Analysis of Statically Determinate Trusses using Exact Method (Joint Resolution Method) and Matrix Stiffness Method

A. Abdullahi[1], I. T. Yusuf[2], M. Abubakar[3], H. O. Aminulai[4], A. Yusuf[5] and B. Alhaji[6]

[1,3,4,5,6]Civil Engineering Department, Federal University of Technology, Minna, Nigeria

[2]Civil Engineering Department, University of Ilorin, Ilorin, Nigeria

katchali20@yahoo.com ityusuf4@gmail.com

Abstract

Matrix Stiffness Method (MSM) as a tool for static analysis of structures is premised on the principle of Finite Element Method (FEM), which in itself is a numerical/approximate method capable of giving only approximate results. However, Joint Resolution Method (JRM) is one of the most popular classical/exact methods of static analysis capable of giving exact results. This paper presents an analysis of a statically determinate 2-D truss using Exact/Joint Resolution Method (JRM) and Matric Stiffness Method (MSM) to ascertain the validity of the latter against the former. In the JRM, the support reactions and internal member forces were obtained from considerations of the equilibrium conditions of the entire truss and isolated joints respectively. On the other hand, a computer program was written in MATLAB 7.8.0 (R2009a) based on the principles of MSM for ease of computation and increased accuracy to solve for member forces and reactions of the same truss. The element properties were obtained and employed to calculate the element stiffness matrices, these were then assembled into the global stiffness matrix, from which the unknown displacements, member forces and support reactions were calculated. The results obtained from using both JRM and MSM were found to be exactly the same or very close, with percentage errors ranging between 0% and 3%. Hence MSM results as compared to JRM have 97% accuracy and above, and can therefore be relied upon.

Keywords

Statically Determinate, Joint Resolution Method, Matrix Stiffness Method, 2-D Truss, MATLAB,

USEP: Journal of Research Information in Civil Engineering, Vol.14, No.2, 2017

Analysis of Statically Determinate Trusses using Exact Method (Joint Resolution Method) and Matrix Stiffness Method

A. Abdullahi[1], I. T. Yusuf[2], M. Abubakar[3], H. O. Aminulai[4], A. Yusuf[5] and B. Alhaji[6]

[1,3,4,5,6]Civil Engineering Department, Federal University of Technology, Minna, Nigeria

[2]Civil Engineering Department, University of Ilorin, Ilorin, Nigeria

katchali20@yahoo.com ityusuf4@gmail.com

Abstract

Matrix Stiffness Method (MSM) as a tool for static analysis of structures is premised on the principle of Finite Element Method (FEM), which in itself is a numerical/approximate method capable of giving only approximate results. However, Joint Resolution Method (JRM) is one of the most popular classical/exact methods of static analysis capable of giving exact results. This paper presents an analysis of a statically determinate 2-D truss using Exact/Joint Resolution Method (JRM) and Matric Stiffness Method (MSM) to ascertain the validity of the latter against the former. In the JRM, the support reactions and internal member forces were obtained from considerations of the equilibrium conditions of the entire truss and isolated joints respectively. On the other hand, a computer program was written in MATLAB 7.8.0 (R2009a) based on the principles of MSM for ease of computation and increased accuracy to solve for member forces and reactions of the same truss. The element properties were obtained and employed to calculate the element stiffness matrices, these were then assembled into the global stiffness matrix, from which the unknown displacements, member forces and support reactions were calculated. The results obtained from using both JRM and MSM were found to be exactly the same or very close, with percentage errors ranging between 0% and 3%. Hence MSM results as compared to JRM have 97% accuracy and above, and can therefore be relied upon.

Keywords

Statically Determinate, Joint Resolution Method, Matrix Stiffness Method, 2-D Truss, MATLAB,

1. Introduction

Exact methods of analysis such as Joint Resolution Method, although have proven very useful over time, are found to be tedious, time consuming and highly prone to manual errors. This has led to the emergence of faster methods (e.g. Matrix Stiffness Method), which, although amenable to computer manipulations, are mostly premised on numerical methods which in themselves are approximate methods and give approximate results which must be checked/validated

A truss is a Civil Engineering Structure consisting of an assemblage of straight members connected together at their ends. These members are subjected to loads and reactions only at the joints (Megson, 2005). An ideal truss is one whose members develop only axial forces (tension and compression) when the truss is loaded (Kassimali, 2009).

For design requirements of safety, economy and aesthetic preservation as specified by BS 8110-1997 to be met, it is important that the truss be well designed, hence the internal forces in the members have to be correctly analyzed (Megson, 2005). It is worth mentioning that a structural design is only as good as the analysis that precedes it, which in turn depends on the accuracy of the method employed in the analysis.

Various methods of analysis can be used for trusses, some of which are exact/classical methods while others are approximate/numerical methods. Classical methods, such as joint resolution method and method of sections for truss analysis, make use of analytical formulations that are applied to simple elastic models and are often solved manually, while numerical methods such as finite element analysis (for continuum structures) from which Matrix Stiffness Method of analysis emanates (for discrete or discontinuous structures) easily suites computers since they require majorly matrix manipulations (Chandramouli, 2013). The method of joint resolution is basically suited for the analysis of statically determinate structures and uses the free-body-diagram of joints in the structure to determine the forces in each member by using the force balance in the horizontal (x) and vertical (y) coordinates of the Cartesian plane at each of the joints in the truss structure. Matrix Stiffness Method on the other hand, is particularly suited for computer-automated analysis of complex structures including the statically indeterminate type as its steps are more definite rather than arbitrary. It is a

matrix method that makes use of the member's stiffness relations for computing member forces and displacements in structures (Kharagpur, 2008). In applying this method, the system is discretized as a set of simpler, idealized elements interconnected at the nodes. The material stiffness properties of these elements are then, through matrix mathematics, compiled into a single matrix equation which governs the behavior of the entire idealized structure, thus member forces of the truss as well as reactive forces are obtained. nIn this wise, the comparison of results as generated by both methods is worthwhile.

2. Methodology

Figure 1 shows a truss with thirteen (13) members and eight (8) nodes. This truss is analysed using JRM and MSM. Both methods involve the use of member angles, member lengths and node coordinates that can be gotten from basic trigonometric and mathematical theorems.

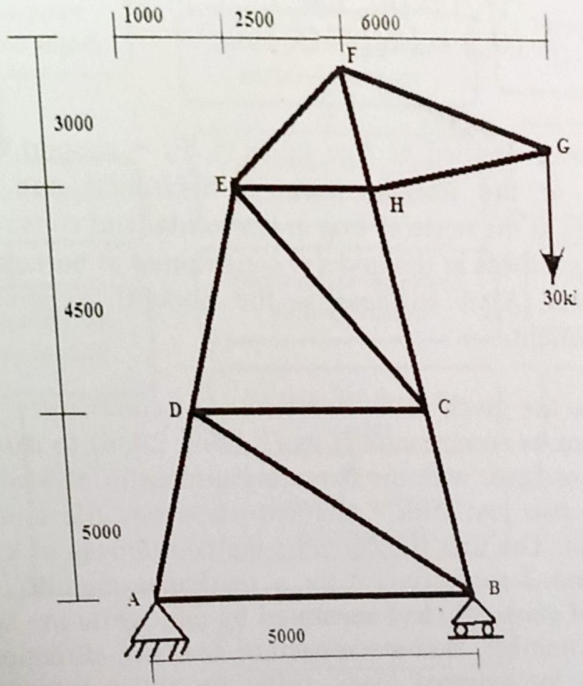

Figure 1: A 12.5m by 9.5m Statically Determinate 2-D Truss

2.1 Joint Resolution Method (JRM)

This method relies on the fact that a structure in equilibrium has all its constituent joints in equilibrium. The structure was descretized into joints/nodes, which were denoted by letters or numbers; equations of static equilibrium were then employed to obtain all forces emanating from and coming into each joint. This process was repeated until all member forces and reactions were obtained as specified by (Hibbeler, 2006).

2.2 Matrix Stiffness Method (MSM) using MATLAB

This method uses the matrix equation 2.1 and 2.2 to obtain member forces, support reactions as well as nodal displacements.

$$\begin{pmatrix} F_u \\ F_c \end{pmatrix} = \begin{bmatrix} K_{uu} & K_{uc} \\ K_{cu} & K_{cc} \end{bmatrix} \begin{pmatrix} U_u \\ U_C \end{pmatrix} \quad (1)$$

This implies that;

$$(F_u) = [K_{uu}](U_u)$$
$$(U_u) = [K_{uu}]^{-1}(F_u) \quad (2)$$

(Ray and Joseph, 1975

Where F_u = external force applied at free node(s), F_c = support Reactive force(s) K_{uu}= stiffness at the node(s) free in horizontal and vertical directions, K_{uc} = stiffness at the node(s) free in horizontal and constrained in vertical directions, K_{cu}= stiffness at the node(s) constrained in horizontal and free in vertical directions, K_{cc}= stiffness at the node(s) constrained in U_c = constrained displacements

In using MSM to analyze the given truss, some lines of codes were written for the MATLAB program as recommended by (Kattan, 2006) to determine the member forces in accordance with the operating principles of MSM. For this MATLAB program, two text files - element-properties file and load-properties file are required. The first file contains the coordinates in x and y directions for nodes 1 and 2 respectively, for a particular member, cross sectional area, modulus of elasticity, and separated by tab, while the second text file contains the node number, load in x-direction, load in y-direction and support constraints (null- for external loads, roller for roller support and pinned for pin support) respectively, for nodes with loads/reactions and separated by tab.

2.3 Flow Chart for both Methods

The procedures for the use of JRM and MSM (along with the MATLAB program) as employed in the truss analysis are as shown in the flow chart presented in Figure 2. Figure 2: Flowchart employed

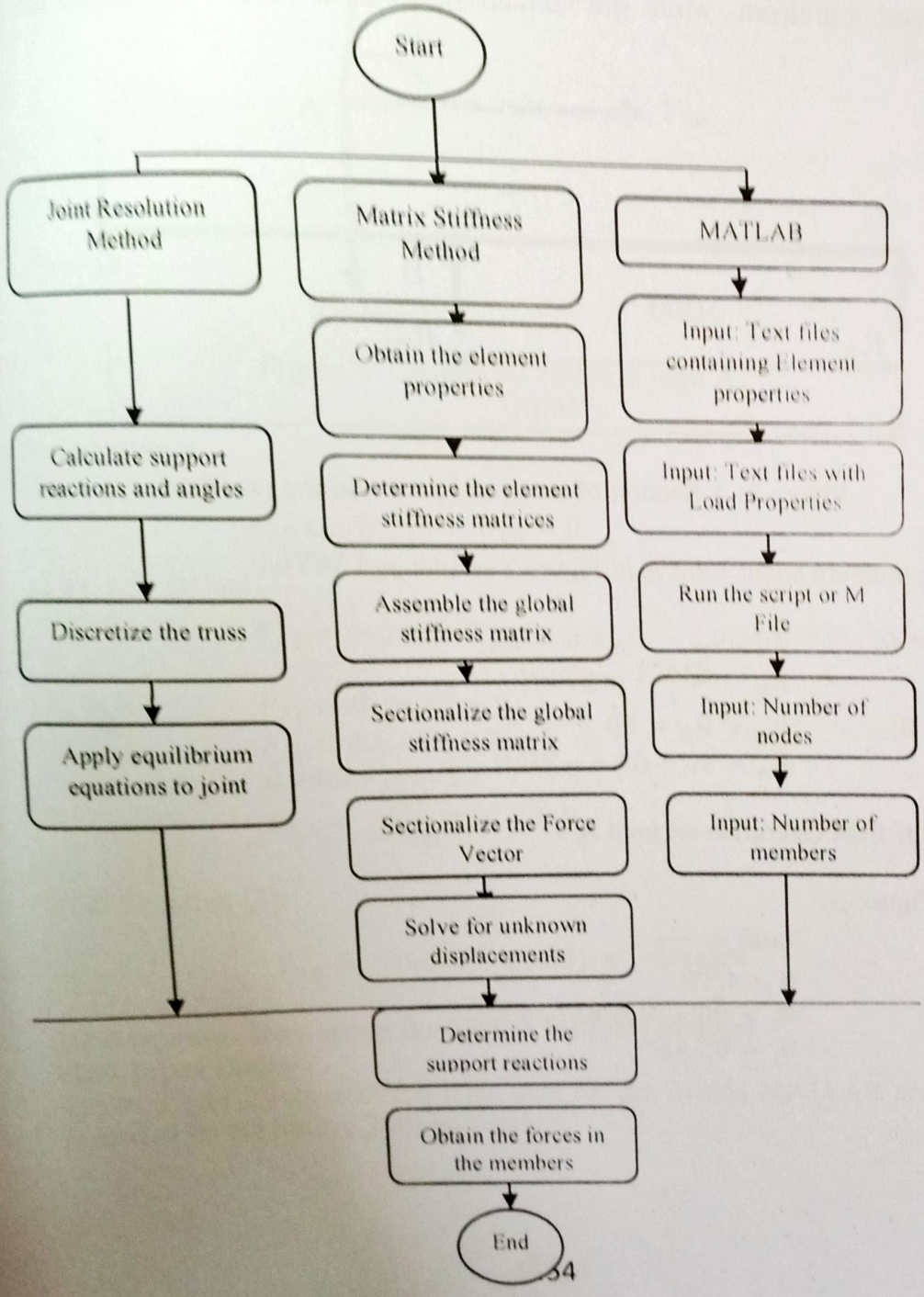

Figure 2: Flowchart employed in the analysis

3. Results and Discussions

3.1 Responses from using the Method of Joint Resolution

Figure 3 shows the external forces acting on truss AB and the reactions developed therefrom, while the concurrent force at joint A are shown in Figure 4.

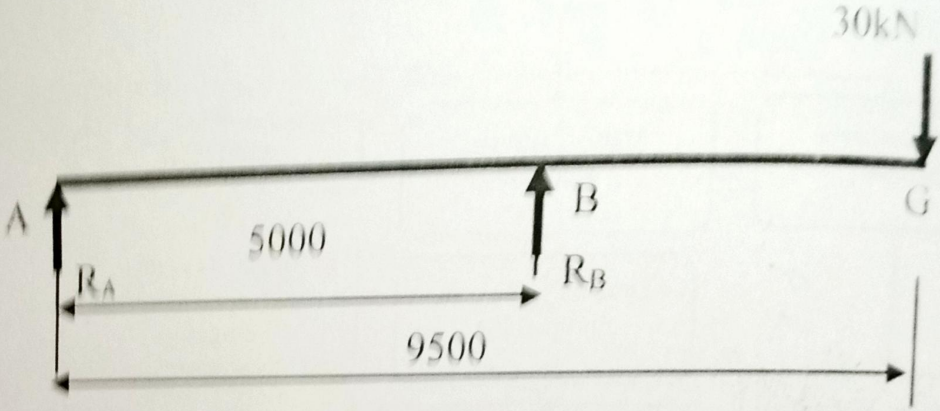

Figure 3: A Section of external forces and the reactions

Taking moment about Joint A in Figure 3 results in,

$$\Sigma M_A = 0; \quad 30 \times 9.5 - R_B \times 5 = 0$$

$$R_B = \frac{30 \times 9.5}{5} = 57 \text{ kN}$$

$$\Sigma F_y = 0; \quad R_A + R_B = 30$$

$$R_A = 30 - 57 = -27\text{kN}$$

A typical JRM Operation on joint A

From Figure ,3,

$$\tan\theta = \frac{1}{12.5}$$

$$\theta = 4.57^o$$

$$\theta_1 = 90 - \theta \equiv 90 - 4.57$$

$$\theta_1 = 85.43^o$$

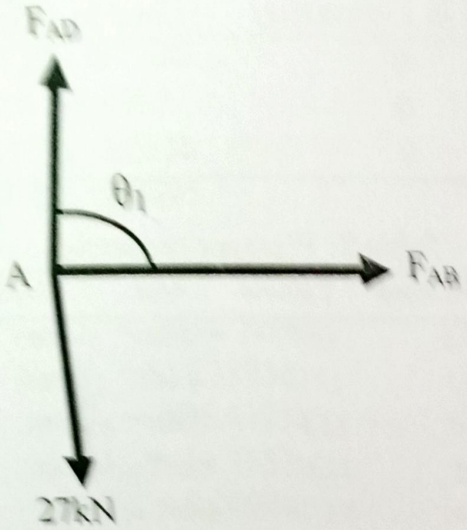

Figure 4: Concurrent forces at joint A

From Figure 4,

$\Sigma F_x = 0$: $F_{AD}\cos\theta_1 + F_{AB} = 0$

$F_{AD}\cos 85.43 + F_{AB} = 0$

$0.0797\,F_{AD} + F_{AB}$

And,

$$F_{AB} = -0.0797F_{AD} \quad (3)$$

$\Sigma F_y = 0$: $F_{AD}\sin\theta_1 - 27 = 0$

$F_{AD}\sin 85.43 - 27 = 0$

$0.9968F_{AD} = 27$

$$F_{AD} = \frac{27}{0.9968} = 27.09\text{kN}$$

From Equation (3),

$$F_{AB} = -0.0797\,(27.09) = -2.16\text{kN}$$

3.2 Responses from using the Matrix Stiffness Method

First Input Data

Tables 1 and 2 represent the input data for the written MATLAB program premised on the Matrix Stiffness Method.

$K(GF) = (1/\alpha)^*$

1	2	3	4
0.1522	-0.0380	-0.1522	0.0380
-0.0380	0.0095	0.0380	-0.0095
-0.1522	0.0380	0.1522	-0.0380
0.0380	-0.0095	-0.0380	0.0095

Displacements in mm

$U_1 = 1228.0975\alpha$; $U_2 = -5434.2496\alpha$; $U_3 = 2028.4194\alpha$;
$U_4 = -704.6498\alpha$; $U_5 = 517.3803\alpha$; $U_6 = 268.6877\alpha$;
$U_7 = 331.3803\alpha$; $U_8 = -696.4495\alpha$; $U_9 = -59.7653\alpha$;
$U_{10} = -354.5233\alpha$; $U_{11} = -14.7653\alpha$; $U_{12} = 137.4793\alpha$;
$U_{13} = -10.8000\alpha$; $U_{14} = 0.0000\alpha$; $U_{15} = 0.0000\alpha$
$U_{16} = 0.0000\alpha$; where $\alpha = L/AE$

Reactive Forces

$R_{14} = 57.0000$kN
$R_{15} = -0.0000$kN
$R_{16} = -27.0000$kN

Member Forces

$F(GF) = 59.93kN$, $F(HG) = 60.17kN$ (C); F(EF) = 73.92 kN (T);
F(HF) = 69.61 kN (C); F(EH) = 60.00 kN (C); F(DE) = 38.37 kN (T);
$F(CE)$ = 21.00 kN (T); $F(CH)$ = 85.19 kN (C); $F(DC)$ = 11.25 kN (C);
$F(AD)$ = 27.09 kN (T); $F(AB)$ = 2.16 kN (C); $F(BD)$ = 15.29 kN (T);
$F(BC)$ = 68.74 kN (C);

3.3 Comparison of Responses from both Methods

Table 3 shows the forces in each member of the truss using JRM and MSM. The table shows that the forces in members AB, AD, DC, CH and GF are exactly the same while forces in members BD, BC, DE, CE, EF, EH, HF and HG differ by a little percentage, with 2.91% as the highest percentage error in the member forces which may have resulted from accumulated approximations. It can therefore be deduced from the results that the Matrix Stiffness Method has at least 97% accuracy as compared to the Joint Resolution Method. The reactions at nodes A and B are 57 and -27 kN, respectively for Joint resolution and Matrix Stiffness methods, correspondingly as presented in Table 4.

Table 3: Forces in each member using both methods

Member	Method of Joint Resolution (kN)	Matrix Stiffness Method (kN)	Difference in Forces (kN)	Percentage Difference
AB	2.16(C)	2.16(C)	0.00	0%
AD	27.09(C)	27.09(C)	0.00	0%
BD	15.23(T)	15.23(T)	0.00	0%
BC	68.73(C)	68.74(C)	0.00	0%
DE	38.43(T)	38.37(T)	0.06	0%
DC	11.25(C)	11.25(T)	0.00	0%
CE	21.02(T)	21.00(T)	0.02	0%
CB	85.09(C)	85.09(C)	0.00	0%
EF	73.90(T)	73.92(T)	0.02	0%
EH	60.86(C)	60.00(C)	1.86	2%
HF	69.13(C)	69.61(C)	0.48	0%
HG	60.16(C)	60.17(C)	0.01	0%
GF	59.95(T)	59.95(T)	0.00	0%

Table 4: Reactions as obtained from both methods

Reactions	Method of Joint Resolution (kN)	Matrix Stiffness Method (kN)
RA	57	57
RB	-27	-27

4. Conclusions

The analysis of the 2-D statically determinate truss was successfully carried out. It was based on the Joint Resolution Method (an exact method) and Matrix Stiffness Method (an approximate method) to determine the member forces and support reactions. The Joint Resolution Method used was essentially done manually, while MATLAB 7.8.0 (R2009a) was employed to write a computer program on the basis of the Matrix Stiffness procedure. F For the Joint Resolution Method, the different member forces were obtained by resolving each joint one after the other until all the internal forces were gotten. For the Matrix Stiffness Method, the element and load properties were gotten and placed in the text files, the program was then run. afterwards, the number of nodes and members were entered as the inputs to

Appendix I

Written Functions used in the Program MATLAB (R2009a)

To calculate the Individual Member Stiffnesses

```
functionmemstiff = assembly( lengt,area,modulus,cos_angle,sin_angle)
%ASSEMBLY collates and returns the individual member stiffnesses
%  Function ASSEMBLY helps to develop the individual member stiffnesses
%  of a truss, once the element properties are gotten and passed in the
%  right order.
%     Calling sequence:
%     memstiff = assembly(length,area,modulus,angle,node1,node2 )
%     Define variables:
%     constant = stiffness constant

    constant = area * modulus/lengt;
            memstiff =
[cos_angle^2cos_angle * sin_angle -(cos_angle^2) - (cos_angle* sin_angle)
cos_angle * sin_anglesin_angle^2 -(cos_angle* sin_angle) -(sin_angle^2)
 -(cos_angle^2) -(cos_angle * sin_angle) cos_angle^2cos_angle* sin_angle-(cos_angle*
sin_angle) -(sin_angle^2) cos_angle * sin_anglesin_angle^2] .*constant;
End
```

To calculate the Global stiffness Matrix

```
function [ globstiff ] = assembled(globstiff,memstiff,node1,node2 )
%ASSEMBLED collates and returns the entire global stiffness
%  Function ASSEMBLED helps to develop the global stiffness matrix for
%  the entire truss structure, once the member stiffnesses and node
%  numbers of the members are known and passed in the right order.

%     Calling sequence:
%     globstiff = assembled(globstiff,memstiff,node1,node2 )

%      Define variable:
%      position = Array containing the sequential values of the degrees
%      of freedom for that member

 position = [(2 * node1)-1 (2 * node1) (2 * node2)-1 (2*node2)];
globstiff(position, position) = globstiff(position, position) + memstiff;
end
```

To calculate the Internal Member Forces

```
function [ meforce ] =
memforce( area, modulus, lengt, cos_angle, sin_angle, u, node1, node2)
%MEMFORCE collates and returns the internal member forces of a truss
%  Function MEMFORCE helps to calculate the internal member forces in a
%  truss once the element properties are known and passed in the right
%  order.
%     Calling Sequence:
%     meforce =
```

```
memforce( area, modulus, lengt, cos_angle, sin_angle, u, node1, node2)
%  Define Variable:
%  displacement = displacement vector for each member
  displacement =
[(u((2 * node2)-1) - u((2 * node1)-1)); (u(2 * node2) - u(2 * node1))];
meforce =
(area * modulus /lengt) .* ([cos_anglesin_angle] * displacement);
end
```

Appendix Ii

MATLAB (R2009a) script (Matrix Stiffness Method for 2-D truss)

```
% Script file :Matrixstiffness.m
% Purpose:
%    This program calculates the member forces of a 2-D truss from a
%    given set of data for each element.
% Define Variables:
%    nodes = The number of nodes in the truss.
%    member = The number of members in the truss.
%    area = The area of the member.
%    modulus = modulus of the member.
%    x1coord,y1coord,x2coord,y2coord = Coordinates in x and y directions
%    for the first and second nodes of a member respectively.
%    node1,node2 = The first and second nodes of a member respectively
%    node = The nodes with either external forces or support reactions
%    loadx,loady = Loads in x and y-directions on a node respectively.
%    constraints = conditions- stating either null for external loads,
%    pinned for pin support or roller for roller support.
%    memstiff = Cell array containing the individual member stiffnesses
%    meforce = Array containing the internal member forces.
%    dof = The total number of degrees of freedom for the truss.
%    globstiff = The global stiffness for the entire truss.
%    pins, rollers = The number of pinned and roller supports in the
%    truss respectively.
%    lengt = Array containing the lengths of each member.
%    cos_angle, sin_angle = The cos and the sine of the angle formed
%    respectively by each member.
%    i, m, k = loop index.
%    restnode = The number of restricted degrees of freedom.
%    loads = Array containing external loads.
%    position = Array containing the degrees of freedom for each member.
%    consi = Arrays containing the restricted degrees of freedom.
%    kcu = Stiffness corresponding to constrained-unconstrained.
%    kuu = Stiffness corresponding to unconstrained-unconstrained.
%    uu = Unknown displacements.
%    react_force = Support Reaction Forces.
%    u = Total displacements.
```

```
nodes = input ('enter no of nodes: ');
member = input ('enter no of members: ');
[x1coord,y1coord,x2coord,y2coord,area,modulus,node1,node2] = ...
textread ('C:/Users/Dania/Desktop/elementproperties.txt','%d%d%d%d%d%d%d%d');
[node,loadx,loady,constraints] =
 textread('C:/Users/Dania/Desktop/loadproperties.txt','%d%d%d%s');
memstiff = cell(member,1);
meforce = zeros(member, 1);
dof = 2 * nodes;
globstiff = zeros (dof);
 pins = 0;
 rollers = 0;
lengt = zeros(member);
cos_angle = zeros(member);
sin_angle = zeros(member);
for i = 1:member,
lengt(i) =
sqrt((x2coord(i) - x1coord(i))^2 + (y2coord(i) -     y1coord(i))^2);
cos_angle(i) = (x2coord(i) - x1coord(i))/lengt(i);
sin_angle(i) = (y2coord(i) - y1coord(i))/lengt(i);
end,
for i = 1:max(size(node)),
ifstrcmpi('roller',constraints(i)),
       rollers = rollers + 1;
elseifstrcmpi('pinned',constraints(i)),
          pins = pins + 1;
end,
end,
end,
restnode = 2 * pins + rollers;
 loads = zeros((dof - restnode), 1);
 letters = ['G','F','E','H','C','D','B','A'];
for m = 1:member,
memstiff{m} =
assembly( lengt(m),area(m),modulus(m),cos_angle(m),sin_angle(m));
fprintf ('K(%s%s) \n', letters(node1(m)), letters(node2(m)));
   position =
[(2 * node1(m))-1 (2 * node1(m)) (2 * node2(m))-1 (2*node2(m))];
firstPos = position(1);
secondPos = position(2);
thirdPos = position(3);
fourthPos = position(4);
fprintf('%10d%10d%10d%10d \n', firstPos, secondPos, thirdPos, fourthPos);
disp(memstiff{m});
globstiff = assembled(globstiff,memstiff{m},node1(m),node2(m));
end,
disp('K =');
disp(globstiff);
const = zeros (1,restnode);
```

```
[illegible]
```