, too.

Then the programs asks if there is any inclined support for aforementioned node:

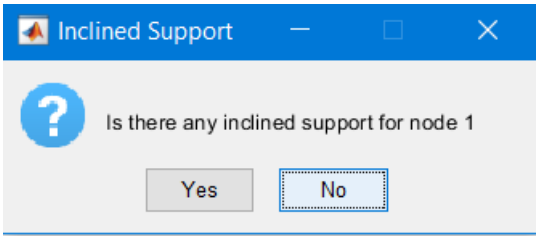

Figure 24: Asks Inclined Support

This time it does not asks this question one after another. It continues the ask questions for node 1. As next question it asks which kind of boundary condition do you want to enter:

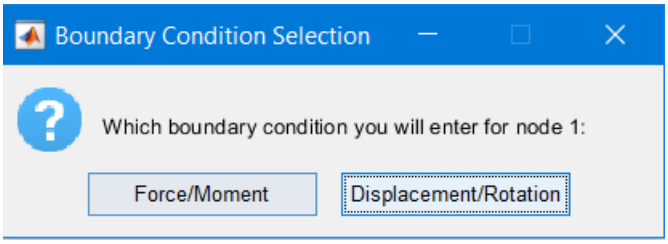

Figure 25: Boundary Condition Selection

For the first node I do not have inclined support and I want to enter Displacement/Rotation boundary conditions. Boundary conditions will be 0 because of it is fixed support. Then a tab opens to enter the values:

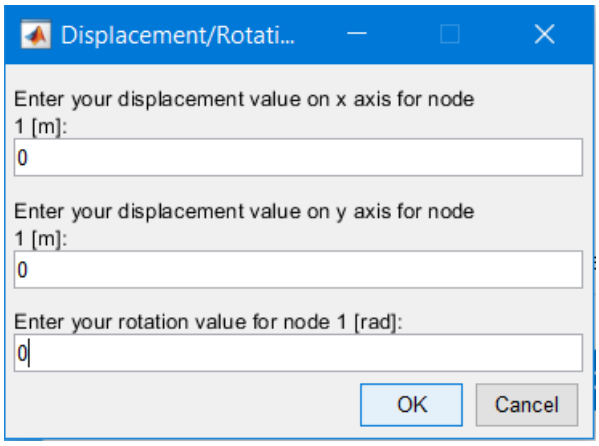

Figure 26: Displacement/Rotation

After I answer three of these questions, it asks them for other nodes, too. For the second node I enter the values which are given in question.

Then it gives us the results:

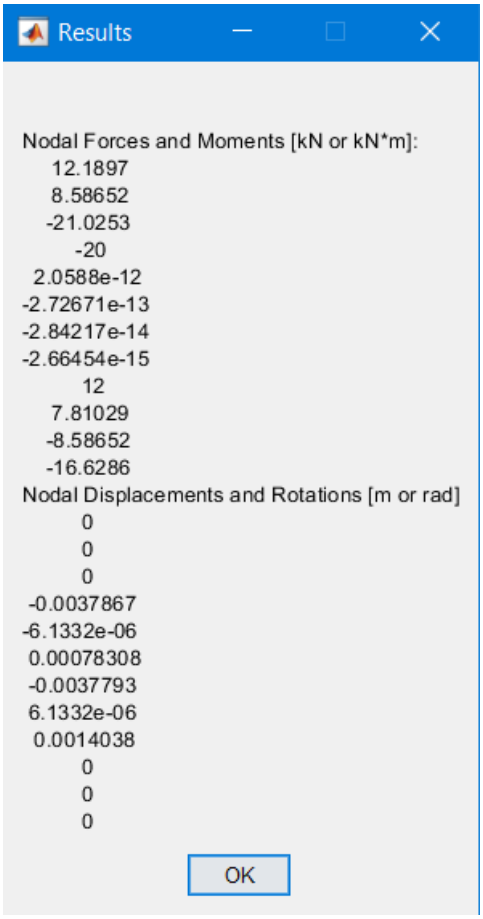

Figure 27: Results

Now, for comparison, I will show the diagrams by putting the diagram of my own computer program first and the result in the second place.

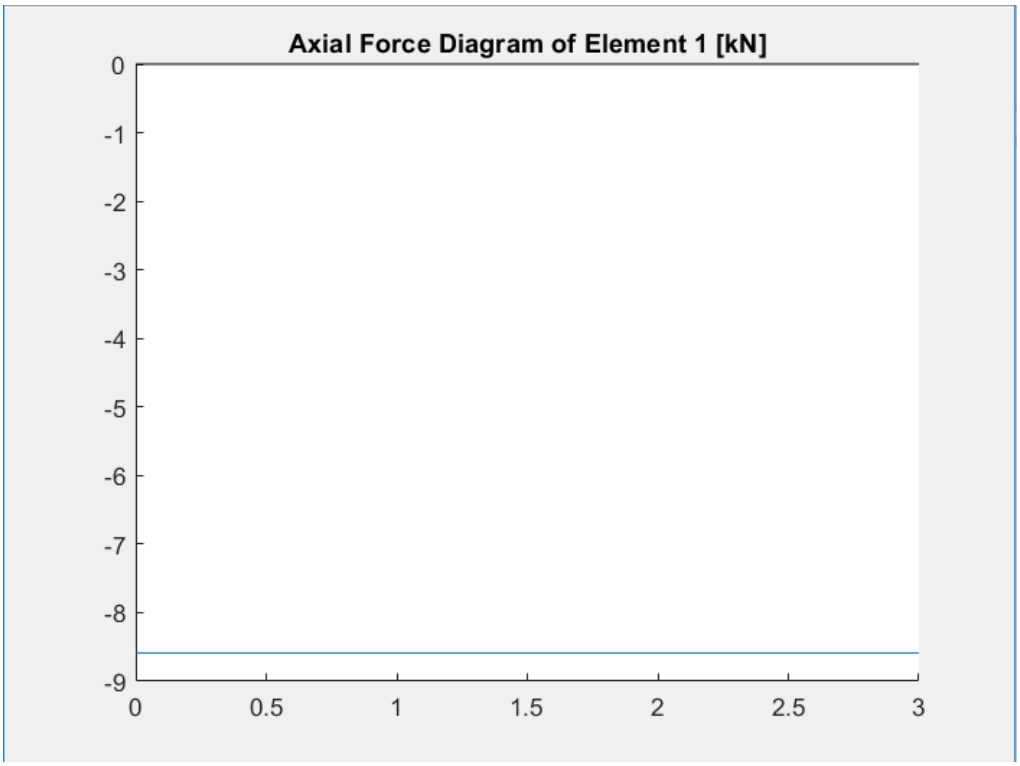

Figure 28: Axial force diagram of element 1

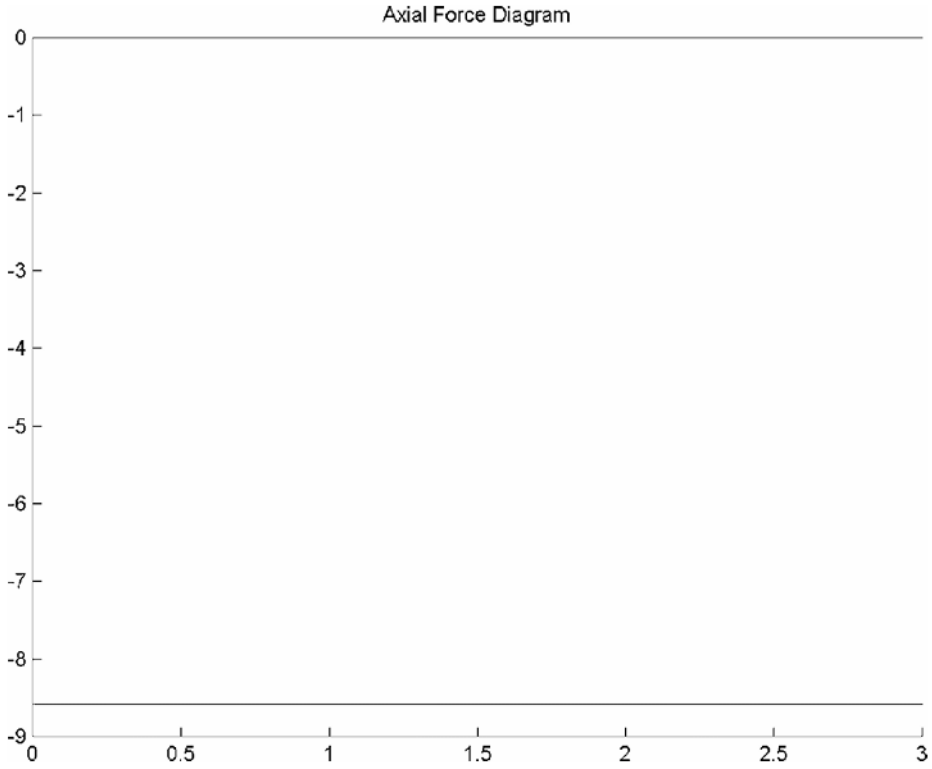

Figure 29: Axial force diagram of element 1 [5]

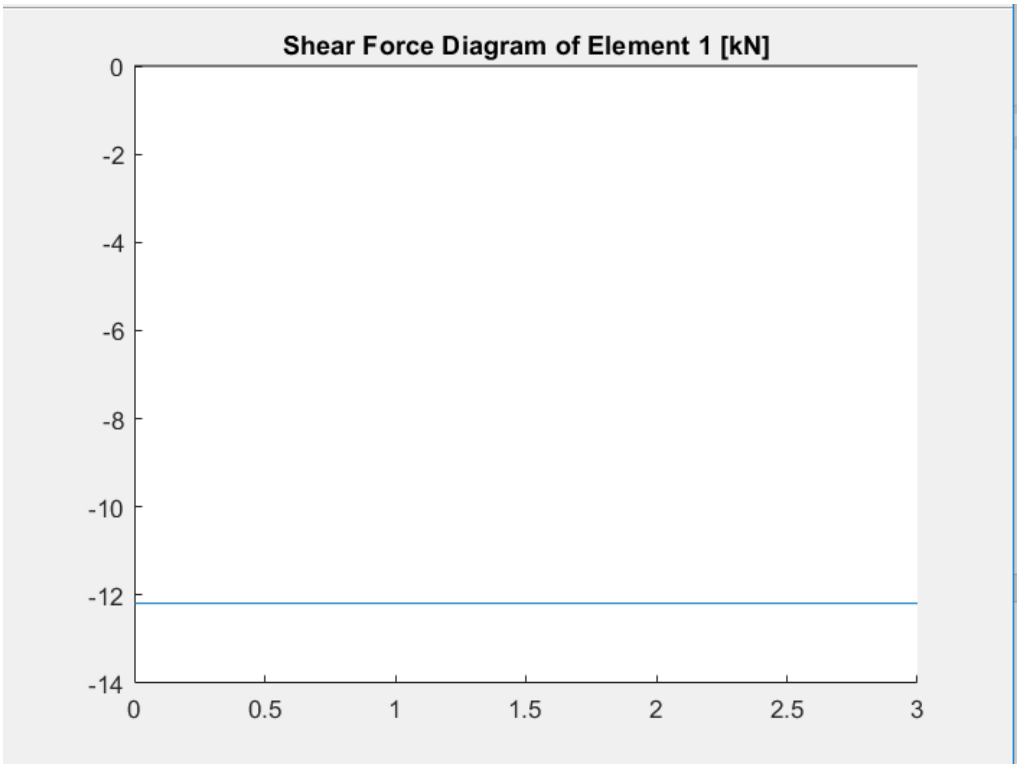

Figure 30: Shear force diagram of element 1

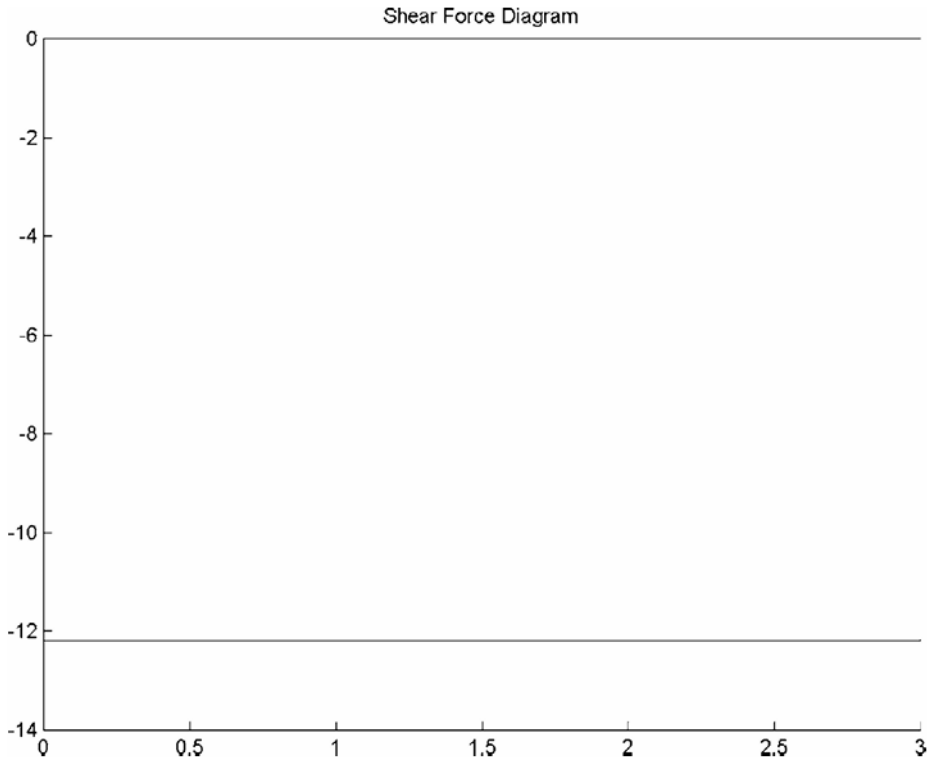

Figure 31: Shear force diagram of element 1 [5]

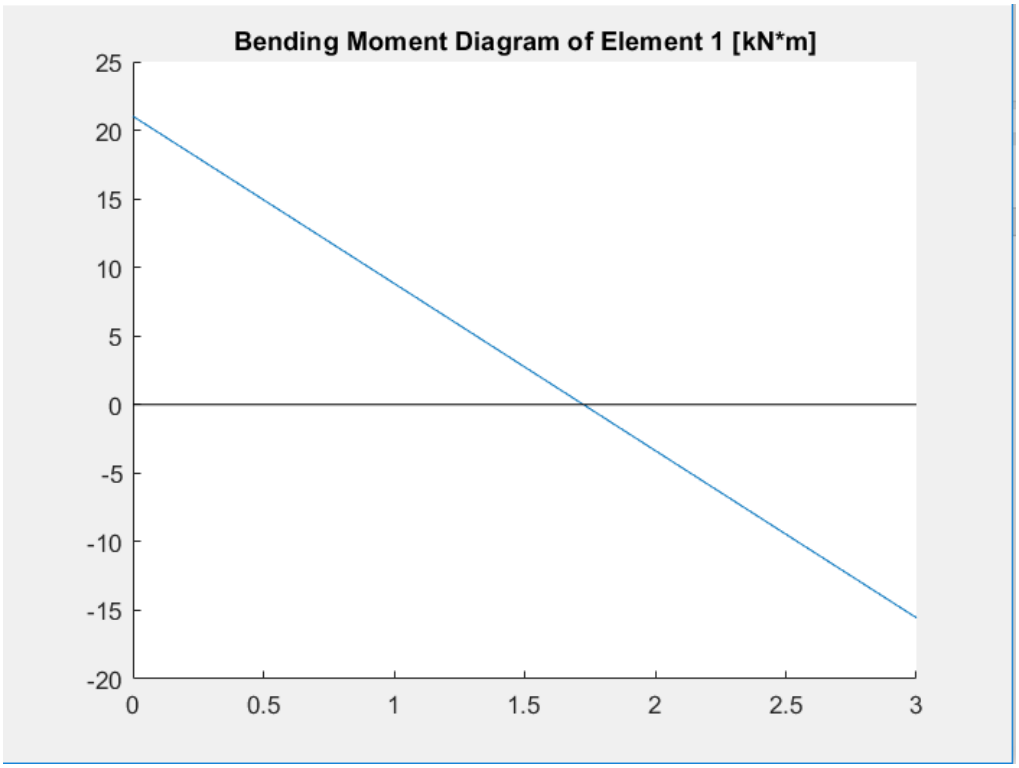

Figure 32: Bending moment diagram of element 1

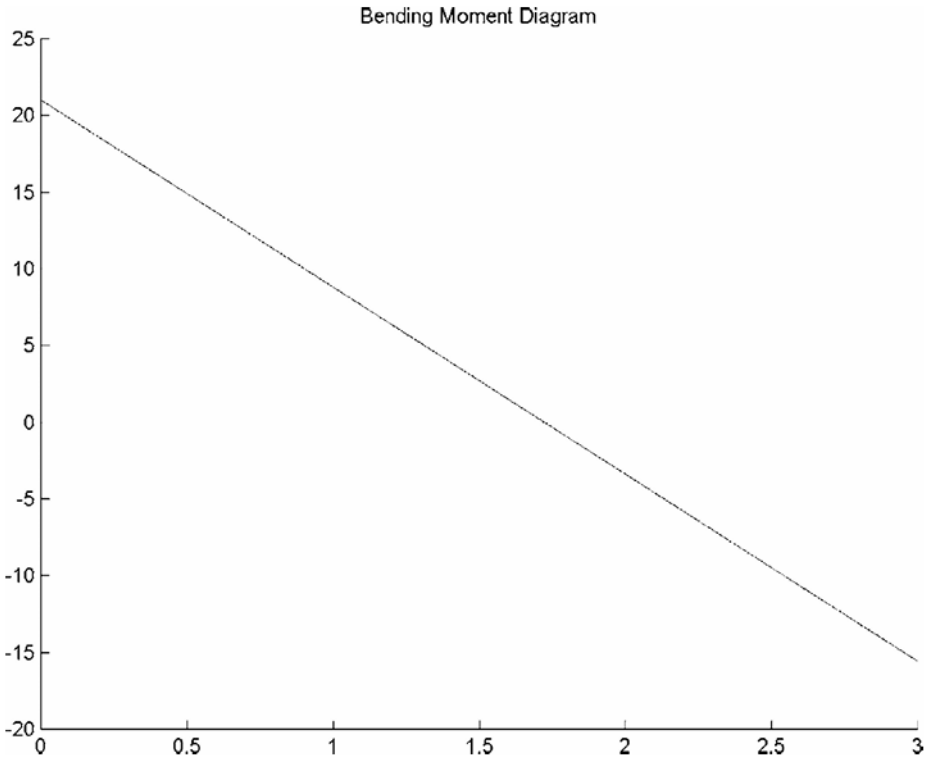

Figure 33: Bending moment diagram of element 1 [5]

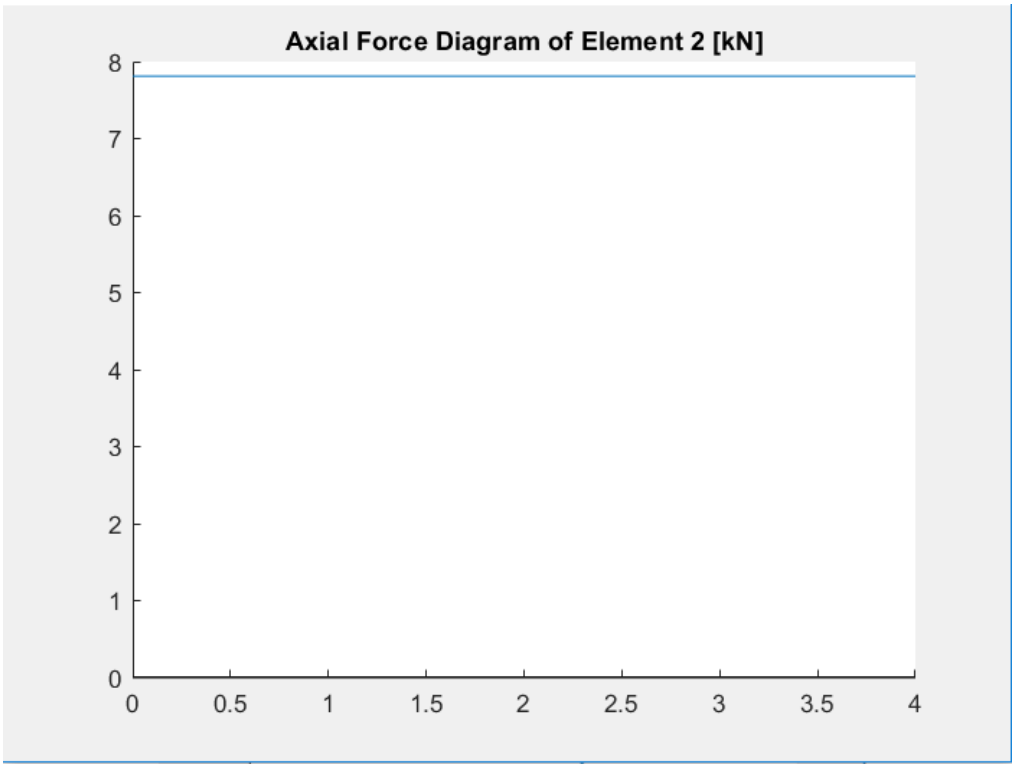

Figure 34: Axial force diagram of element 2

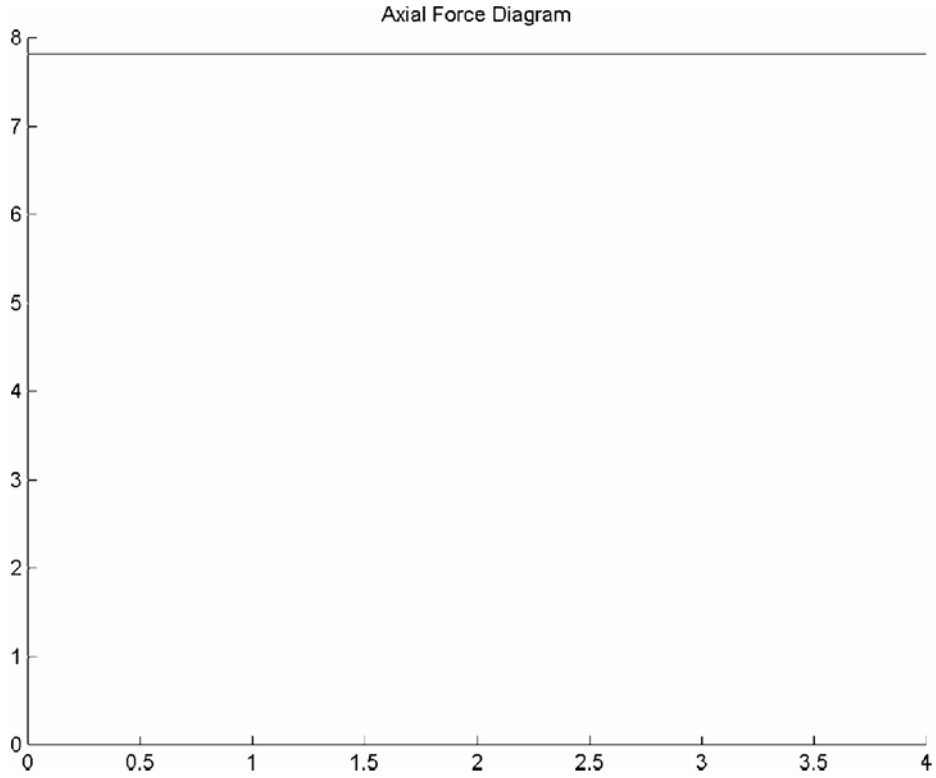

Figure 35: Axial force diagram of element 2 [5]

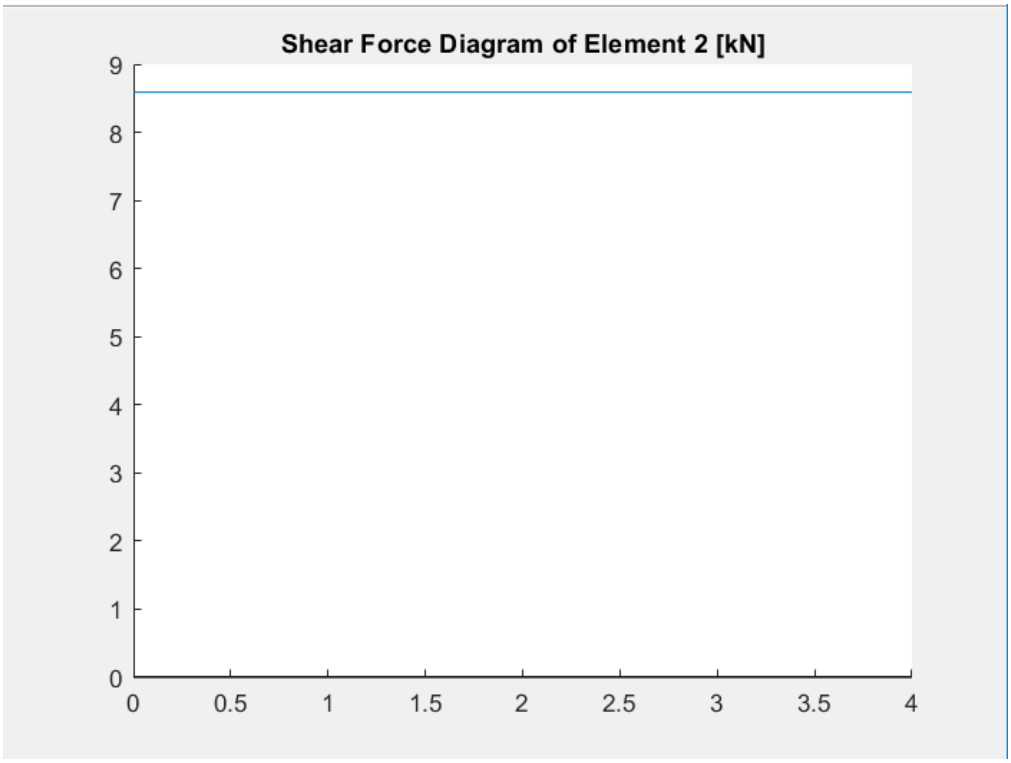

Figure 36: Shear force diagram of element 2

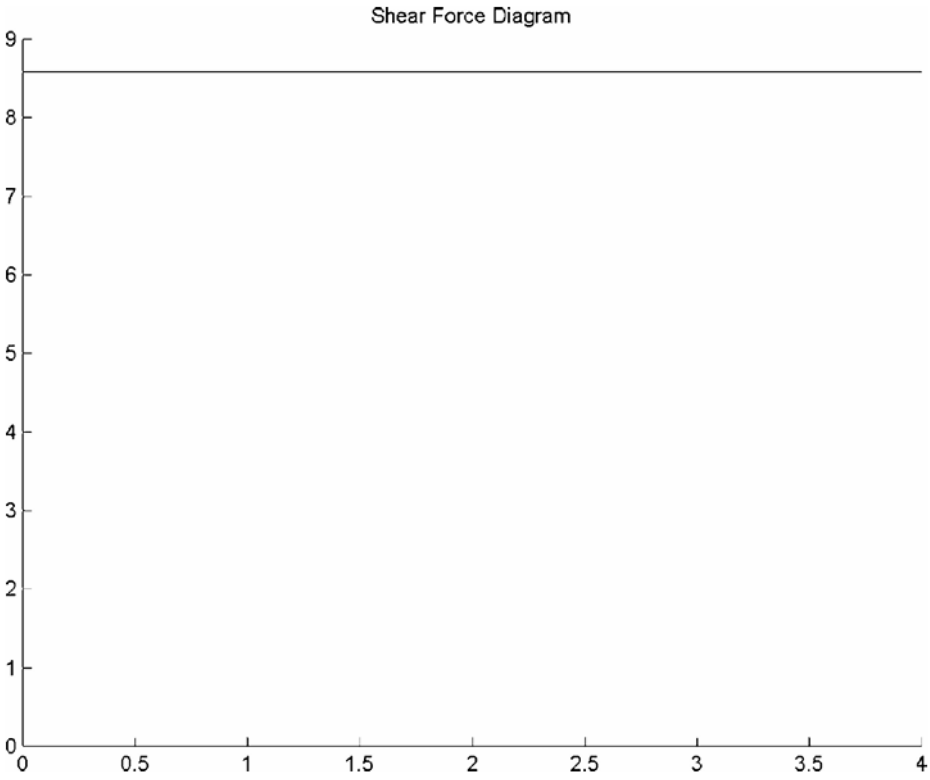

Figure 37: Shear force diagram of element 2 [5]

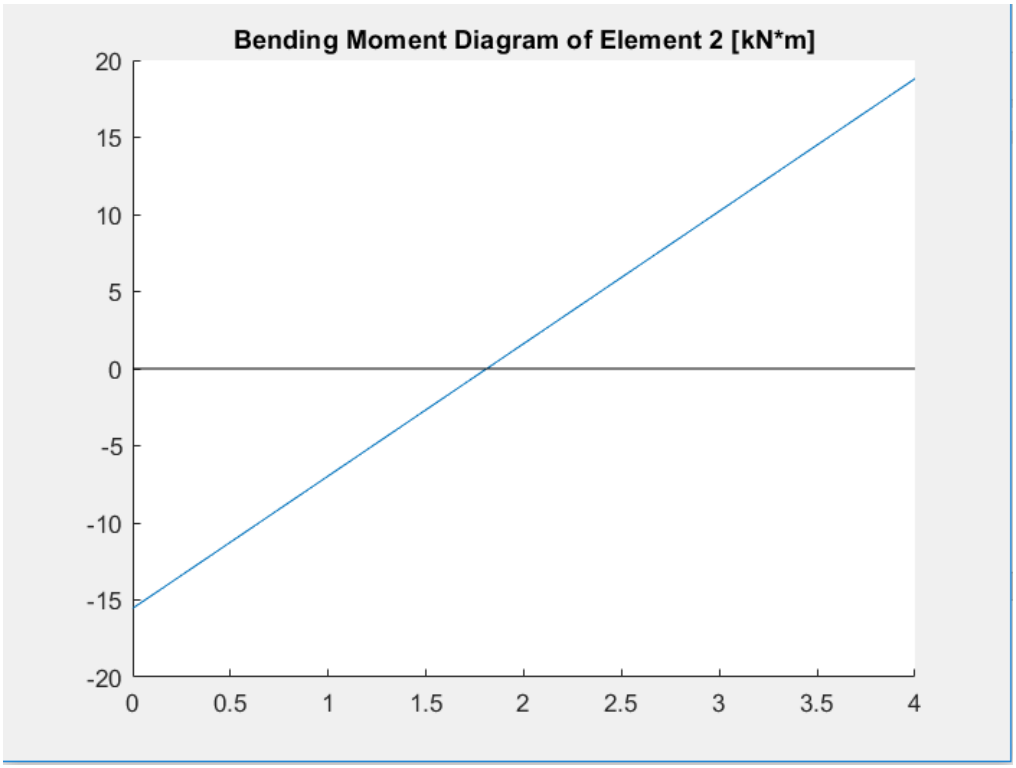

Figure 38: Bending moment diagram of element 2

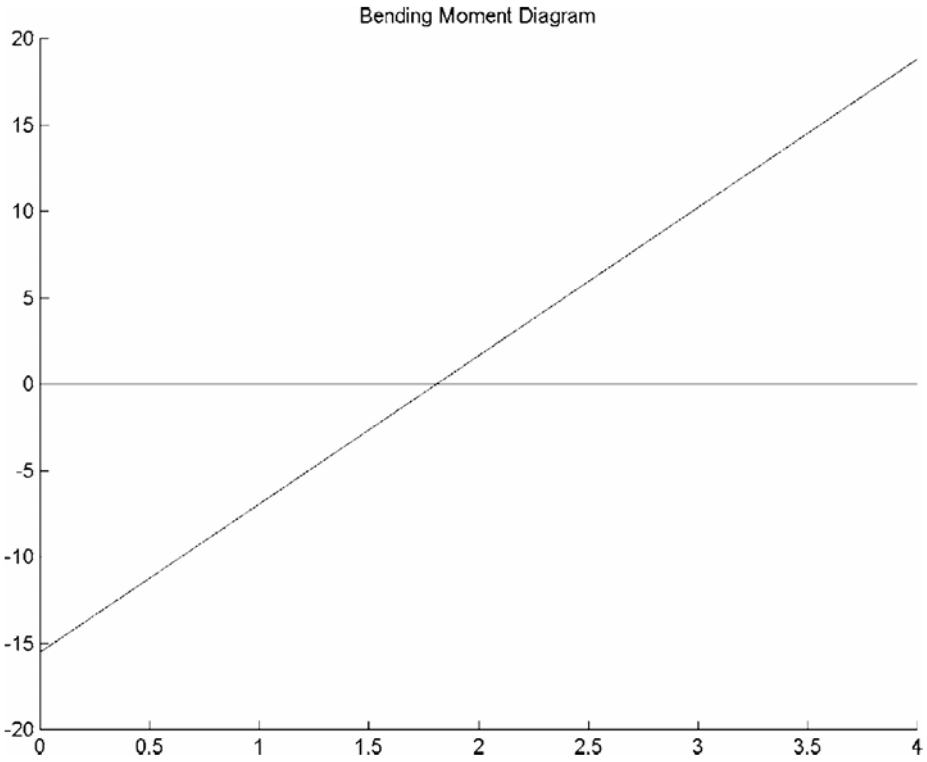

Figure 39: Bending moment diagram of element 2 [5]

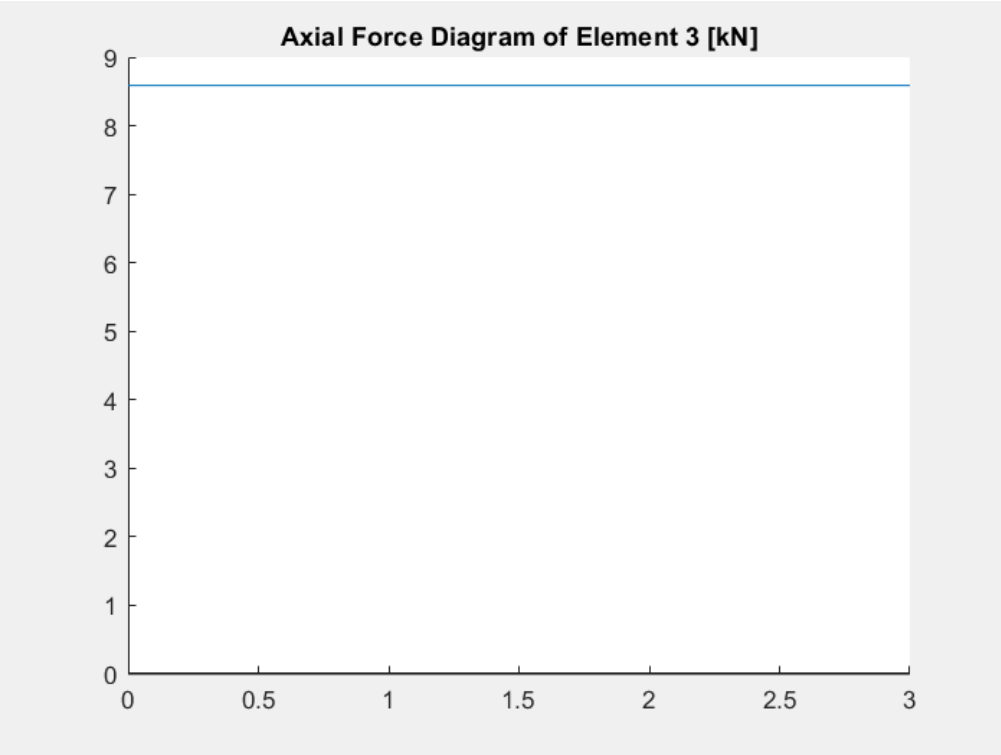

Figure 40: Axial force diagram of element 3

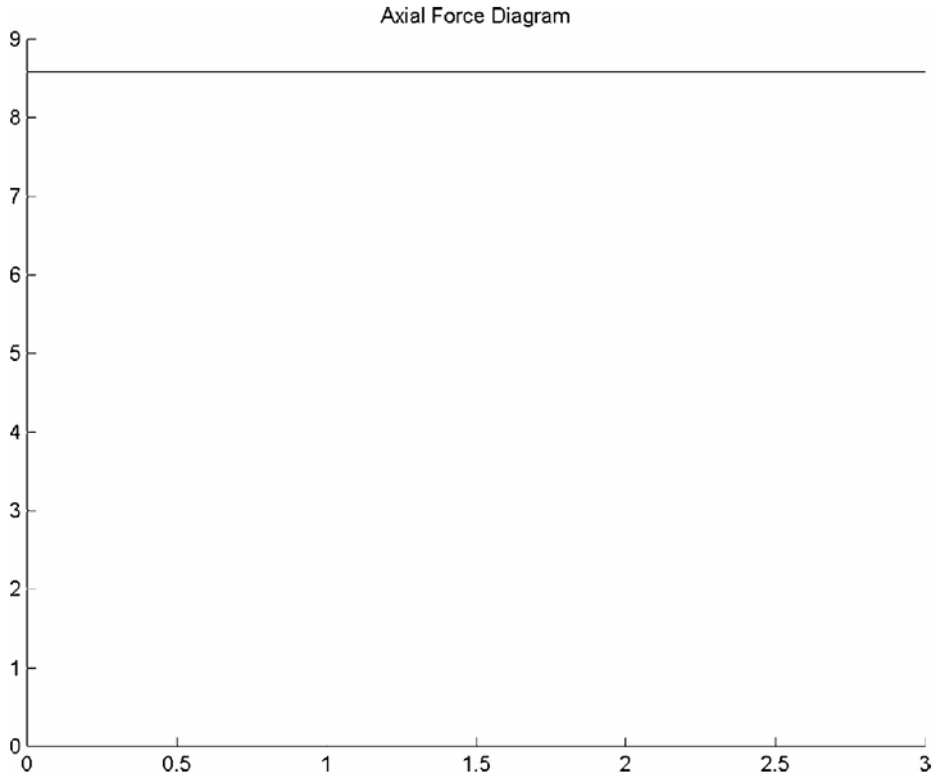

Figure 41: Axial force diagram of element 3 [5]

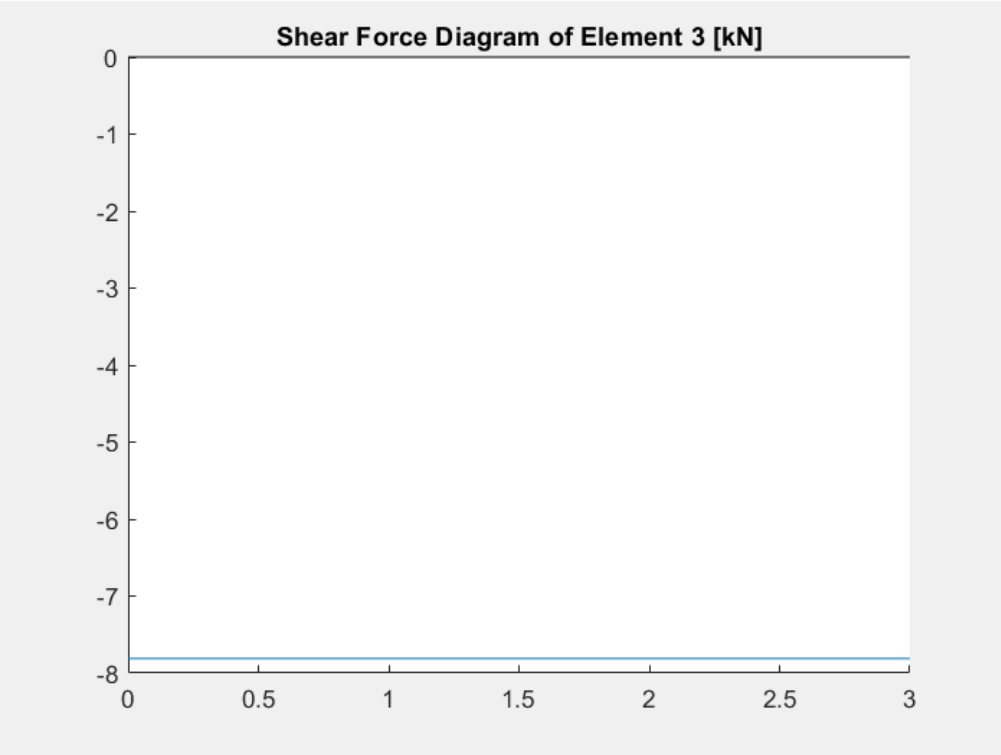

Figure 42: Shear force diagram of element 3

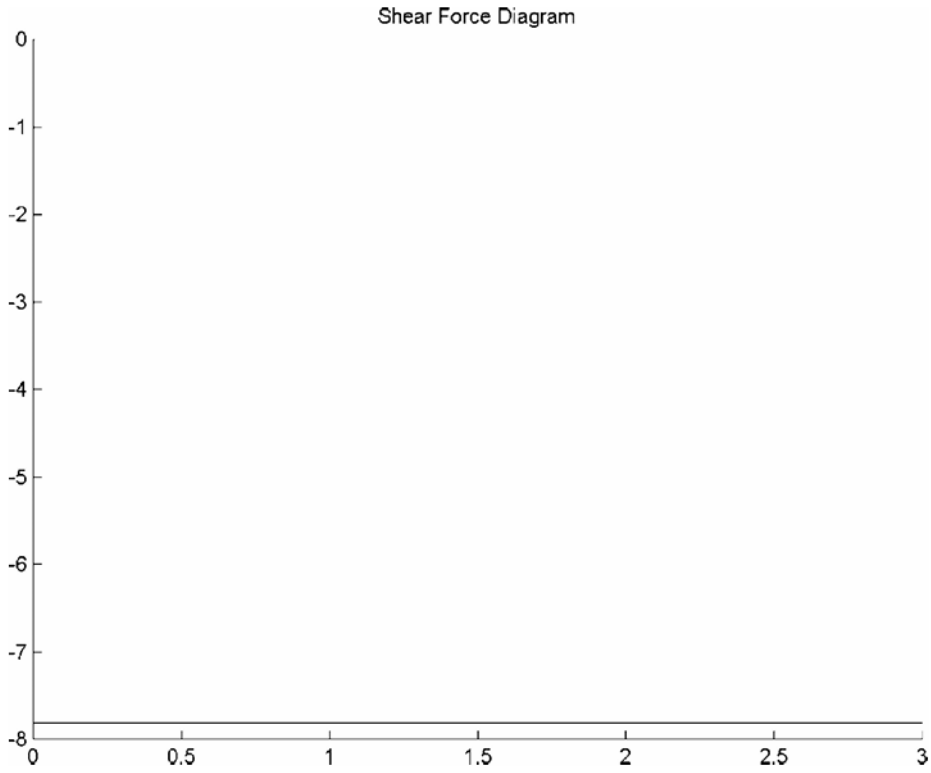

Figure 43: Shear force diagram of element 3 [5]

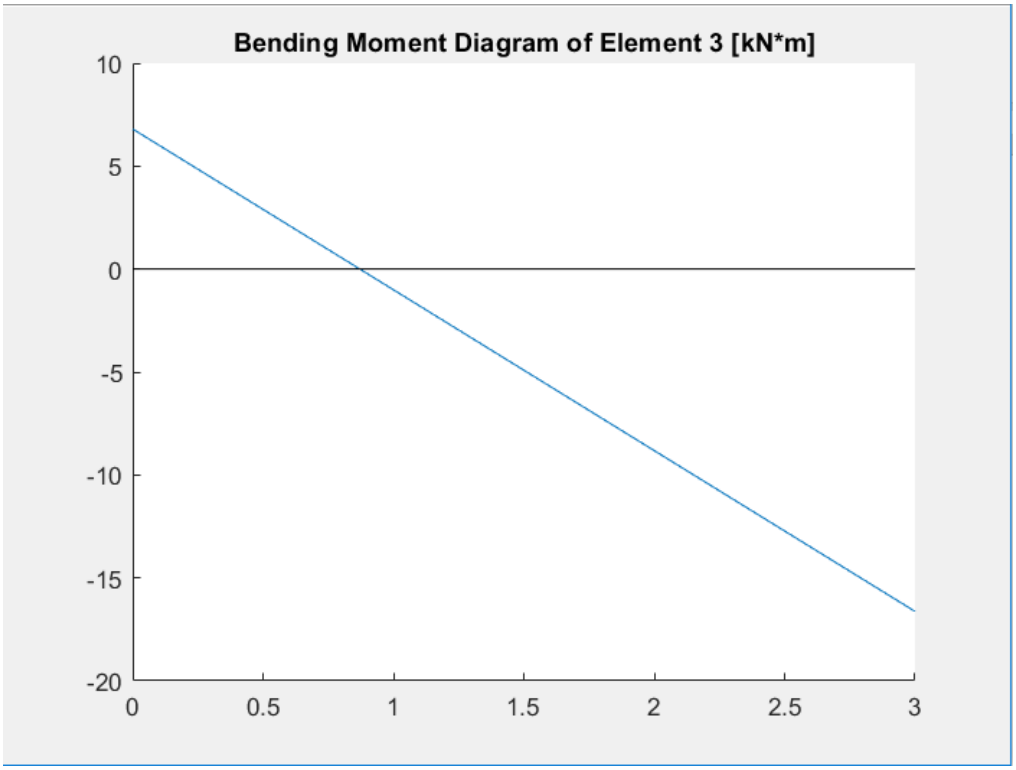

Figure 44: Bending moment diagram of element 3

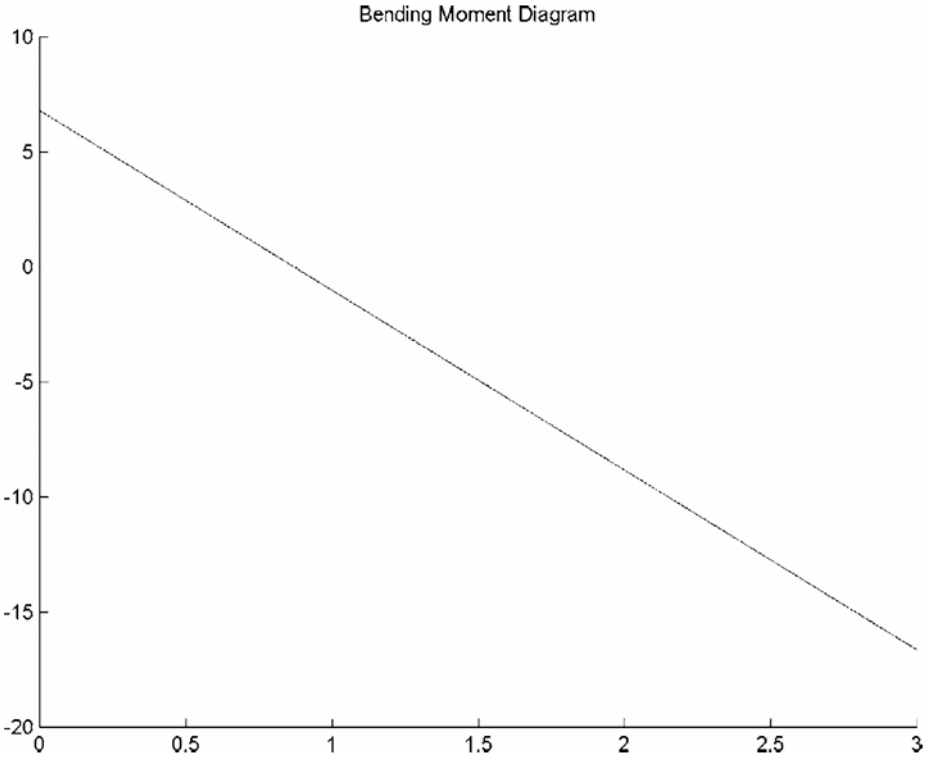

Figure 45: Bending moment diagram of element 3 [5]

Global stiffness matrix that calculated for solve problem at the background:

4.6667e+03	0	-7000	-4.6667e+03	0	-7000	0	0	0	0	0	0
0	1400000	0	0	-1400000	0	0	0	0	0	0	0
-7000	0	14000	7000	0	7000	0	0	0	0	0	0
-4.6667e+03	0	7000	1.0547e+06	0	7000	-1050000	0	0	0	0	0
0	-1400000	0	0	1.4020e+06	3.9375e+03	0	-1.9688e+03	3.9375e+03	0	0	0
-7000	0	7000	7000	3.9375e+03	24500	0	-3.9375e+03	5250	0	0	0
0	0	0	-1050000	0	0	1.0547e+06	0	7000	-4.6667e+03	0	7000
0	0	0	0	-1.9688e+03	-3.9375e+03	0	1.4020e+06	-3.9375e+03	0	-1400000	0
0	0	0	0	3.9375e+03	5250	7000	-3.9375e+03	24500	-7000	0	7000
0	0	0	0	0	0	-4.6667e+03	0	-7000	4.6667e+03	0	-7000
0	0	0	0	0	0	0	-1400000	0	0	1400000	0
0	0	0	0	0	0	7000	0	7000	-7000	0	14000

Figure 46:Global stiffness matrix

5. CONCLUSION

In this project I focused on develop a computer program for finite element analysis and design of planar frame structures. Last term, I learnt about what is finite element method, history of finite element method. And I searched how to do finite element analysis on spring, bar element, planar truss, spatial truss, beam element, plane frame, and spatial frame. I prepared my work plan week by week.

This term I started to write codes for my computer program. First, I asked from user that how many nodes and elements he or she has. Then I asked from user the required material properties and according to these, I wrote a function that calculates the angle and length of elements. After that I wrote codes to calculate element stiffness matrices and according to these matrices, I wrote a code for assemble all these elements stiffness matrices in one general matrix. After that I asked to user which boundary condition does user want to enter and get these values from user. Then by using entered nodal displacement vector, I wrote code to calculate nodal forces with Gauss elimination. I used these nodal force boundary conditions to calculate unknown nodal displacement vector. Then I showed all result to user and plot the element axial shear bending diagrams.

This project helped me to understand finite element analysis deeply and improve my MATLAB coding skills. At the end of my 2 terms diligent working, I have reached to my aim and I have developed a computer program for the finite element analysis and design of planar frame structures.

Also, we wrote codes for this project. So, cost analysis is not applicable for this project.

REFERENCES

[1] RAO, S.S., (2005), The Finite Element Method in Engineering, Fourth edition, Elsevier Inc, Florida, USA.

[2] ERHUNMWUN, I. D.; IKPONMWOSA, U. B. Review on finite element method. Journal of Applied Sciences and Environmental Management, 2017, 21.5: 999-1002.

[3] HUTTON, D.V., (2004), Fundamentals of Finite Element Analysis, 1st edition, McGraw Hill, New York, USA

[4] HIBBELER, R. C., (2017), Mechanics of Materials, 10th edition, Pearson Education, United States.

[5] KATTAN, P., (2008), MATLAB Guide to Finite Elements an Iterative Approach, 2nd edition, Springer Science + Business Media