

Statics Truss Problem

2.1 Statics

We are going to start our discussion of Finite Element Analysis (FEA) with something very familiar. We are going to look at a simple statically determinate truss. Trusses are characterized by linear elements (beams) which are pinned together at their ends. The pinned connection offers **no** resistance to rotation so there is **no** moment around the joint.

Consider the frame shown below. On this frame we assume that elements 1, 2, 4, and 5 are all equal in length.

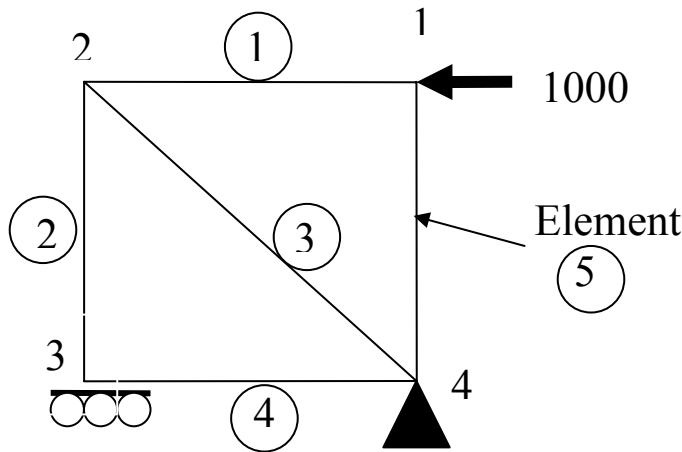


Figure 1 - Square truss with constraints and a load.

The members in the structure are numbered with circles around them. We call these members elements. The points where the elements connect are called nodes. In the problem above there are 5 elements and 4 nodes.

We can draw free body diagrams at the nodes. They are shown in Figure 2. There is always one force per element. In the diagram, we are using the variables F for the forces in the elements. The subscripts on these forces refer to the element number. The variables R represent the reactions. Their subscripts are assigned arbitrarily.

We know that the sum of the forces at each node must equal zero. There are two equations for each point, the horizontal equation and the vertical equation. We can write the equations as:

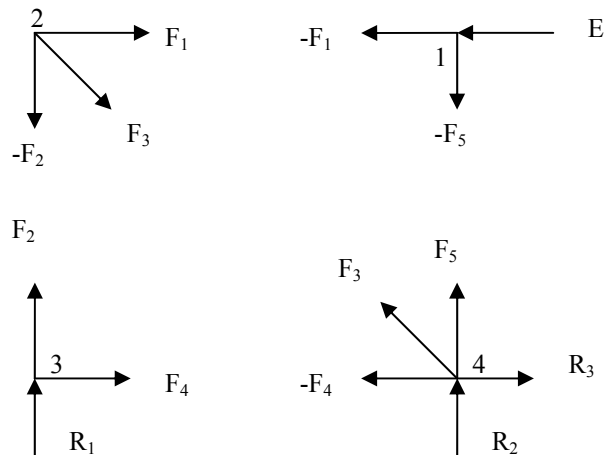


Figure 2 - Free body diagram of the truss

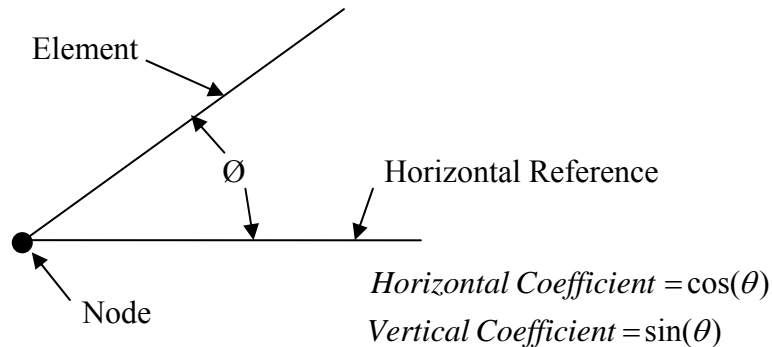
Node	F ₁	F ₂	F ₃	F ₄	F ₅	R ₁	R ₂	R ₃	External	Sum
1H	- F ₁								-E	= 0
1V					- F ₅				0	= 0
2H	F ₁		+ .707F ₃						0	= 0
2V		- F ₂	- .707F ₃						0	= 0
3H				+ F ₄					0	= 0
3V		F ₂				+ R ₁			0	= 0
4H			- .707F ₃	- F ₄				+ R ₃	0	= 0
4V			+ .707F ₃		+ F ₅		+ R ₂		0	= 0

Figure 3 - The equations defining the forces and reactions written in a matrix form.

Each row in the table shows the equation for summing either the horizontal or vertical forces about a node. As you know from statics, the forces must sum to zero.

Each row lists all of the unknowns. The equations are written to keep the unknown terms in their respective columns. The column on the left list the equations needed to solve the forces in the system. The number in this column refers to the node, the H for horizontal forces, and V for vertical forces. Empty cells in the equation table indicate the force or reaction specified in the column does not apply to that particular node. These empty cells could be filled with zeros.

The coefficients and sign of the coefficients for the element forces can be readily determined by looking at the direction of the element with respect to the node we are analyzing. This is shown in the diagram below.



There are several things to notice about the equations above:

1. In this particular problem, there are 5 elements and one unknown force per element. There are also 3 unknown reactions. This gives us a total of 8 unknowns in the problem. There are 2 equations for each node, a vertical equation and a horizontal equation which gives us a total of 8 equations. We can solve this system since we have as many equations as we have unknowns. In general, problems of this type must satisfy the equation shown below if they are solvable.

$$2 * \text{Nodes} = \text{Elements} + \text{Reactions}$$

- If we add another element as shown at the right, there will be 8 equations and 9 unknowns. The problem will no longer be statically determinate and cannot be solved using the technique we are discussing.
- In this type of problem, there must be at least 3 reactions. Two reactions are required to eliminate the X and Y translation and another reaction is required to eliminate any rotation of the object. The basic premise is that the object cannot move. It is in static equilibrium.

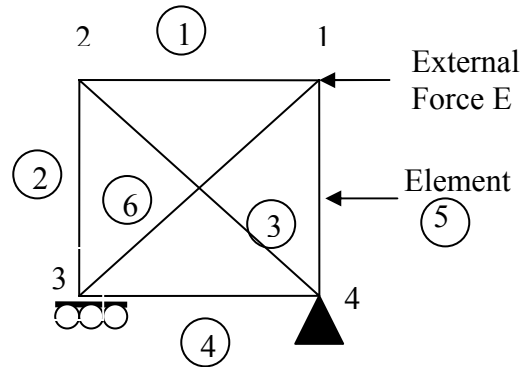


Figure 4 - Statically indeterminate truss.

2.2 Matrix Notation

The equations above can be rewritten in matrix notation as shown below. Here we are moving the external forces to the other side of the equal sign which means we must change its sign.

$$\begin{bmatrix}
 -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
 1 & 0 & .707 & 0 & 0 & 0 & 0 & 0 \\
 0 & -1 & -.707 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & -.707 & -1 & 0 & 0 & 0 & 1 \\
 0 & 0 & .707 & 0 & 1 & 0 & 1 & 0
 \end{bmatrix}
 \times
 \begin{Bmatrix}
 F_1 \\
 F_2 \\
 F_3 \\
 F_4 \\
 F_5 \\
 R_1 \\
 R_2 \\
 R_3
 \end{Bmatrix}
 =
 \begin{Bmatrix}
 1,000 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0
 \end{Bmatrix}
 \quad (2.1)$$

Or as:

$$[M] \times \{F\} = \{E\} \quad (2.2)$$

↖ External Forces
↖ Forces and Reactions
↖ Matrix based upon shape of the structure

This structure is common to many types of engineering problems. The left-hand-side is completely dependent upon the geometry and the right-hand-side upon the driving forces. If the geometry does not change, we can examine many load cases without

changing the left-hand-side of the equation. This can lead to solution efficiencies we will discuss later.

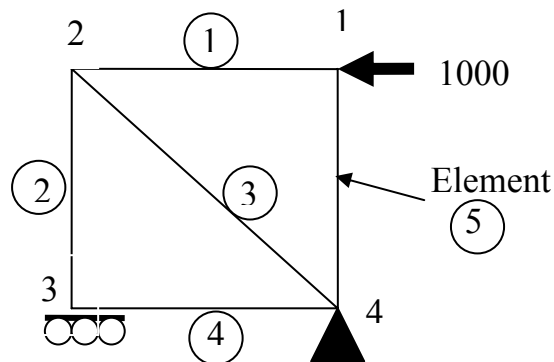
Problems like this can easily be solved in MATLAB. If we define M as the matrix and E as the loading forces on the right hand side of the equations, we can solve with:

$$M \backslash E$$

$$\begin{matrix} -1000 \\ -1000 \\ 1414 \\ 0 \\ 0 \\ 1000 \\ -1000 \\ 1000 \end{matrix}$$

Referring to equation 2.1 we see that:

$$\begin{matrix} F_1 = -1000 \\ F_2 = -1000 \\ F_3 = 1,414 \\ F_4 = 0 \\ F_5 = 0 \\ R_1 = 1000 \\ R_2 = -1000 \\ R_3 = 1000 \end{matrix}$$



The force subscript refers to the element the force is acting on. A negative value is a compressive force and a positive value is a tensile force.

We can divide these forces by the cross sectional area of the elements to compute the stresses in each element.

$$\sigma = F / A \tag{2.3}$$

We can compute the total displacement of the length of the element with the equation:

$$Displacement = \frac{L\sigma}{E} = \frac{LF}{AE} \tag{2.4}$$

Where:

- L = the length of the element
- F = the force on the element
- σ = the stress in the element
- A = the cross sectional area of the element
- E = Young's modulus for the element material

2.2 Developing a Systematic Method for Solving

This type of problem lends itself to a computer solution but if you generate the matrix by hand, it will be a very slow process to solve a large problem. A more systematic method is needed. We will re-examine the problem we just solved to see if we can develop such a system.

First, we number the nodes and define their coordinates. We will place the origin for the coordinate system at node 3 with X in the horizontal direction and Y in the vertical. We could place this coordinate system anywhere but this is a convenient location. The placement of the coordinate system is shown in the figure below.

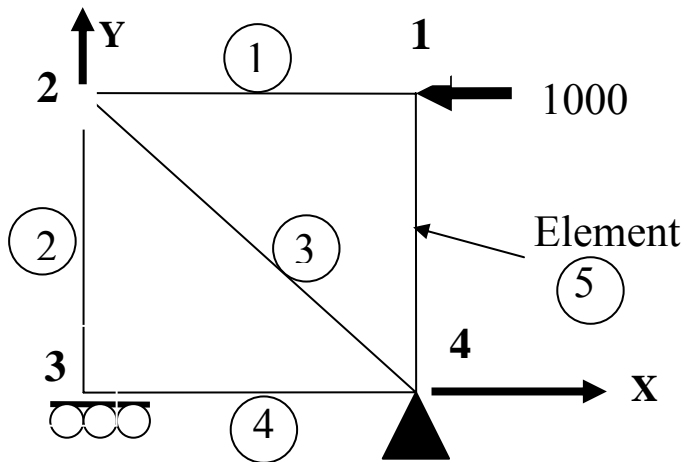


Figure 5 - Diagram of the truss showing the coordinate system.

We can now define this problem with three tables. The first will contain the nodal coordinates, the second will specify which two nodes are connected by each element, and the third will specify where the reactions are located and their direction. These tables are shown below.

Node Coordinates		
Node	X	Y
1	10	10
2	0	10
3	0	0
4	10	0

Table 1 - Coordinates of each node.

Element Nodal Connectivity		
Element	From Node	To Node
1	2	1
2	3	2
3	2	4
4	3	4
5	4	1

Table 2 - Table showing which nodes the elements connect.

Reactions at Nodes		
Reaction	Node	Direction
1	3	Y
2	4	Y
3	4	X

Table 3 - Table showing which node the reaction is applied and the direction of the reaction.

At this point we have defined the nodes, elements, and reactions. We can form the matrix defining the truss geometry. It will be a square matrix with 2 equations or

rows per node. The first row for each node will define the horizontal forces on that node and the second row will define the vertical forces. All of the forces will come from elements connecting to the node or reactions at the node. External forces will be stored in a separate array.

We can construct the matrix by first filling it with zeros. This is illustrated below. We fill it with zeros so that we can add the forces imposed by each element into the proper column. This initial matrix is shown below.

	F ₁	F ₂	F ₃	F ₄	F ₅	R ₁	R ₂	R ₃
1H	0	0	0	0	0	0	0	0
1V	0	0	0	0	0	0	0	0
2H	0	0	0	0	0	0	0	0
2V	0	0	0	0	0	0	0	0
3H	0	0	0	0	0	0	0	0
3V	0	0	0	0	0	0	0	0
4H	0	0	0	0	0	0	0	0
4V	0	0	0	0	0	0	0	0

Table 4 - The truss geometry matrix with all of the equation coefficients set to zero.

The first column lists the nodes. The H stands for horizontal forces and the V stands for vertical forces. The top row lists the forces and reactions. Each force is associated with an element. The zeros are the actual matrix.

Now we can go through all of the elements adding their contribution to this matrix. First we use the node and element tables (Tables 1 and 2) to compute the length of each element.

$$L_n = \sqrt{(x_{from} - x_{to})^2 + (y_{from} - y_{to})^2} \quad (2.5)$$

Where:

L_n = the length of element n

X_{from} = the x coordinate for the “from node” for element n

Y_{from} = the y coordinate for the “from node” for element n

X_{to} = the x coordinate for the “to node” for element n

Y_{to} = the y coordinate for the “to node” for element n

The horizontal and vertical components for the “from node” can be computed with:

$$H_{from,element} = \frac{X_{to} - X_{from}}{L_n} \quad (2.6)$$

$$V_{from,element} = \frac{Y_{to} - Y_{from}}{L_n} \quad (2.7)$$

The horizontal and vertical components for the “to node” can be computed with:

$$H_{to,element} = \frac{X_{from} - X_{to}}{L_n} \quad (2.8)$$

$$V_{to,element} = \frac{Y_{from} - Y_{to}}{L_n} \quad (2.9)$$

We continue applying these formulas to all of the elements. The values of H and V are added into each element’s column in the matrix shown in Table 4.

Next we add in the reaction forces. In each reaction column, add in a value of the reaction. Since the reactions are purely in the X and Y direction, all values will be either 1 or 0. For example, reaction 1 is applied at node 3 and it only works in the Y or vertical direction. We will place a 0 in the 3H row, R_1 column since there is not horizontal component for this reaction and a 1 in the 3V, R_1 column since there is a vertical reaction. We continue this process for all of the reactions. This completes the geometry matrix and we move on to the external forces vector.

The external forces vector is the vector on the right hand side of the equation. It defines the horizontal and vertical forces applied at each node. Only external forces are considered. Reactions and the forces caused by elements are handled by the matrix on the left hand side of the equation. If two different forces are applied to a node, the horizontal and vertical components are added and the values are placed in this vector.

Once this vector is in place, you can solve the set of equations for the forces in the elements and the reaction forces as was demonstrated earlier in the numerical example.

2.3 Writing a General Program

Most programs except for the simplest are too complex for us to keep the entire program and all of its logical interactions in our mind at one time. Since we can keep the entire program in our minds at one time, the entire writing process becomes a frustrating and time consuming affair with many false starts and failed attempts. The problem is not that you cannot write a program. The problem is that you cannot keep the entire design in your mind long enough to write the program.

A simple way to overcome this problem is to break the program into parts and treat each part as a smaller program. This process is called hierarchical decomposition. It is a divide and conquer strategy that is applicable to computer programs as well as engineering designs. If the problem is too large, you cannot get your mind around it, and if you cannot get your mind around it, you cannot solve the problem.

When you break the program into parts, try to create **no more** than 6 or 7 parts. Experiments have shown that most people cannot keep more than 6 or 7 things in mind at one time. You want a simple structure – one that you can easily keep in mind. If you divide the program into too many parts, you will not be able to keep them all in your mind at one time and will not be able to easily write the program.

We can illustrate this approach with a program to compute the stresses in a static truss. Assume there is a data file that defines the truss. The structure becomes:

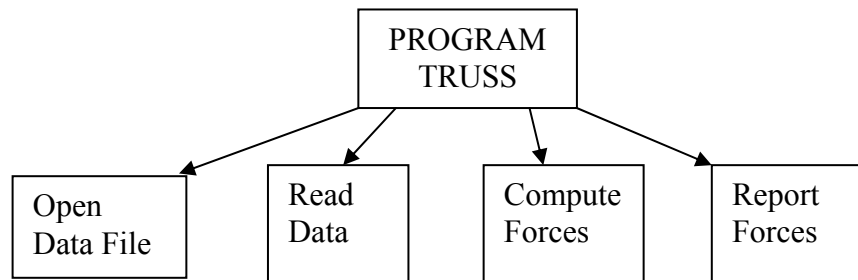


Figure 6 - Initial hierarchical decomposition of the Truss program.

Here, we have taken the Truss program and divided it into 4 smaller modules. We can think of each of these modules as smaller, simpler programs. Some of these modules may be more complex than others but that is not important. We now look at each of the modules and ask the question: “Do I understand all of the inputs and outputs for each of these modules and the logical interactions that must take place inside each module?” If the answer is yes, we are finished with the definition of that module. If the answer is no, we repeat the process for module dividing it into simpler parts.

The “Open Data File” is very simple and can be written in one line of code but the other modules are more complex. We will divide each of them.

The “Read Data” module can be divided into four smaller modules.

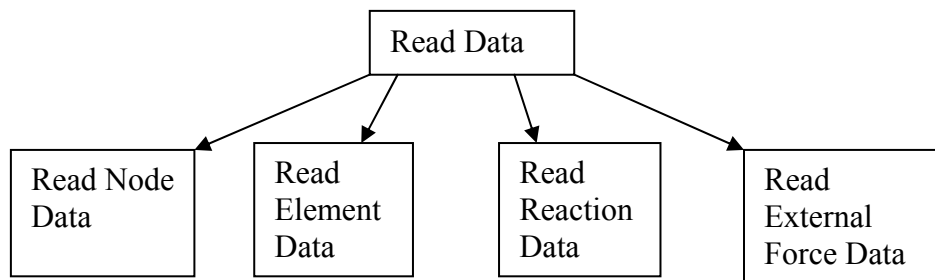


Figure 7 - Decomposition of the "Read data" module.

If we let the “Read data” routines read in the data and place this data into Matlab data structures, then the “Compute Forces” module is a very simple one line program. There is no need to divide it into smaller parts.

The “Report Forces” module can be divided into two parts as shown below.

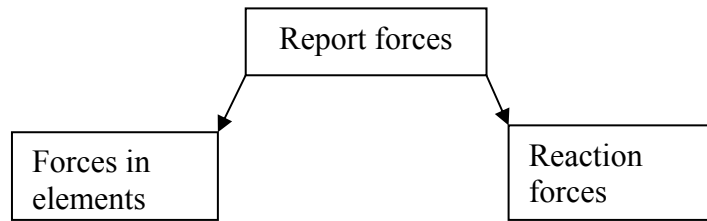


Figure 7 - Decomposition of the "Report forces" module.

Now that we have divided each of the module into smaller modules, we then repeat the process asking the question: “Do I understand all of the inputs and outputs for each of module and the logical interactions that must take place inside each module?” If the answer is no, we must repeat the process to create simpler lower level modules. Eventually, we will get to a place where all of the lowest level modules are very simple and can be easily understood. I think we have reached that point with the Truss program.

Program Construction

The next step is to write the program using the hierarchical diagram we have created. Write the code for one of the boxes at a time starting at the top of the hierarchical structure starting with a function statement marking the first line of code and a comment marking the end of the program.

```

function truss (file_name)
    .
    .
    .
% End of program
  
```

Most programmers find it very useful to put in comments defining each of the boxes in the hierarchical decomposition. There would be a comment saying “Open Data File” followed by “Read Data”, “Compute Forces”, and “Report Forces”. Go through the entire program laying out the structure. These comments will help you keep the structure in mind as you write the program.

Next, write the “Open Data File” code. This is a simple one. It consists of only one line.

One of the major mistakes people make when writing a program or creating an engineering design is to write too many lines without testing what they have done. As soon as you have written the “Open Data File” code, run the program and test it. The *file id* returned by the *fopen* statement should be positive number. If you are getting a -1 value, the file is not being opened. Did you misspell a variable name or did you make a mistake with the file name when you ran the program? Fix this problem before continuing with the development.

It is very important to test the program every few lines of code. Write a few lines of code then test it. If there is a mistake, you do not have to look through mountains of

code to find it. It is more than likely in the last few lines of code you have written. Everything else has already been tested.

Continue writing a few lines of code and testing it until you have completed the program.

2.4 The completed Program

The complete Matlab program for computing the forces in trusses is shown below. It follows the diagrams shown above and was written with the methodology discussed. Using this methodology, the program only took an hour or so to write.

```
% truss - this program uses a matrix method to solve for the forces in
%         the members of a statically determinate truss. It also
%         computes the reaction forces.
%
%         The program is started by typing
%
%         truss('input_data_file.txt')
%
%         at the MATLAB prompt. In this case the input data file is
%         named input_data_file.txt Any file name can be used. The
%         file is a common text file and can be created with NOTEPAD
%
%         Written by: Robert Greenlee

function truss(file_name);

% OPEN DATA FILE

fid = fopen(file_name, 'r');

% READ DATA - READ NODE DATA - First read the number of nodes in the
truss

Number_nodes=fscanf(fid, '%d', 1);

% Read the coordinates for each node

for i=1:Number_nodes
    Node=fscanf(fid, '%d', 1);
    Coordinate(Node, 1)=fscanf(fid, '%g', 1);
    Coordinate(Node, 2)=fscanf(fid, '%g', 1);
end

% READ DATA - READ ELEMENT DATA - Now read the number of elements and
the
%         element definition

Number_elements=fscanf(fid, '%d', 1);
M=zeros(2*Number_nodes, 2*Number_nodes);
```

```

for i=1:Number_elements
    Element=fscanf(fid, '%d', 1);
    Node_from=fscanf(fid, '%d', 1);
    Node_to=fscanf(fid, '%d', 1);

    dx=Coordinate(Node_to,1)-Coordinate(Node_from,1);
    dy=Coordinate(Node_to,2)-Coordinate(Node_from,2);
    Length=sqrt(dx^2 + dy^2);

    M(2*Node_from-1,Element) = dx/Length;
    M(2*Node_to-1, Element) = -dx/Length;
    M(2*Node_from, Element) = dy/Length;
    M(2*Node_to, Element) = -dy/Length;
end

% READ DATA - READ REACTION DATA - Now read in the reactions

Number_reactions = fscanf(fid, '%d', 1);

if (2*Number_nodes ~= (Number_elements + Number_reactions))
    error('Invalid number of nodes, elements, and reactions');
end

for i=1:Number_reactions;
    Reaction = fscanf(fid, '%d', 1);
    Node=fscanf(fid, '%d', 1);
    Direction = fscanf(fid, '%s', 1);
    if ((Direction == 'y') || (Direction == 'Y'))
        M(2*Node, Number_elements+Reaction)=M(2*Node,
Number_elements+Reaction)+1;
    elseif ((Direction == 'x') || (Direction == 'X'))
        M(2*Node-1, Number_elements+Reaction)=M(2*Node-1,
Number_elements+Reaction)+1;
    else
        error('Invalid direction for reaction')
    end
end

% READ DATA - READ EXTERNAL FORCE DATA - Now read in the external
forces

External=zeros(2*Number_nodes,1);
Number_forces=fscanf(fid, '%d', 1);

for i=1:Number_forces
    Node =fscanf(fid, '%d', 1);
    Force=fscanf(fid, '%g', 1);
    Direction = fscanf(fid, '%g', 1);
    External(2*Node-1)=External(2*Node-1)-Force*cos(Direction*(pi/180));
    External(2*Node) =External(2*Node) -Force*sin(Direction*(pi/180));
end

External

% COMPUTE FORCES - Solve the system of equations

```

```

A=M\External;

% REPORT FORCES - FORCES IN THE ELEMENTS

for i = 1:Number_elements
    fprintf('Element %d = %g \n', i, A(i))
end

% REPORT FORCES - REACTION FORCES

for i = 1:Number_reactions
    fprintf('Reaction %d = %g \n', i, A(Number_elements + i))
end

end          % End of program

```

2.4 The Input Data File

The input data file is shown below. The format is very general and almost any type of truss can be defined with this file.

```

4          % The number of nodes in the truss
1  10  10  % Node number, X coordinate, Y coordinate
2   0  10  %   for each node
3   0   0
4  10   0
5          % The number of elements in the truss
1  1  2    % Element number, Node from, Node to
2  2  3    %   for each element
3  2  4
4  3  4
5  4  1
3          % The number of constraints in the truss
1  3  Y    % Constraint number, Node, direction
2  4  Y    %   for each constraint
3  4  X
1          % The number of external forces
1  1000 180 % Node number, Force magnitude, Force Direction
          %   for each external force

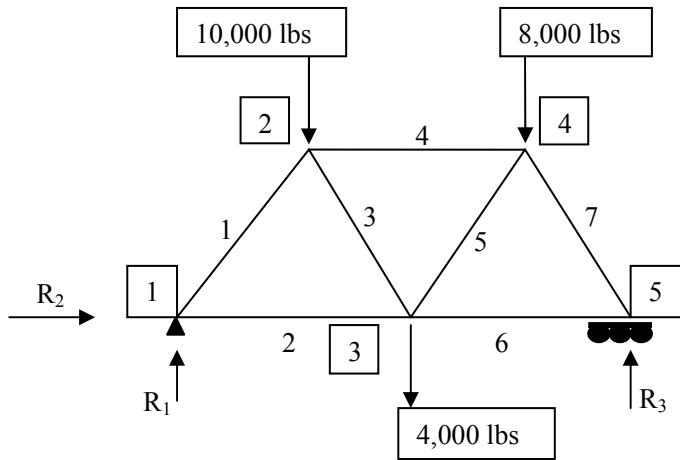
```

You can create an input file for each truss you want to analyze using NOTEPAD. The TRUSS program is run using

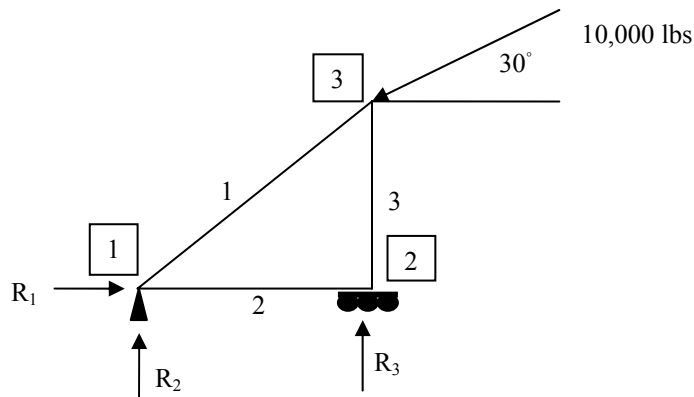
```
truss('data.txt')
```

where “data.txt” is the name of the data file defining the problem you are analyzing.

PROBLEMS



1. In the problem above, how many equations will be generated?
2. In the problem above, how many unknowns are there?
3. In the problem above, each element is 10 feet long. Construct the matrix you would solve to find the forces in the elements and the reactions. Use the element and node numbering shown.



4. In the problem above, solve for the forces in the elements and the reactions using a manual technique. Use the node and element numbering shown. Element 1 is 5 feet long, element 2 is 4 feet long, and element 3 is 3 feet long. You can use either Matlab or your calculator to solve the system of equations. Show all other work.

5. Modify the attached Matlab program so that Young's modulus and the cross sectional area of each element is defined in the input file. Compute the stresses then use the area and Young's modulus to print a table showing

Element Number	From Node	To Node	Force	Stress	Length Change
---------------------------	----------------------	--------------------	--------------	---------------	--------------------------

Use the problem drawn in Problem 1 and assume each element is 10 feet long to demonstrate that your program works. Assume each element has a cross sectional area of 2 and Young's modulus of 30,000,000.