A Project Report on

"MINI ANSYS SOFTWARE USING SIMULINK"

Submitted

In Partial Fulfillment of the Requirements For of the

B. Tech. in Robotics and Automation Engineering

Submitted by

Shivani (PRN: 2014111295)

Rohit Patil (PRN: 2014111289)

Anand Patil (PRN: 2014111290)

Gauri kadbane (PRN: 20141112)

Under the Guidance of

Mrs. Shubhada Mam

Department of Robotics and Automation

Bharati Vidyapeeth

(Deemed to be University)

College of Engineering, Pune

(2022-2023)

Bharati Vidyapeeth

(Deemed to be University)

College of Engineering, Pune

CERTIFICATE

This is to certify that,

Shivani (PRN: 2014111295)

Rohit Patil (PRN: 2014111289)

Anand Patil (PRN: 2014111290)

Gauri kadbane (PRN: 20141112)

have carried out the project entitled "MINI ANSYS SOFTWARE USING SIMULINK**"** under my guidance in partial fulfillment of the requirement for the degree of Bachelor of Technology in Robotics and Automation Engineering of Bharati Vidyapeeth (Deemed to be University), Pune during the academic year 2022-2023. To the best of my knowledge and belief, this work has not been submitted elsewhere for the award of any other degree.

Mrs. Shubhada Mam.	**Dr. K. B. Sutar**
(Project Guide)	(Co-guide and H.O.D.)

DECALARATION

I hereby declare that the project work entitled “MINI ANSYS SOFTWARE USING SIMULINK”, is a record of an original work done by Shivani Shinde , Rohit Patil, Anand Patil and Gauri Kadbane under the guidance of Mrs.Shubhada Mam Faculty Of Signals & Systems Bharati Vidyapeeth deemed to be University, College of Engineering , and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Robotics and Automation Engineering. The results embodied in this have not been submitted to any other University or Institute for the award of any degree or diploma.

ACKNOWLEDGEMENT

We would like to express our earnest gratitude to the supervisor Mrs.Shubhada Mam for her support, mentorship, and providing the opportunities to help us develop into a better researcher.

First and foremost, we would like to thank, the almighty and our Parents for providing us the strength and support in every possible way for successfully completing our Progress Work. Working with highly developed and experienced teachers' community at the Department of Robotics and Automation Engineering, Bharati Vidyapeeth (Deemed to be) University, College of Engineering, Pune will remain the most eventful and memorable experience of our lives.

We would also like to show our greatest appreciation to Dr. K. B. Sutar who helped us and showed us the correct path and thereby overcoming difficulties faced during preparation of this Project.

ABSTRACT

Mini ANSYS software will actually useful to find displacement solution of small problems related to different types of element.

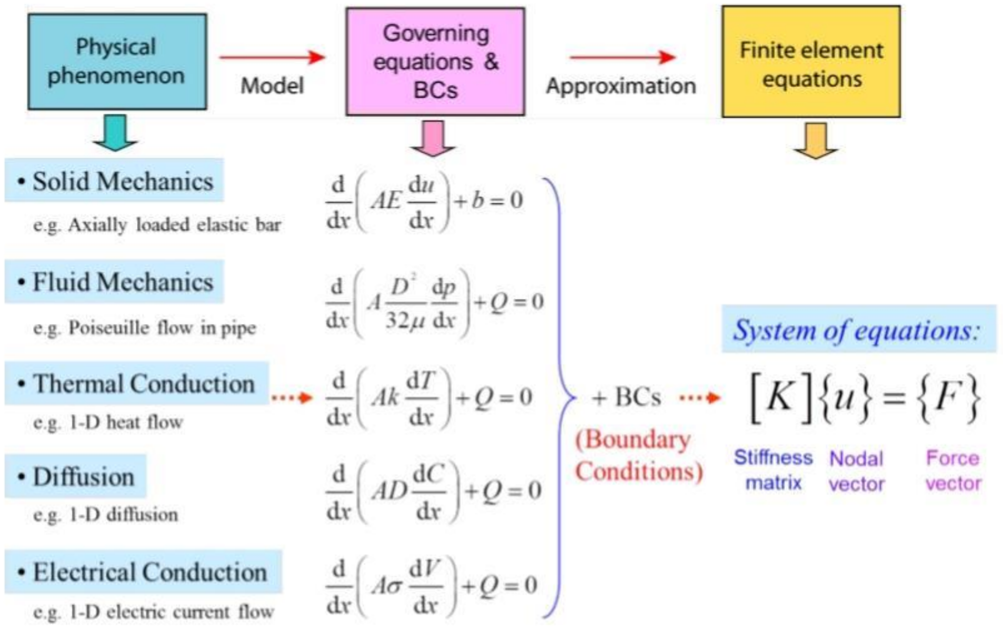
Here we are with idea of Analysis of Spring , Bar , and Bar element with different material which are divided into 3 nodes and 2 element. In this analysis we will find the the displacement on nodes using MATLAB Simulink.

We are going to use Gauss elimination method logic to find the displacement in the simulink
There are two kinds of finite element equation solvers : the direct and iteration methods.
The direct method is based on the Gaussian elimination method, and its original form was
Modified by various ways to reduce the number of operations in computing, to improve the
Matrix condition number for accuracy, and to reduce the storage space of the coefficient
Matrix.

INTRODUCTION

Ansys Mechanical finite element analysis software is used to simulate computer models of structures, electronics, or machine components for analyzing the strength, toughness, elasticity, temperature distribution, electromagnetism, fluid flow, and other attributes. Ansys is used to determine how a product will function with different specifications, without building test products or conducting crash tests. For example, Ansys software may simulate how a bridge will hold up after years of traffic, how to best process salmon in a cannery to reduce waste, or how to design a slide that uses less material without sacrificing safety.

The finite element method (FEM) is a numerical technique for solving a wide range of complex physical phenomena, particularly those exhibiting geometrical and material non-linearities (such as those that are often encountered in the physical and engineering sciences). These problems can be structural in nature, thermal (or thermo-mechanical), electrical, magnetic, acoustic etc. plus any combination of. It is used most frequently to tackle problems that aren't readily amenable to analytical treatments.



Physical phenomenon
Model
Governing equations & BCs
Approximation
Finite element equations
• Solid Mechanics
e.g. Axially loaded elastic bar
$\frac{d}{dx}\left(AE\frac{du}{dx}\right)+b=0$
• Fluid Mechanics
e.g. Poiseuille flow in pipe
$\frac{d}{dx}\left(A\frac{D^2}{32\mu}\frac{dp}{dx}\right)+Q=0$
• Thermal Conduction
e.g. 1-D heat flow
$\frac{d}{dx}\left(Ak\frac{dT}{dx}\right)+Q=0$
• Diffusion
e.g. 1-D diffusion
$\frac{d}{dx}\left(AD\frac{dC}{dx}\right)+Q=0$
• Electrical Conduction
e.g. 1-D electric current flow
$\frac{d}{dx}\left(A\sigma\frac{dV}{dx}\right)+Q=0$
+ BCs
(Boundary Conditions)
System of equations:
$[K]\{u\}=\{F\}$
Stiffness matrix
Nodal vector
Force vector

Fig:1

The premise is very simple; continuous domains (geometries) are decomposed into discrete, connected regions (or finite elements). An assembly of element-level equations is subsequently solved, in order to establish the response of the complete domain to a particular set of boundary conditions. The premise is very simple; continuous domains (geometries) are decomposed into discrete, connected regions (or finite elements). An assembly of element-level equations is subsequently solved, in order to establish the response of the complete domain to a particular set of boundary conditions.

The Direct Stiffness Method and the Stiffness Matrix

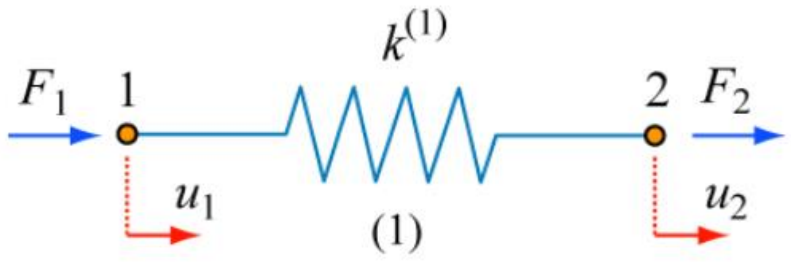
There are several finite element methods. These are the Direct Approach, which is the simplest method for solving discrete problems in 1 and 2 dimensions; the Weighted Residuals method which uses the governing differential equations directly (e.g. the Galerkin method), and the Variational Approach, which uses the calculus of variation and the minimisation of potential energy (e.g. the Rayleigh-Ritz method).

We analyse the Direct Stiffness Method here, since it is a good starting point for understanding the finite element formulation. We consider first the simplest possible element – a 1-dimensional elastic spring which can accommodate only tensile and compressive forces. For the spring system shown in Fig.2, we accept the following conditions:

· Condition of Compatibility – connected ends (nodes) of adjacent springs have the same displacements

· Condition of Static Equilibrium – the resultant force at each node is zero

· Constitutive Relation – that describes how the material (spring) responds to the applied loads



The constitutive relation can be obtained from the governing equation for an elastic bar loaded axially along its length:

$$\frac{d}{du}\left(AE\frac{\Delta l}{l_0}\right)+k=0 \tag{1}$$

$$\frac{\Delta l}{l_0}=\varepsilon \tag{2}$$

$$\frac{d}{du}(AE\varepsilon)+k=0 \tag{3}$$

$$\frac{d}{du}(A\sigma)+k=0 \tag{4}$$

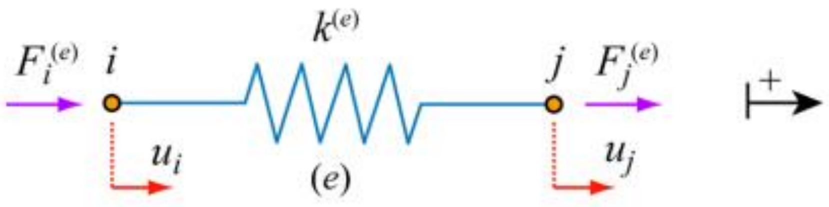
$$\frac{dF}{du}+k=0 \tag{5}$$

$$\frac{dF}{du}=-k \tag{6}$$

$$dF=-kdu \tag{7}$$

The spring stiffness equation relates the nodal displacements to the applied forces via the spring (element) stiffness. From here on in we use the scalar version of Eqn.7.

Derivation of the Stiffness Matrix for a Single Spring (Element



From inspection, we can see that there are two degrees of freedom in this model, u_i and u_j. We can write the force equilibrium equations:

$$k^{(e)}u_i - k^{(e)}u_j = F_i^{(e)} \tag{8}$$

$$-k^{(e)}u_i + k^{(e)}u_j = F_j^{(e)} \tag{9}$$

In matrix form

$$\begin{bmatrix} k^e & -k^e \\ -k^e & k^e \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \end{Bmatrix} = \begin{Bmatrix} F_i^{(e)} \\ F_j^{(e)} \end{Bmatrix} \tag{10}$$

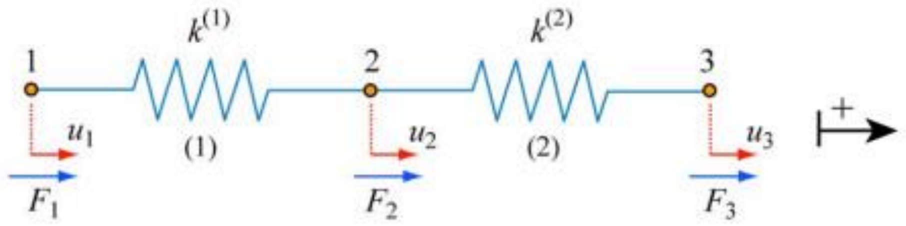
The order of the matrix is [2×2] because there are 2 degrees of freedom. Note also that the matrix is symmetrical. The 'element' stiffness relation is:

$$[K^{(e)}]\{u^{(e)}\} = \{F^{(e)}\} \tag{11}$$

Where $K^{(e)}$ is the element stiffness matrix, $u^{(e)}$ the nodal displacement vector and $F^{(e)}$ the nodal force vector. (The element stiffness relation is important because it can be used as a building block for more complex systems. An example of this is provided later.)

Derivation of a Global Stiffness Matrix

For a more complex spring system, a 'global' stiffness matrix is required – i.e. one that describes the behaviour of the complete system, and not just the individual springs.



From inspection, we can see that there are two springs (elements) and three degrees of freedom in this model, u_1, u_2 and u_3. As with the single spring model above, we can write the force equilibrium equations:

$$k^1u_1 - k^1u_2 = F_1 \tag{12}$$

$$-k^1u_1 + (k^1 + k^2)u_2 - k^2u_3 = F_2 \tag{13}$$

$$k^2u_3 - k^2u_2 = F_3 \tag{14}$$

In matrix form

$$\begin{bmatrix} k^1 & -k^1 & 0 \\ -k^1 & k^1 + k^2 & -k^2 \\ 0 & -k^2 & k^2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix} \tag{15}$$

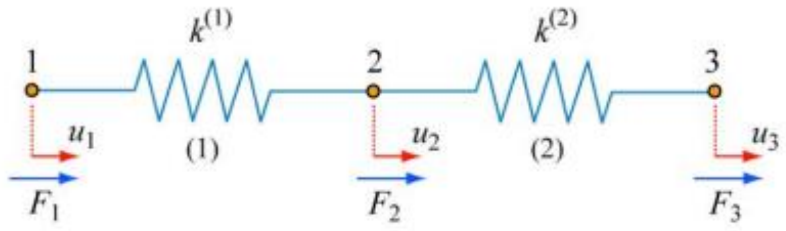
The 'global' stiffness relation is written in Eqn.16, which we distinguish from the 'element' stiffness relation in Eqn.11.

$$[K]\{u\} = \{F\} \tag{16}$$

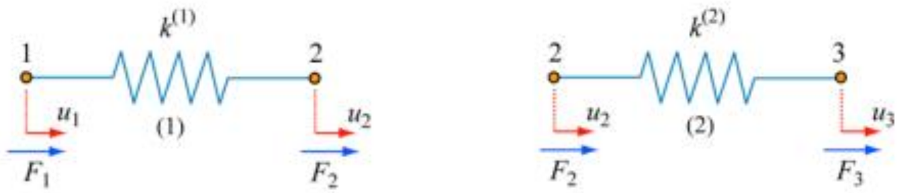
Note the shared k^1 and k^2 at $k22$ because of the compatibility condition at u_2. We return to this important feature later on.

Assembling the Global Stiffness Matrix from the Element Stiffness Matrices

Although it isn't apparent for the simple two-spring model above, generating the global stiffness matrix (directly) for a complex system of springs is impractical. A more efficient method involves the assembly of the individual element stiffness matrices. For instance, if you take the 2-element spring system shown,



split it into its component parts in the following way



and derive the force equilibrium equations

$$k^1u_1 - k^1u_2 = F_1 \tag{17}$$

$$k^1u_2 - k^1u_1 = k^2u_2 - k^2u_3 = F_2 \tag{18}$$

$k^2 u_3 - k^2 u_2 = F_3$ (19)

then the individual element stiffness matrices are:

$$\begin{bmatrix} k^1 & -k^1 \\ -k^1 & k^1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} \text{ and } \begin{bmatrix} k^2 & -k^2 \\ -k^2 & k^2 \end{bmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} F_2 \\ F_3 \end{Bmatrix} \tag{20}$$

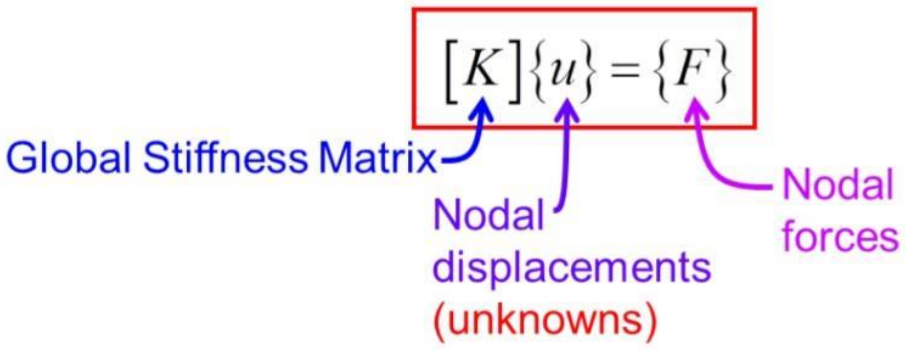
such that the global stiffness matrix is the same as that derived directly in Eqn.15:

$$\begin{bmatrix} k^1 & -k^1 & 0 \\ -k^1 & k^1 + k^2 & -k^2 \\ 0 & -k^2 & k^2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix} \tag{21}$$

(Note that, to create the global stiffness matrix by assembling the element stiffness matrices, $k22$ is given by the sum of the direct stiffnesses acting on node 2 – which is the compatibility criterion. Note also that the indirect cells (k_{ij}) are either zero (no load transfer between nodes i and j), or negative to indicate a reaction force.)

<u>Solving for (u)</u>

The unknowns (degrees of freedom) in the spring systems presented are the displacements u_{ij}. Our global system of equations takes the following form:



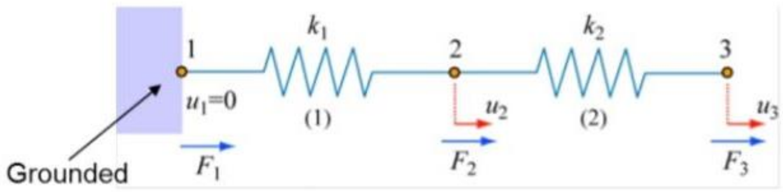
To find $\{u\}$ solve

$\{u\} = \{F\}[K]^{-1}$ (22)

Recall that $[k][k]^{-1} = \mathrm{I} = \text{Identitiy Matrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Enforcing Boundary Conditions

By enforcing boundary conditions, such as those depicted in the system below, $[K]$ becomes invertible (non-singular) and we can solve for the reaction force F_1 and the unknown displacements $\{u_2\}$ and $\{u_3\}$, for known (applied) F_2 and F_3.



No BC:

$$\begin{bmatrix} k^{(1)} & -k^{(1)} & 0 \\ -k^{(1)} & k^{(1)}+k^{(2)} & -k^{(2)} \\ 0 & -k^{(2)} & k^{(2)} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix} \rightarrow \det[K]=0$$

Enforce BC: $u_1 = 0$

$$\begin{bmatrix} k^{(1)} & -k^{(1)} & 0 \\ -k^{(1)} & k^{(1)}+k^{(2)} & -k^{(2)} \\ 0 & -k^{(2)} & k^{(2)} \end{bmatrix} \begin{Bmatrix} 0 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix} \rightarrow \det[K]\neq 0$$

$$[K] = \begin{bmatrix} k^1+k^2 & -k^2 \\ -k^2 & k^2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{27}$$

$$det[K] = ab - cd \tag{28}$$

$$det[K] = (k^1+k^2)k^2 - {k^2}^2 = k^1k^2 \neq 0 \tag{29}$$

Unique solutions for F_1, $\{u_2\}$ and $\{u_3\}$ can now be found

$$-k^1u_2 = F_1$$

$$(k^1+k^2)u_2 - k^2u_3 = F_2 = k^1u_2 + k^2u_2 - k^2u_3$$

$$-k^2u_2 + k^2u_3 = F_3$$

Unique solutions:

$$-k^{(1)}u_2 = F_1$$

$$u_2 = \frac{1}{k^{(1)}}(F_2+F_3)$$

$$u_3 = \frac{F_2}{k^{(1)}} + \left(\frac{1}{k^{(1)}} + \frac{1}{k^{(2)}}\right)F_3$$

In this instance we solved three equations for three unknowns. In problems of practical interest the order of $[K]$ is often very large and we can have thousands of unknowns. It then becomes impractical to solve for $\{u\}$ by inverting the global stiffness matrix. We can instead use Gauss elimination which is more suitable for solving systems of linear equations with thousands of unknowns.

MATLAB SIMULINK

Simulink is a block diagram environment for multidomain simulation and Model-Based Design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB, enabling you to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis.

Simulink is a simulation and model-based design environment for dynamic and embedded systems, integrated with MATLAB. Simulink, also developed by MathWorks, is a data flow graphical programming language tool for modelling, simulating and analyzing multi-domain dynamic systems. It is basically a graphical block diagramming tool with customizable set of block libraries.

It allows you to incorporate MATLAB algorithms into models as well as export the simulation results into MATLAB for further analysis.

Simulink supports –

- system-level design
- simulation
- automatic code generation
- testing and verification of embedded systems

There are several other add-on products provided by MathWorks and third-party hardware and software products that are available for use with Simulink.

The following list gives brief description of some of them –

- **Stateflow** allows developing state machines and flow charts.
- **Simulink Coder** allows the generation of C source code for real-time implementation of systems automatically.
- **xPC Target** together with **x86-based real-time systems** provide an environment to simulate and test Simulink and Stateflow models in real-time on the physical system.
- **Embedded Coder** supports specific embedded targets.
- **HDL Coder** allows to automatically generate synthesizable VHDL and Verilog.
- **SimEvents** provides a library of graphical building blocks for modelling queuing systems.

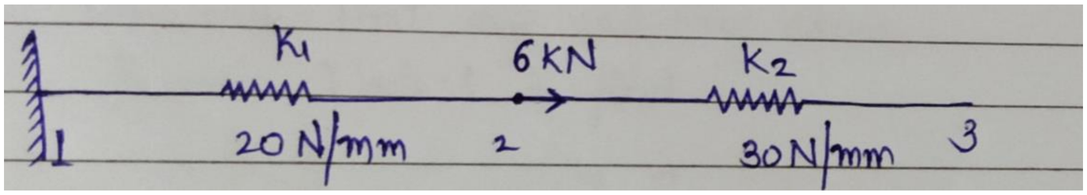
Simulink is capable of systematic verification and validation of models through modelling style checking, requirements traceability and model coverage analysis.

Simulink Design Verifier allows you to identify design errors and to generate test case scenarios for model checking.

IMPLEMENTATION

PROGRAMME CODE

Problem:

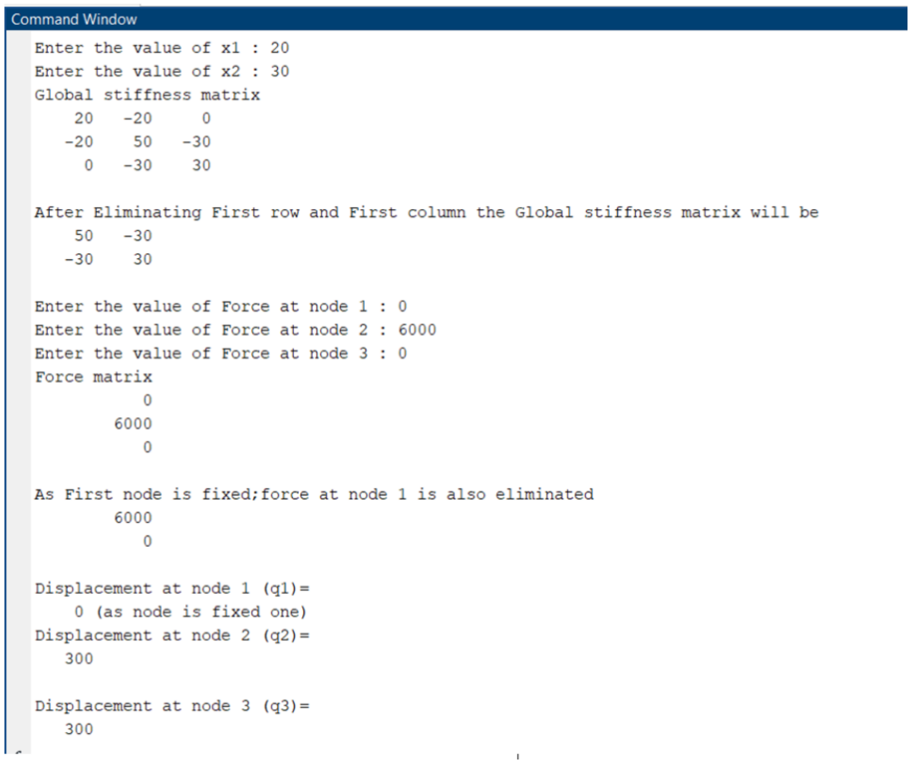


- For Spring Element:
  Code using different logic

  a. First logic:

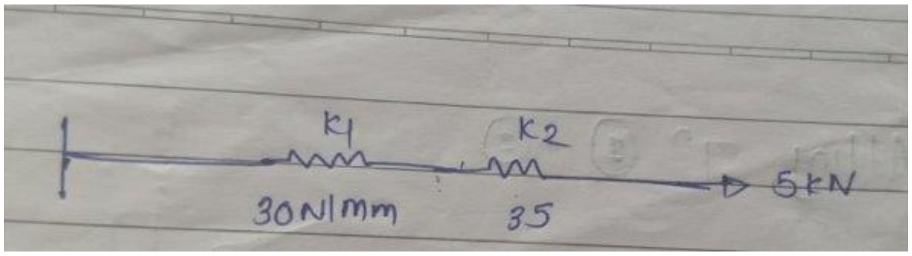
```
Clc;
Clear;
X1=input('Enter the value of x1 : ');
X2=input('Enter the value of x2 : ');
K1=[x1 -x1 0;-x1 x1 0; 0 0 0];
K2=[0 0 0;0 x2 -x2;0 -x2 x2];
K=k1+k2;
Disp('Global stiffness matrix');
Disp(K);
K(1,☺=[];
K(:,1)=[];
Disp('After Eliminating First row and First column the Global stiffness matrix will be');
Disp(K);
F1= input('Enter the value of Force at node 1 : ');
F2= input('Enter the value of Force at node 2 : ');
F3= input('Enter the value of Force at node 3 : ');
F=[f1;f2;f3];
Disp('Force matrix');
Disp(F);
F(1,☺=[];
Disp('As First node is fixed;force at node 1 is also eliminated');
Disp(F);
Q=inv(K)*F;
Disp('Displacement at node 1 (q1)= ')
Disp('    0 (as node is fixed one)');
Disp('Displacement at node 2 (q2)= ');
Q2=q(1,1);
Disp(q2);
Disp('Displacement at node 3 (q3)= ');
Q3=q(2,1);
Disp(q3);
```

OUTPUT:


Command Window

```
Enter the value of x1 : 20
Enter the value of x2 : 30
Global stiffness matrix
    20   -20     0
   -20    50   -30
     0   -30    30

After Eliminating First row and First column the Global stiffness matrix will be
    50   -30
   -30    30

Enter the value of Force at node 1 : 0
Enter the value of Force at node 2 : 6000
Enter the value of Force at node 3 : 0
Force matrix
           0
        6000
           0

As First node is fixed;force at node 1 is also eliminated
        6000
           0

Displacement at node 1 (q1)=
    0 (as node is fixed one)
Displacement at node 2 (q2)=
   300

Displacement at node 3 (q3)=
   300
```

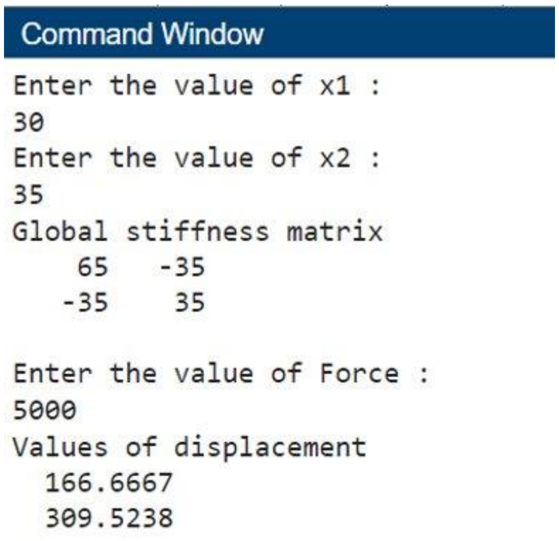
Problem:



b. Second logic (force only at end node):

```
clc;
clear;
x1=input(‘Enter the value of x1 : ‘);
x2=input(‘Enter the value of x2 : ‘);
k1=[x1 -x1 0;-x1 x1 0; 0 0 0];
k2=[0 0 0;0 x2 -x2;0 -x2 x2];
disp(‘Global stiffness matrix’);
K=[(x1+x2) -x2;-x2  x2];
Disp(K);
F1= input(‘Enter the value of Force : ‘);
F=[0; f1];
Q=inv(K)*F;
Disp(‘Values of displacement’);
Disp(q);
```

OUTPUT:

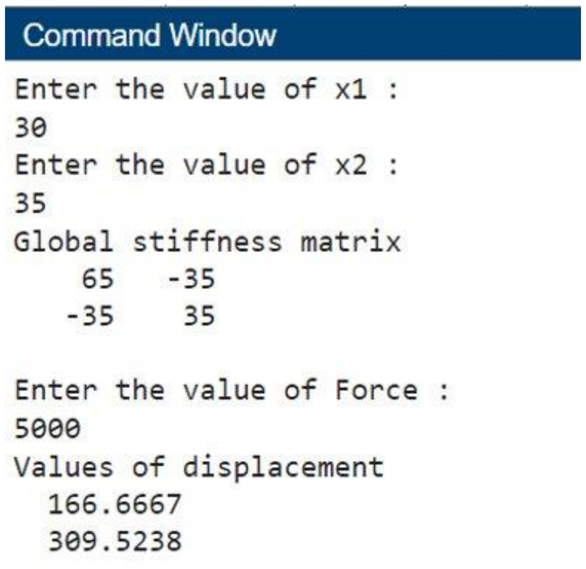


```
Command Window
Enter the value of x1 :
30
Enter the value of x2 :
35
Global stiffness matrix
    65   -35
   -35    35

Enter the value of Force :
5000
Values of displacement
  166.6667
  309.5238
```

c. Third logic (force only at end node):

```
Clc;
Clear;
X1=input('Enter the value of x1 : ');
X2=input('Enter the value of x2 : ');
C=(x1+x2);
Disp('Global stiffness matrix');
K=[c -x2;-x2  x2];
Disp(K);
F1= input('Enter the value of Force : ');
F=[0; f1];
Q=inv(K)*F;
Disp('Values of displacement');
Disp(q);
```

OUTPUT:

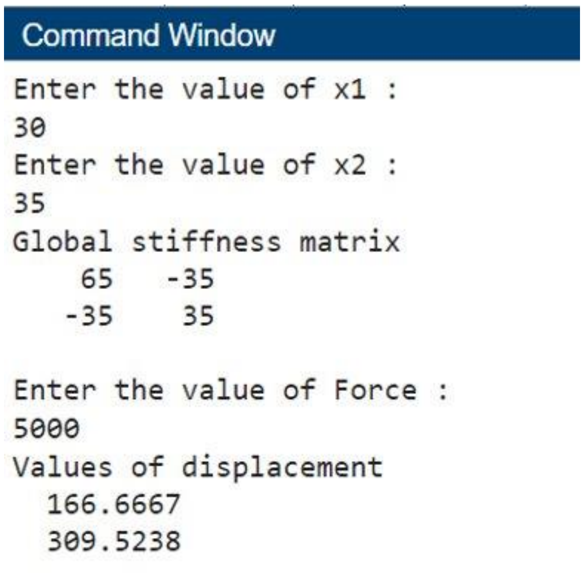


```
Enter the value of x1 :
30
Enter the value of x2 :
35
Global stiffness matrix
    65   -35
   -35    35

Enter the value of Force :
5000
Values of displacement
  166.6667
  309.5238
```

d. Fourth logic (force only at end node):

```
Clc;
Clear;
X1=input(‘Enter the value of x1 : ‘);
X2=input(‘Enter the value of x2 : ‘);
Disp(‘Global stiffness matrix’);
K=[ -x2;-x2  x2];
Disp(K);
F1= input(‘Enter the value of Force : ‘);
F=[0; f1];
Q=inv(K)*F;
Disp(‘Values of displacement’);
Disp(q);
```

OUTPUT:



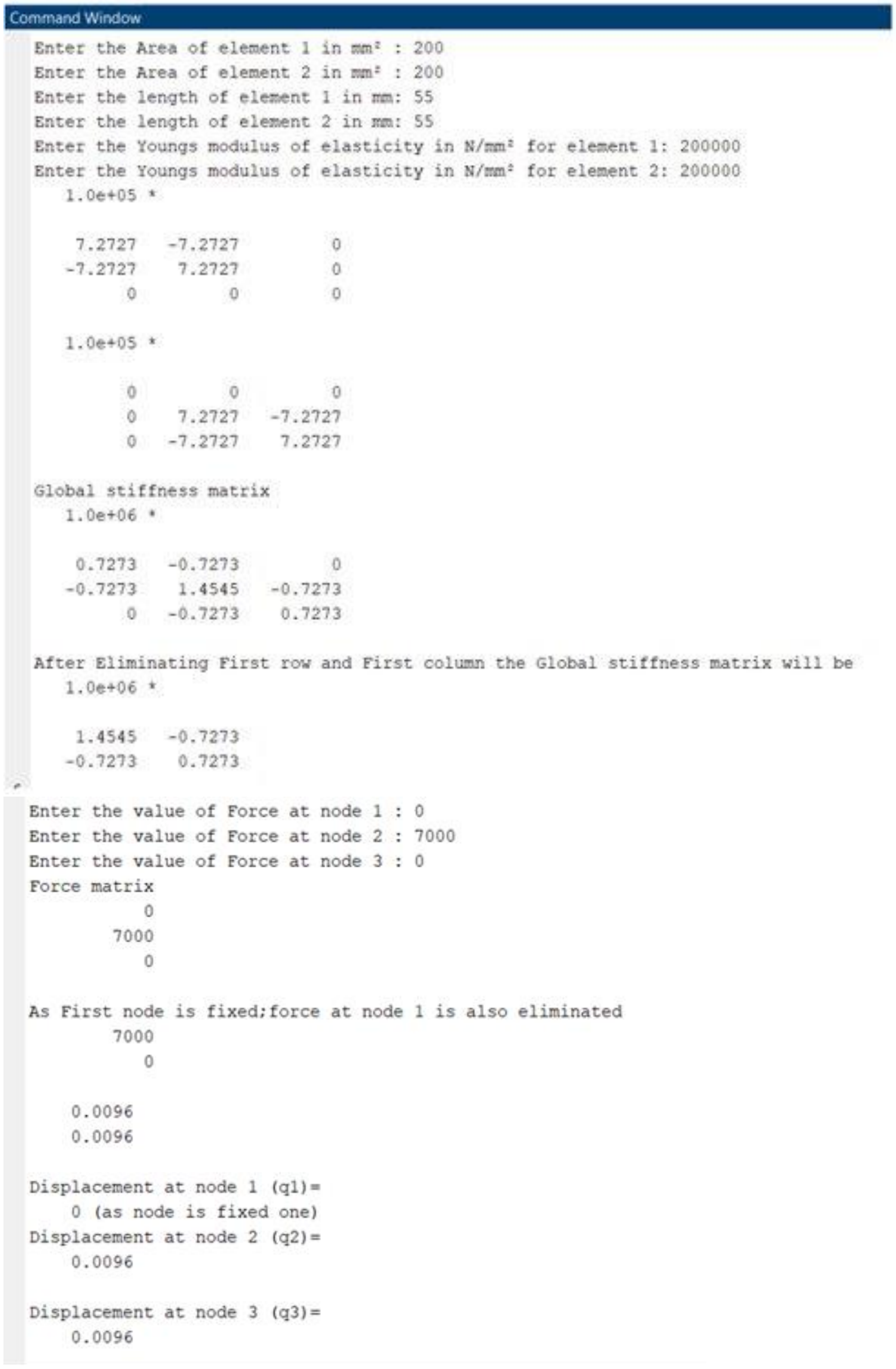
Command Window

```
Enter the value of x1 :
30
Enter the value of x2 :
35
Global stiffness matrix
    65   -35
   -35    35

Enter the value of Force :
5000
Values of displacement
  166.6667
  309.5238
```

- For Bar Element:

```
Clear;
A1=input('Enter the Area of element 1 in mm² : ');
A2=input('Enter the Area of element 2 in mm² : ');
L1=input('Enter the length of element 1 in mm: ');
L2=input('Enter the length of element 2 in mm: ');
E1=input('Enter the Youngs modulus of elasticity in N/mm² for element 1: ');
E2=input('Enter the Youngs modulus of elasticity in N/mm² for element 2: ');
K1=A1*E1*[1 -1 0;-1 1 0; 0 0 0];
K1=k1./L1;
Disp(k1);
K2=A2*E2*[0 0 0;0 1 -1;0 -1 1];
K2=k2./L2;
Disp(k2);
K=k1+k2;
Disp('Global stiffness matrix');
Disp(K);
K(1,:)=[];
K(:,1)=[];
Disp('After Eliminating First row and First column the Global stiffness matrix will be');
Disp(K);
F1= input('Enter the value of Force at node 1 : ');
F2= input('Enter the value of Force at node 2 : ');
F3= input('Enter the value of Force at node 3 : ');
F=[f1;f2;f3];
Disp('Force matrix');
Disp(F);
F(1,:)=[];
Disp('As First node is fixed;force at node 1 is also eliminated');
Disp(F);
Q=inv(K)*F;
Disp(q);
Disp('Displacement at node 1 (q1)= ')
Disp('    0 (as node is fixed one)');
Disp('Displacement at node 2 (q2)= ');
Q2=q(1,1);
Disp(q2);
Disp('Displacement at node 3 (q3)= ');
Q3=q(2,1);
Disp(q3);
```

OUTPUT:

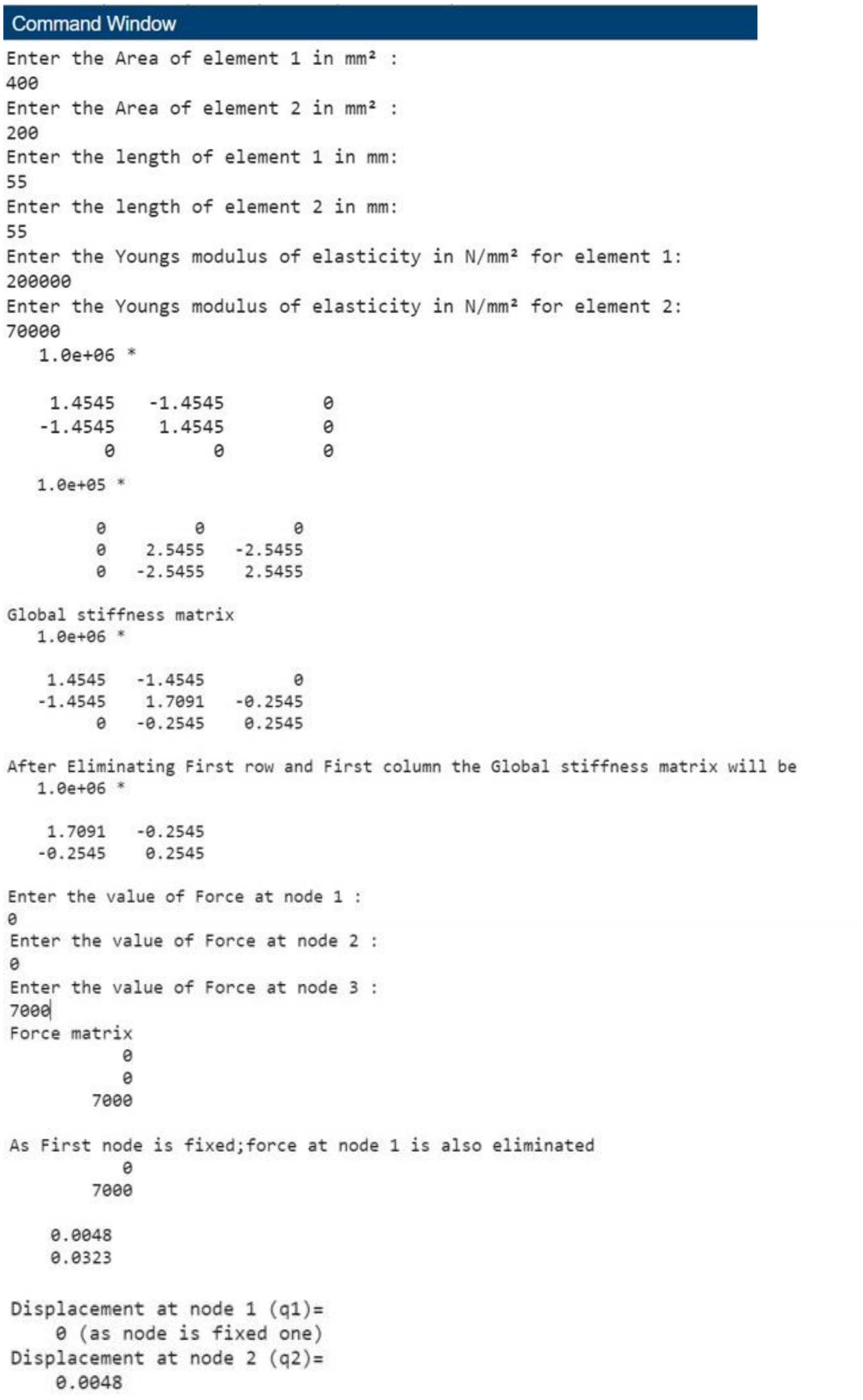

Command Window

```
Enter the Area of element 1 in mm² : 200
Enter the Area of element 2 in mm² : 200
Enter the length of element 1 in mm: 55
Enter the length of element 2 in mm: 55
Enter the Youngs modulus of elasticity in N/mm² for element 1: 200000
Enter the Youngs modulus of elasticity in N/mm² for element 2: 200000
   1.0e+05 *

    7.2727   -7.2727         0
   -7.2727    7.2727         0
         0         0         0

   1.0e+05 *

         0         0         0
         0    7.2727   -7.2727
         0   -7.2727    7.2727

Global stiffness matrix
   1.0e+06 *

    0.7273   -0.7273         0
   -0.7273    1.4545   -0.7273
         0   -0.7273    0.7273

After Eliminating First row and First column the Global stiffness matrix will be
   1.0e+06 *

    1.4545   -0.7273
   -0.7273    0.7273

Enter the value of Force at node 1 : 0
Enter the value of Force at node 2 : 7000
Enter the value of Force at node 3 : 0
Force matrix
           0
        7000
           0

As First node is fixed;force at node 1 is also eliminated
        7000
           0

    0.0096
    0.0096

Displacement at node 1 (q1)=
    0 (as node is fixed one)
Displacement at node 2 (q2)=
    0.0096

Displacement at node 3 (q3)=
    0.0096
```

- For Bar Element with different materials:

```
clear;
A1=input('Enter the Area of element 1 in mm² : ');
A2=input('Enter the Area of element 2 in mm² : ');
L1=input('Enter the length of element 1 in mm: ');
L2=input('Enter the length of element 2 in mm: ');
E1=input('Enter the Youngs modulus of elasticity in N/mm² for element 1: ');
E2=input('Enter the Youngs modulus of elasticity in N/mm² for element 2: ');
k1=A1*E1*[1 -1 0;-1 1 0; 0 0 0];
k1=k1./L1;
disp(k1);
k2=A2*E2*[0 0 0;0 1 -1;0 -1 1];
k2=k2./L2;
disp(k2);
K=k1+k2;
disp('Global stiffness matrix');
disp(K);
K(1,:)=[];
K(:,1)=[];
disp('After Eliminating First row and First column the Global stiffness matrix will be');
disp(K);
f1= input('Enter the value of Force at node 1 : ');
f2= input('Enter the value of Force at node 2 : ');
f3= input('Enter the value of Force at node 3 : ');
F=[f1;f2;f3];
disp('Force matrix');
disp(F);
F(1,:)=[];
disp('As First node is fixed;force at node 1 is also eliminated');
disp(F);
q=inv(K)*F;
disp(q);
disp('Displacement at node 1 (q1)= ')
disp('    0 (as node is fixed one)');
disp('Displacement at node 2 (q2)= ');
q2=q(1,1);
disp(q2);
disp('Displacement at node 3 (q3)= ');
q3=q(2,1);
disp(q3);
```

OUTPUT:



Command Window

```
Enter the Area of element 1 in mm² :
400
Enter the Area of element 2 in mm² :
200
Enter the length of element 1 in mm:
55
Enter the length of element 2 in mm:
55
Enter the Youngs modulus of elasticity in N/mm² for element 1:
200000
Enter the Youngs modulus of elasticity in N/mm² for element 2:
70000
   1.0e+06 *

    1.4545   -1.4545         0
   -1.4545    1.4545         0
         0         0         0

   1.0e+05 *

         0         0         0
         0    2.5455   -2.5455
         0   -2.5455    2.5455

Global stiffness matrix
   1.0e+06 *

    1.4545   -1.4545         0
   -1.4545    1.7091   -0.2545
         0   -0.2545    0.2545

After Eliminating First row and First column the Global stiffness matrix will be
   1.0e+06 *

    1.7091   -0.2545
   -0.2545    0.2545

Enter the value of Force at node 1 :
0
Enter the value of Force at node 2 :
0
Enter the value of Force at node 3 :
7000
Force matrix
           0
           0
        7000

As First node is fixed;force at node 1 is also eliminated
           0
        7000

    0.0048
    0.0323

Displacement at node 1 (q1)=
    0 (as node is fixed one)
Displacement at node 2 (q2)=
    0.0048
```

CONCLUSION

- Advantages :

  -Useful to find diplacement of element with 3 nodes
  -Time saving
  -Less effort
  -No need of mathematical calculation
  -Accurate and efficient

- Drawbacks:

  -Not able to find displacement of Element More than 3 node
  -Limited Accuracy as node increases after 3

- Modification

  -Analysis of more than 3 nodes can be done using array method
  -Reactions at nodes can also be found

REFERENCES

- https://www.ccg.msm.cam.ac.uk/system/files/documents/FEMOR_Lecture_1.pdf
- https://www.tutorialspoint.com/matlab/matlab_simulink.htm
- https://in.mathworks.com/help/simulink/
- https://matlab.mathworks.com/