

6.3 Solving trusses on a computer

The method of joints is routine and is easily implemented on a computer one way or another.

- First, some software packages will accept a collection of algebraic equations, say the joint equilibrium equations, and solve them for the unknowns.
- Second, one can take the set of algebraic equations as written by hand, and organize them into matrix form and solve that form on a computer as described in Section 1.5 (see page 119).
- Finally, one can treat the whole truss problem as one for which you want to do all the algebra and solution on the computer.

The first two approaches are general purpose, using the linearity of the equations and nothing special about trusses. They are as useful for trusses as for any other situation in which you have several simultaneous equations to solve.

Here we take the third approach. We will set up and solve the equations for a truss using no hand-calculations whatsoever. We describe a method which you can program in whatever is your preferred computer package. The advantages of having a general-purpose computer program available include:

- you can quickly solve any truss
- you are less likely to make an error
- if you find an error in data entry, you can quickly correct it without having to redo all other data entry and calculation
- you can change the truss geometry easily to see the effect on the bar tensions and reactions
- you can just as easily solve non-simple trusses in which neither the method of joints nor the method of sections gives you equations that you can solve one at a time.

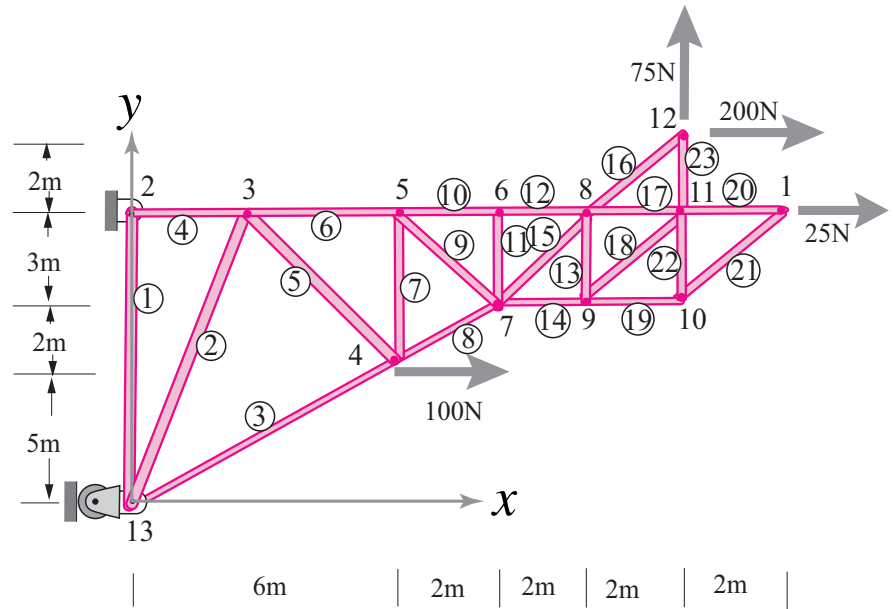
The act of understanding and writing such a program should also give you a better sense of the overall meaning of the method of joints.

The rest of this section is a description of the recipe, a presentation of the final program (on page 9), and some samples using that program. This program is just a systematic application of the method of joints.

The data that defines a truss problem

We first show how to define the truss, how it is supported, and the loads on it, with an organized collection of numbers rather than a picture. For definiteness, refer to the picture in fig. 6.41 which we want to communi-

cate to a computer.



① If you use and are comfortable with object-oriented programming some of the data structures below can be written in a more transparent form using suggestive naming rather than array locations for the various bits of data.

$$J = \begin{matrix} & \begin{matrix} \text{joint \#} \\ m \end{matrix} & \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ \vdots \\ 6 \\ \vdots \\ 13 \end{matrix} & \begin{bmatrix} 14 & 10 \\ 0 & 10 \\ 2 & 10 \\ 6 & 5 \\ \vdots & \vdots \\ 8 & 10 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \end{matrix}$$

X & Y coordinates

Figure 6.42: The matrix $[J]$ gives the location of the joints.

$$B = \begin{bmatrix} 1 & 13 & 2 \\ 2 & 13 & 3 \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ 11 & 7 & 6 \\ \dots & \dots & \dots \\ 23 & 11 & 12 \end{bmatrix}$$

bar no. base and tip joint no. S

Figure 6.43: The matrix $[B]$ gives the base and tip joint # of each bar.

Figure 6.41: A truss with 23 bars (numbered with circles around them), 13 joints, 4 loads and 3 reaction components.

First pick an origin, coordinate directions, units to use for length and units to use for force. First a few numbers that say how many other numbers are needed.

n_{joints} is the number of joints, often called j . In the example $n_{joints} = 13$.

n_{bars} is the number of bars (rods), often called b . In the example $n_{bars} = 23$.

n_{bcs} is the number of reaction components (or boundary conditions), often called r . n_{bcs} is commonly 3: an x and y component at one joint and just an x or y component at another, as in the example.

The descriptions of the joints, the bars, the reactions and the loads are held in 4 matrices^①.

$[J]$ is a matrix defining the joints. Each joint is identified by a number (1 or 2 or ...) with each number from 1 to n_{joints} associated with one joint. It doesn't matter which joint has which number. Each row of $[J]$ is the information for one joint. The first entry of a row is the joint number, and the next two numbers are the coordinates of the joint. If joint 6 is at $x = 8$ m, $y = 10$ m then the row 6 of $[J]$ would be $[6 \ 8 \ 10]$. $[J]$ has n_{joints} rows and 3 columns (fig. 6.42).

$[B]$ is a matrix defining the bars. It has one row for each bar (n_{bars} of them) and three columns. The bars are identified by numbers 1, 2,

...(sometimes circled, to distinguish them from the joint numbering). It doesn't matter which bar has which number so long as every integer from 1 to n_{bars} is associated with a bar (see fig. 6.43).

The first row of $[B]$ describes bar 1, the second describes bar 2, etc. The first element of each row is the bar number. This is also the number of the row, but it makes your data easier to read. The second two numbers are the numbers of the joints at the two ends of the bar. So if bar 11 connects base joint 7 with tip joint 6 the 11th row of $[B]$ is $[11 \ 7 \ 6]$. It is equivalent and ok to have instead the 11th row of $[B]$ be $[11 \ 6 \ 7]$; neither end of a bar is more special than the other. But once you have set the base and tip they are used to define angles in the calculations below.

$[R]$ is a matrix of reactions. It has as many rows as there are unknown reaction components, typically 3. $[R]$ has 4 columns. For easier reading, the first element of each row is the number of the row. The second element is the node at which the reaction applies. The next two numbers indicate the direction of the force acting on the truss (x and y components of a unit vector in the direction of the reaction):

- for a roller at a joint the last two numbers in the row are in the direction normal to the rollers. For normal support rollers they would be $[0 \ 1]$, for rollers against a vertical wall to the right of the structure they would be $[-1 \ 0]$. For a roller on a 45° slope the two components could be $[0.707 \ 0.707]$
- for a pin joint there are two rows in $[R]$: one for the x direction and one for the y .

Often $[R]$ will have exactly 3 rows. For the example matrix $[R]$ would be

$$[R] = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 2 & 0 & 1 \\ 3 & 13 & 1 & 0 \end{bmatrix}$$

$[F]$ is a matrix of applied loads. It has a row for each joint at which there is a non-zero load. It has three columns. The first entry of each row is the joint to which the load is applied. The next two numbers are the x and y components of the load applied to that joint. Any units can be used, they just have to be the same units for all loads. And the numerical answer for the tensions will be in these same units. If there is a rightwards load of 100 N at joint 4 one line of $[F]$ will read $[4 \ 100 \ 0]$ (see fig. 6.44).

$$F = \begin{bmatrix} 4 & 100 & 0 \\ 1 & 0 & -50 \\ 12 & 200 & 75 \end{bmatrix}$$

Joint no. x and y components
of applied load

Figure 6.44: The matrix $[F]$ defines the applied loads, one row for each joint at which a load is applied.

All the information about a truss that we usually communicate with a sketch is in the 4 matrices $[J]$, $[B]$, $[R]$, and $[F]$.

These specify the locations of the joints, which joints the bars are connected to, the directions and locations of reaction forces and the applied

loads. Given these matrices and nothing else one could draw the truss, supports, and loading.

The unknowns

Solving the truss, finding the tensions in the bars and reaction components, is just a matter of manipulating the numbers in the four data matrices. We will hold that answer in the list $[T]$:

$[T]$ is a column vector holding the unknowns. It has as many elements as there are unknowns ($n_u \equiv n_{bars} + n_{bcs}$). The first n_{bars} elements are the unknown tensions, the last n_{bcs} elements are the unknown reaction components.

The problem

Our goal now is to use the data matrices $[J]$, $[B]$, $[R]$, and $[F]$ to find the unknowns $[T]$. We know it can be done by hand and, because the equations are linear, computer solutions should be straightforward.

Setting up the joint equations in matrix form

We now apply the method of joints.

For each joint we draw a free-body diagram (in our mind). And we apply force balance in the x and y directions. Thus we will have $2n_{joints}$ equations in terms of our $n_u \equiv n_{bars} + n_{bcs}$ unknowns. The strategy is to write all these equations long hand (in our mind) and then assemble those into matrix form.

If joint 1 has emanating from it bars 20 and 21 and also has a 25 N horizontal load to the right the first of these $2n_{joints}$ equations is (see fig. 6.45):

$$\cos \theta_{20} T_{20} + \cos \theta_{21} T_{21} + 25 \text{ N} = 0,$$

where θ_{20} and θ_{21} are the angles of bars 20 & 21, measured CCW from the plus x direction. We can write this again as

$$0 * T_1 + 0 * T_2 + \cdots + A_{1,20} * T_{20} + A_{1,21} * T_{21} = -25 \text{ N}$$

where the cosines have been rewritten as elements of a matrix. If we assume that lots of these matrix elements are zero we can rewrite the first equation once again as

$$A_{1,1} * T_1 + A_{1,2} * T_2 + A_{1,3} * T_3 + \cdots + A_{1,n_u} * T_{n_u} = -F_{11}.$$

using $[A]$ as a matrix with lots of zeros, but sines and cosines of bar angles where appropriate. Recall that n_u is the number of unknown bar tensions and reaction components and $F_{11} = 25 \text{ N}$ is the x component of the load applied to joint 1.

For the second equation we similarly write the equation for force balance in the y direction for joint 1.

$$\sin \theta_{20} T_{20} + \sin \theta_{21} T_{21} = 0,$$

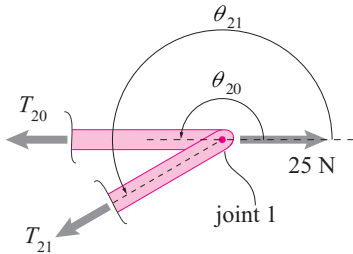


Figure 6.45: Free-body diagram of joint 1 on the truss of fig. 6.41 on page 2.

which can also be written out with the terms of $[A]$ (see fig. 6.46) as

$$A_{21} * T_1 + A_{22} * T_2 + A_{23} * T_3 + \cdots + A_{2n_u} * T_{n_u} = -F_{12}.$$

The next two equations describe joint 2, etc. Thus the assembly of $2n_{joints}$ equations looks like this

$$\begin{array}{ccccccc} A_{11} * T_1 & A_{12} * T_2 & \cdots & = & -F_{11} \\ A_{21} * T_1 & A_{22} * T_2 & \cdots & = & -F_{12} \\ \cdots & & & & \\ \cdots & & & & \\ A_{2n_{joints} 2} T_1 & A_{2n_{joints} 2} T_2 & \cdots & = & -F_{n_{joints} 2} \end{array}$$

which we can write more compactly as

$$[A][T] = [L] \quad (6.19)$$

where $[A]$ is a matrix with cosines and sines of the bar angles and lots of zeros (because most bars don't touch a given joint) and $[L]$ is a list of negative of the loads applied in the x and y directions at the joints.

The point is, that all the information needed to calculate all the terms in $[A]$ and $[L]$ is in our four truss-definition matrices $[J]$, $[B]$, $[R]$ and $[F]$. And eqn. (6.19) for the unknown $[T]$ is exactly of the type that computers are great at solving.

Some preliminary geometry

The matrix $[A]$ is made up of sines and cosines of bar angles and we have specified the truss by the x and y positions of the ends of the bars. We first tell the computer to do some simple trig to find the sines and cosines.

$[X]$ is a list of x coordinates of each bar tip relative to its base. $[X]$ is a single column with n_{bars} entries. To find the entries of $[X]$ subtract the base-joint x coordinate from the tip-joint x coordinate. For bar 13 this would be

$$X(13) = J(B(13,3), 2) - J(B(13,2), 2)$$

because $B(13,3)$ is the joint at the tip of bar 13 and $B(13,2)$ is the joint at the base. Thus $J(B(13,3), 2)$ and $J(B(13,2), 2)$ are the x coordinates of the joints at the tip and base of bar 13. To find all of the elements of $[X]$ you may need to loop through all the bars or, depending on your package, you may be able to do the subtraction in one step.

$[Y]$ is a list of base-to-tip y coordinates for the bars defined analogously to $[X]$ above. Thus

$$Y(13) = J(B(13,3), 3) - J(B(13,2), 3)$$

$[D]$ is a list of bar lengths (distances), so

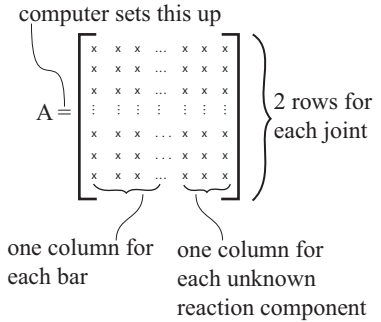


Figure 6.46: The matrix $[A]$ contains all the coefficients in the linear joint equations. It is made of sines and cosines of bar angles and the direction cosines of reaction-force directions.

$$D(13) = (X(13)^2 + Y(13)^2)^{.5}$$

[C] is a list of n_{bars} cosines for the bars, one cosine for each bar. It is defined as the counter-clockwise angle of the base-to-tip bar relative to the positive x axis. Thus

$$C(13) = X(13)/D(13) \quad \% \text{ cosine}$$

[S] is a similar list of sines so

$$S(13) = Y(13)/D(13) \quad \% \text{ sine}$$

② The calculation of [X], [Y] and [D] are just intermediate steps to simplify the presentation. If you can tolerate dense coding and use a package that deals well with matrices, [C] and [S] can be generated with as few as 2 dense lines of code.

All we need from the above are the [C] and [S] column vectors^②.

Building up [A] from [J], [B] and [R]

The only difficult work in setting up a statically-determinate truss for computer solution is making up the matrix [A]. First let's set [A] to be a matrix with $2n_{joints}$ rows and n_u columns and with every entry zero.

$$A = [0]$$

We now need to put a bunch of cosines and sines into the right places.

Cycle through the bars and put in cosines and sines of bar angles.

If we look at the whole [A] matrix we see that the information about bar 7, say, only occurs in column 7 of [A]; column 7 of [A] consists of the terms that multiply T_7 . Furthermore, information about bar 7 only shows up in the rows corresponding to the x and y force balance for the joints at its two ends; that's 4 places in total.

③ **Naive approaches.** One could imagine working one row at a time, corresponding to working one joint equation at a time rather than one bar at a time. For each joint we then would need to hunt through the list of bars and see which are connected to that joint. One could write a program to do this, it's just more complex than the approach we present. Alternately, you might imagine that in our original data set we would have associated each joint with the bars that connect to it (rather than the other way around as we did). This is also legitimate. But, because the number of connected bars varies from joint to joint the data structure would be more complex. Finally, because the key information is the location of the bar ends, we could have used those coordinates in our data array for the bars. But this would have required our entering the coordinates of each joint over and over, once for each bar-end connected to that joint.

- Bar 7 pulls on its base joint B(7, 2) in the x direction. Because we write 2 equations for each joint this equation corresponds to row $2*B(7, 2)-1$. Thus we can make the assignment

$$A((2*B(7, 2)-1), 7) = C(7)$$

- Bar 7 pulls on its base joint in the y direction. This equation corresponds to the next row $2 * B(7, 2)$ Thus we can make the assignment

$$A((2*B(7, 2)), 7) = S(7)$$

- Bar 7 pulls in the opposite direction on its tip joint B(7, 3) so

$$A(2*B(7, 3) -1, 7) = -C(7)$$

- and

$$A(2*B(7, 3) , 7) = -S(7)$$

One needs to cycle through all the bars^③ and make these 4 assignments, 7 was just used as an example. In a package that deals well with matrices all four assignments associated with one bar could be in a single line of code.

Cycling through the reactions to fill in the right-most columns of [A]. The unknown reaction components have much the same role as do the bar tensions. But they act on only one joint. Thus each reaction component only affects 2 rows of [A], the x and y components of that joint equation.

For reaction 3, say, the relevant joint is $R(3, 2)$ and thus the relevant rows are $2 \cdot R(3, 2) - 1$ and $2 \cdot R(3, 2)$. The relevant column is $n_{bars} + 3$.

- for the x component of reaction 3

$$A((2 \cdot R(3, 2) - 1), (n_{bars} + 3)) = R(3, 3)$$

- for the y component of reaction 3

$$A((2 \cdot R(3, 2)), (n_{bars} + 3)) = R(3, 4)$$

Most often, for trusses that are rigid even when floating, one only has three such reaction components to cycle through.

The load vector [L] The load vector is just made up of the forces applied to the joints. For load 2, for example, applied at joint $F(2, 1)$, the two relevant rows of [L] are $2 \cdot F(2, 1)$ and $2 \cdot F(2, 1) - 1$ at which act the x , and y components of the force $F(6, 2)$ and $F(6, 3)$, respectively. Thus, for load 6, we have

$$L(2 \cdot F(2, 1) - 1) = -F(2, 2)$$

$$L(2 \cdot F(2, 1)) = -F(2, 3)$$

Recall that the minus sign follows from moving the applied load to the right side of the equation. This pair of commands needs to be applied to each line of the [F] matrix.

Solution

We have now constructed all the unknowns in eqn. (6.19)

$$[A][T] = [L] \quad (6.20)$$

and can thus hand the problem to the computer for solution

Solve $\{A T = L\}$ for T

The resulting column vector [T] is a list of bar tensions and reaction components.

The complete truss program

The complete truss program, in pseudo-code that you need to convert to your preferred computer language/package, is shown in fig. 6.41 on page 9. Some of the loops can be ‘vectorized’ if your package supports such. The output [T] is a column with the tensions followed by the reaction components.

What can go wrong?

Besides the various careless errors you will discover the first 10 or so times you try to run your code, there are possible deeper problems.

Because we are not trying to write general purpose super-robust software we assume the simple check for determinacy (number of unknowns = number of equations):

$$n_{rods} + n_{bcs} = 2n_{joints} \quad \text{or} \quad b + r = 2j$$

has been satisfied. Thus $[A]$ will be square. If the truss is determinate the computer will give you a nice solution. If the truss is not determinate, with $[A]$ square or not, the result of the computer calculation will depend on the software package, ranging from an error message (e.g., “Matrix singular!” or “Divide by zero!”) to the computer’s making its best guess at what you want (even though the equations may have no solution, or there may be many solutions to select from). Some computer packages don’t tell you when they are guessing.

How the pros solve trusses

The algorithm here is one way to set up and solve statically-determinate mechanics problems on a computer. In detail, however, this recipe is simpler than that commonly used in the *finite-element* method. Finite-element programs can also solve statically-indeterminate problems. A statically-indeterminate truss has tensions which can’t be found from statics alone, but which can be found if the bar stiffnesses are known. Finite-element programs don’t assume the bars are rigid. Rather, they take account of the small deformations of the bars.

A simple finite-element program for statically-indeterminate trusses would not use the tensions in the bars as unknowns, but rather the displacements of the joints. Such a program would be a little longer than the one presented here, and also requires introduction of a ‘stiffness matrix’^④, a topic a shade too advanced to cover in detail here.

④ **Stiffness matrix.** The stiffness matrix $[K]$ for a truss has $2n_{joints}$ rows and columns. It satisfies the equation $[\Delta] = [K][L]$ where $[L]$ is a list of x and y components of the loads applied to all the joints and $[\Delta]$ are the x and y displacements of the joints for those loads. The matrix $[K]$ can be *assembled* if the properties of all the bars are given.


```

%PSEUDO-CODE TO SOLVE ANY 2D STATICALLY DETERMINATE TRUSS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assign values to the matrices which define the truss and loading
J = [ 1 . . . ] % specify the joint locations
B = [ 1 . . . ] % specify the joints that the bars connect
R = [ 1 . . . ] % specify which nodes connect to the ground and how
F = [ . . . . ] % specify which nodes have what applied loads
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Program TRUSS, input is (J,B,R,F) output is (T)
% Set up
A = a square matrix of zeros with twice as many rows as J
L = a column of zeros with twice as many rows as J
nbars = the number of rows of B
% Fill in the columns of the matrix A associated with bar tensions
Loop for every bar (each row i of B)
    base = B(i,2) % joint at one end of a bar
    tip = B(i,3) % joint at the other end
    X = J( tip, 2 ) - J( base, 2 ) % base to tip x shadow of bar
    Y = J( tip, 3 ) - J( base, 3 ) % base to tip y shadow of bar
    D = ( X^2 + y^2 )^.5 % length of bar
    C = X/D % cosine of bar angle
    S = Y/D % sine of bar angle
    A( (2*base-1), i ) = C % x comp of pull direction on base
    A( (2*base ), i ) = S % y comp of pull direction on base
    A( (2*tip -1), i ) = -C % x comp of pull direction on tip
    A( (2*tip ), i ) = -S % y comp of pull direction on tip
End Loop
% Fill in rightmost columns of A, associated with reaction forces
Loop for every reaction component (each row j of R)
    joint = R(j,2) % joint at ground connection
    A( (2*joint-1), (nbars+j) ) = R(j,3) % x comp of reaction direction
    A( (2*joint ), (nbars+j) ) = R(j,4) % y comp of reaction direction
End Loop
Loop for all joints with loads (each row k of F)
    joint = F(k,1) % joint at which load is applied
    L( 2*joint -1 ) = - F(k,2) % x component of load
    L( 2*joint ) = - F(k,3) % y component of load
End Loop

% Solve the truss (solve the set of simultaneous joint-equilibrium equations)
Solve {AT = L} for T % The whole calculation is done in this one line.
                    % T is a list of bar tensions
                    % followed by reaction components

End Program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Figure 6.47: **Pseudo-code for solving any statically determinate truss.** This ‘program’ calculates the bar tensions and reactions in a statically determinate truss. The algorithm is described in detail starting on page 1. This program could be reduced to 10 lines of code in some common computer languages (with some loss of clarity).

SAMPLE 6.10 For the truss shown in the figure, the coordinates of the three joints are: A(0,0), B(2m,2m), and C(4m,0). Find all reactions and bar forces using computer analysis. Show the input data to the program used and the matrices [A] and [L] generated by the program.

Solution The free-body diagram of the truss with the unknown reactions serially numbered is shown in fig. 6.49. We have also numbered the bars and joints for preparing the input data file as described in the text. Here, we have three bars and three joints, three unknown reactions, and one externally applied load. Therefore, the input matrices [B] for bar data, [J] for joint data, [R] for support reaction data, and [F] for applied load data are as follows (see page 341 for row and column descriptions).

$$B = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 2 & 3 \\ 3 & 1 & 3 \end{bmatrix}, \quad J = \begin{bmatrix} 1 & 0.0 & 0.0 \\ 2 & 2.0 & 2.0 \\ 3 & 4.0 & 0.0 \end{bmatrix},$$

$$R = \begin{bmatrix} 1 & 1 & 1.0 & 0.0 \\ 2 & 1 & 0.0 & 1.0 \\ 3 & 3 & 0.0 & 1.0 \end{bmatrix}, \quad F = \begin{bmatrix} 2 & 0.0 & -5.0 \end{bmatrix}$$

The computer program based on the pseudocode described in the text generates the following matrices [A] and [L], before solving for the tensions and reactions:

$$A = \begin{bmatrix} 0.7071 & 0 & 1.0000 & 1.0000 & 0 & 0 \\ 0.7071 & 0 & 0 & 0 & 1.0000 & 0 \\ -0.7071 & 0.7071 & 0 & 0 & 0 & 0 \\ -0.7071 & -0.7071 & 0 & 0 & 0 & 0 \\ 0 & -0.7071 & -1.0000 & 0 & 0 & 0 \\ 0 & 0.7071 & 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

$$L = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 5 \\ 0 \\ 0 \end{bmatrix}$$

The final step, Solve {A T = F} for T, gives the following output

$$T = \begin{bmatrix} -3.5355 \\ -3.5355 \\ 2.5000 \\ 0 \\ 2.5000 \\ 2.5000 \end{bmatrix}$$

which means, $T_1 = T_2 = -3.5355 \text{ N}$, $T_3 = 2.5 \text{ N}$, $R_1 = 0$, and $R_2 = R_3 = 2.5 \text{ N}$.

$$T_1 = T_2 = -3.5355 \text{ N}, T_3 = 2.5 \text{ N}, R_1 = 0, R_2 = R_3 = 2.5 \text{ N}$$

Note: If you write a truss code, you can use this sample to check your code.

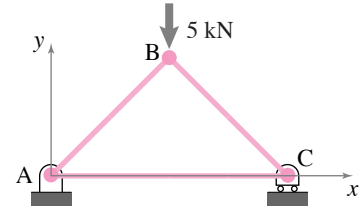


Figure 6.48

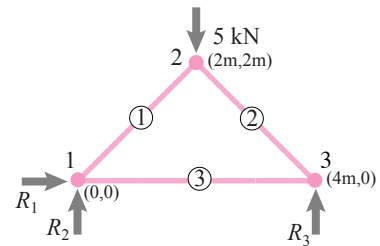


Figure 6.49

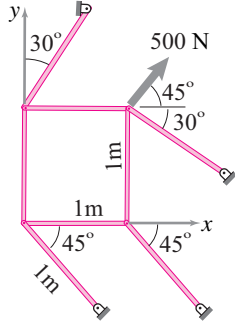


Figure 6.50

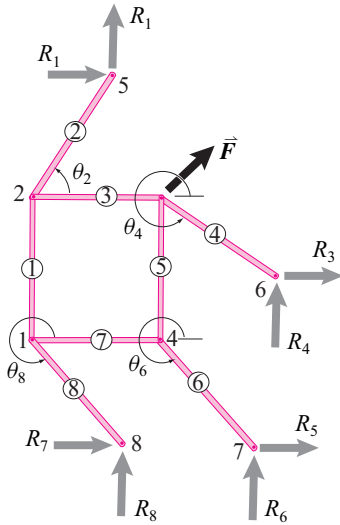


Figure 6.51: Free-body diagram of the truss along with bar and joint numbers. From the given geometry, $\theta_1 = \pi/3$, $\theta_2 = 11\pi/6 = -\pi/6$, and $\theta_3 = \theta_4 = 7\pi/4$. Assuming the bar length to be ℓ , we can write the coordinates of all joints in terms of ℓ and θ 's. For example, $x_5 = \ell \cos \theta_1$, $y_5 = \ell + \ell \sin \theta_1$.

SAMPLE 6.11 The truss shown in the figure has no triangles, yet it is rigid in the configuration shown as discussed in the text. It is also an example of a truss where you cannot find a sequence of joints that will let you solve for the bar forces ‘locally’, that is, without solving all joint equations simultaneously. Assume all bars to be 1 m long. Find all reactions and bar forces. Show the input data to the program used.

Solution The free-body diagram of the truss with the unknown reactions serially numbered is shown in fig. 6.51. Note that support reactions have been taken as unknown x and y components of the reaction at each support point. We could have, alternatively, taken the reaction components to be along and normal to the bars at each support point.

The bars and joints are numbered as shown. Here, we have eight bars and eight joints, eight unknown reactions, and one externally applied load. Let the length of each bar be $\ell = 1$ m. The angle of outer bars with the x -axis are $\theta_2 = \pi/3$, $\theta_4 = -\pi/6$, $\theta_6 = \theta_8 = -\pi/4$. Therefore, the input matrices $[B]$ (bar data), $[J]$ (joint data), $[R]$ (support reaction data), and $[F]$ (applied load data) are as follows (see page 341 for row and column descriptions).

$$B = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 2 & 5 \\ 3 & 2 & 3 \\ 4 & 3 & 6 \\ 5 & 3 & 4 \\ 6 & 4 & 7 \\ 7 & 1 & 4 \\ 8 & 1 & 8 \end{bmatrix}, \quad J = \begin{bmatrix} 1 & 0.0 & 0.0 \\ 2 & 0.0 & \ell \\ 3 & \ell & \ell \\ 4 & \ell & 0.0 \\ 5 & \ell \cos \theta_2 & \ell + \ell \sin \theta_2 \\ 6 & \ell + \ell \cos \theta_4 & \ell + \ell \sin \theta_4 \\ 7 & \ell + \ell \cos \theta_6 & \ell \sin \theta_6 \\ 8 & \ell \cos \theta_8 & \ell \sin \theta_8 \end{bmatrix},$$

$$R = \begin{bmatrix} 1 & 5 & 1.0 & 0.0 \\ 2 & 5 & 0.0 & 1.0 \\ 3 & 6 & 1.0 & 0.0 \\ 4 & 6 & 0.0 & 1.0 \\ 5 & 7 & 1.0 & 0.0 \\ 6 & 7 & 0.0 & 1.0 \\ 7 & 8 & 1.0 & 0.0 \\ 8 & 8 & 0.0 & 1.0 \end{bmatrix}, \quad F = \begin{bmatrix} 3 & 500/\sqrt{2} & 500/\sqrt{2} \end{bmatrix}$$

The computer program based on the pseudocode described in the text generates the following output, $[T]$, for the tensions and reactions: The final step, **Solve** $\{A \ T = F\}$ for T , gives the following result for bar tensions and reactions.

$$\begin{array}{ll} T_1 = -418.26 \text{ N}, & R_1 = -241.48 \text{ N}, \\ T_2 = -482.96 \text{ N}, & R_2 = -418.26 \text{ N}, \\ T_3 = 241.48 \text{ N}, & R_3 = -112.07 \text{ N}, \\ T_4 = -129.41 \text{ N}, & R_4 = 64.70 \text{ N}, \\ T_5 = 418.26 \text{ N}, & R_5 = 418.26 \text{ N}, \\ T_6 = 591.51 \text{ N}, & R_6 = -418.26 \text{ N}, \\ T_7 = 418.26 \text{ N}, & R_7 = -418.26 \text{ N}, \\ T_8 = -591.51 \text{ N}, & R_8 = 418.26 \text{ N}. \end{array}$$

Note: It is easy to check that $R_1 + R_3 + R_5 + R_7 = -F \cos(45^\circ)$ and $R_2 + R_4 + R_6 + R_8 = -F \sin(45^\circ)$ where $F = 500$ N. That is, for the free-body diagram of the truss, $\sum F_x = 0$ and $\sum F_y = 0$.

6.3 Solving trusses on computers

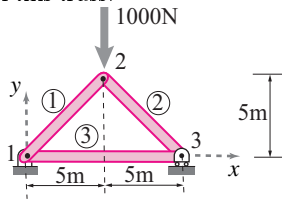
Preparatory Problems

You should check your mastery of the method of joints problems before working on this section.

6.3.1 Define these matrices and column vectors used to define a truss, the loading on it, the bar tensions, the reactions, and the coefficients in the matrix form of the joint equilibrium equations:

- $[J]$
- $[B]$
- $[R]$
- $[F]$
- $[T]$
- $[L]$
- $[A]$

6.3.2 By hand, with no use of a computer, find all of the matrices and column vectors above for this truss.



Problem 6.3.2

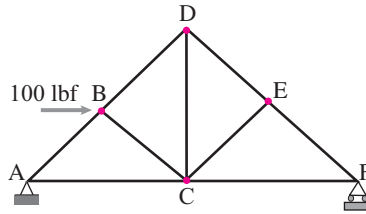
6.3.3 When does the numerical recipe presented here succeed and when does it fail? When it fails, how does it fail?

6.3.4

- Write a computer program, using your preferred language or package, that takes as input the matrices $[J]$, $[B]$, $[R]$, and $[F]$ and calculates $[T]$.
- Test this program on the truss of problem 6.3.2.

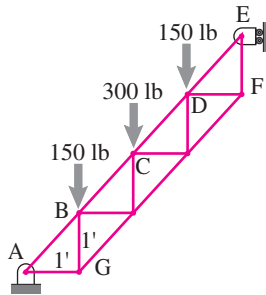
More-Involved Problems

6.3.5 All of the bars in the symmetric truss below are either level or at 30° from the horizontal. Find all the bar forces and reactions.



Problem 6.3.5

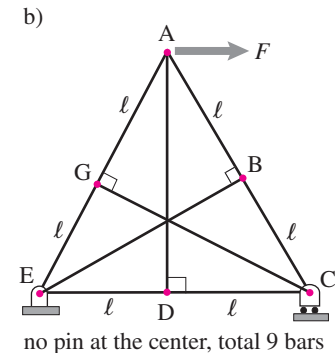
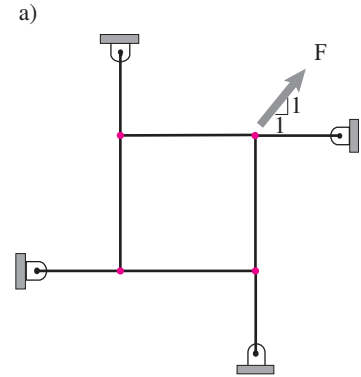
6.3.6 Find the force in each bar of the staircase truss shown in the figure by writing the required number of equilibrium equations and then solving them on a computer.



Problem 6.3.6

6.3.7 Find the tensions in all the bars, and all the reactions for these structures.

- A square supported by four bars. This is perhaps the simplest rigid structure that has no triangles.
- The 9-bar structure shown. This structure also has no triangles in that there is no closed circuit that involves only three bars (for example, from D to A to B to C and back to D involves 4 bars).



Problem 6.3.7

6.3.8 Analyze the truss given in Problem 6.1.20 and solve for all bar tensions and support reactions.

6.3.9 Solve Problem 6.1.21.

6.3.10 Solve Problem 6.1.22.

6.3.11 Using your program from problem 6.3.4 solve each of the following problems:

- Problem 6.1.10
- Problem 6.1.11
- Problem 6.1.13
- Problem 6.2.5
- Problem 6.2.7
- Problem 6.2.12

6.3.1 Define these matrices and column vectors used to define a truss, the loading on it, the bar tensions, the reactions, and the coefficients in the matrix form of the joint equilibrium equations:

- | | |
|----------|----------|
| a) $[J]$ | c) $[R]$ |
| b) $[B]$ | d) $[F]$ |
| | e) $[T]$ |
| | f) $[L]$ |
| | g) $[A]$ |

a) $[J]$ - contains Joint Information

Columns : Joint Number, x-coordinate, Y-coordinate

b) $[B]$ - contains Bar Information

Columns : Bar Number, Joint-1, Joint-2

c) $[R]$ - contains Reaction Forces Information

Columns : Reaction Force Number, Joint, X-component, Y-component

d) $[F]$ - contains External Force Information

Columns : Joint, X-component, Y-component

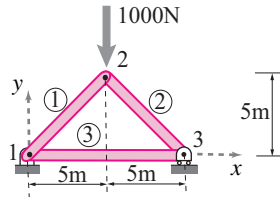
e) $[T]$ - contains Value of Tension on each Bar

Column : Tension

f) $[L]$ - contains List of negative of loads applied in x and y directions.

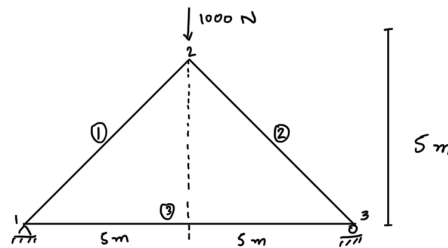
g) $[A]$ - contains cosines and sines of the bar angles.

6.3.2 By hand, with no use of a computer, find all of the matrices and column vectors above for this truss.



Problem 6.2

Sketch:



$$[S] = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 5 & 5 \\ 3 & 10 & 0 \end{bmatrix}$$

$$[B] = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 2 & 3 \\ 3 & 1 & 3 \end{bmatrix}$$

$$[R] = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 3 & 3 & 0 & 1 \end{bmatrix}$$

$$[f] = \begin{bmatrix} 2 & 0 & -1000 \end{bmatrix}$$

$$[A] = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 1 & 1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & -1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[L] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1000 \\ 0 \\ 0 \end{bmatrix}$$

$$[T] = [A]^{-1} [L]$$

$$[T] = \begin{bmatrix} 1000/\sqrt{2} \\ 1000/\sqrt{2} \\ -500 \\ 0 \\ -500 \\ -500 \end{bmatrix}$$

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

6.3.3 When does the numerical recipe presented here succeed and when does it fail? When it fails, how does it fail?

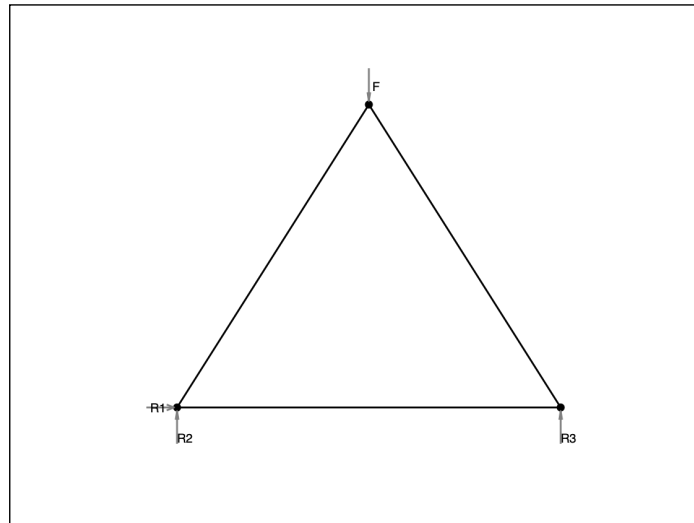
The numerical method of analysing trusses succeeds when $[A]$ is a square matrix, i.e. a determinate truss. The method fails when the truss is non-determinate, i.e. $[A]$ is a non-square matrix. In the case of a non-determinate truss, an error message is displayed by the program.

6.3.4

- a) Write a computer program, using your preferred language or package, that takes as input the matri-

ces $[J]$, $[B]$, $[R]$, and $[F]$ and calculates $[T]$.

- b) Test this program on the truss of problem 6.3.2.



Code:

a)

```

1 function T = TrussAnalyser(B,J,R,F)
2
3     % Receiving Matrix A from mat() %
4     A = TrussMatrix(B,J,R);
5     [sr,g] = size(A);
6
7     % Finding Matrix Containing External Forces %
8     L = zeros(sr,1);
9     [sf, g] = size(F);
10    for i = 1:sf
11        L((F(i)*2)-1,1) = F(i,2);
12        L((F(i)*2),1) = F(i,3);
13    end
14
15    % Finding Tensions on Each Rod Based on External Forces
16    %
17    T = A\L;
18
19    % Checking if values of Tensions are Valid %
20    if MatrixCheck(A,T,L) == 1

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.


```

20         disp("Values are Valid")
21     else
22         disp("Values are not Valid")
23     end
24 end

```

```

1  function A = TrussMatrix(B,J,R)
2      % Finding Number of Rows in Matrix A%
3      [num_reactions, ~] = size(R);
4      [sr,g] = size(B);
5      A = zeros(sr+num_reactions,sr+num_reactions);
6
7      % Finding Matrix A Based on Joint and Rod Data%
8      for i = 1:sr
9          % Finding Joints connecting to ith Rod %
10         jnt1 = B(i,2);
11         jnt2 = B(i,3);
12
13         % Finding Angle between ith Rod and Horizontal
14         theta = atan((J(jnt1,3)-J(jnt2,3))/(J(jnt1,2)-
J(jnt2,2)));
15         % Adding values of cos and sin theta into A Matrix %
16         A((jnt1*2)-1,i) = cos(theta);
17         A(jnt1*2,i) = sin(theta);
18         A((jnt2*2)-1,i) = -cos(theta);
19         A(jnt2*2,i) = -sin(theta);
20     end
21     for i = 1:num_reactions
22         if R(i,3) == 1
23             A((R(i,2)*2)-1,size(A)-num_reactions+i) = 1;
24         elseif R(i,4) == 1
25             A((R(i,2)*2),size(A)-num_reactions+i) = 1;
26         end
27     end
28 end

```

```

1  function a = MatrixCheck(A,T,L)
2      [sr, g] = size(A);
3      a = 1;

```

```

4     tol = 0.0001;      % Setting Tolerance Value
5     for i = 1:sr
6         tmp = A(i,:)*T;
7         disc = abs(tmp-L(i));
8
9         % If the Difference is Greater than the Tolerance %
10        if (disc >= tol)
11            disp(strcat("Not valid at Joint:",num2str(i)))
12            a = 0;
13        end
14    end
15 end

```

b)

```

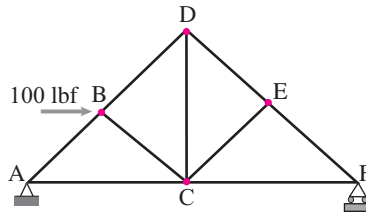
1  J = [1 0 0;
2       2 5 5;
3       3 10 0;];
4
5  B = [1 1 2;
6       2 2 3;
7       3 1 3;];
8
9  R = [1 1 1 0;
10      2 1 0 1;
11      3 3 0 1;];
12
13 F = [2 0 -1000];
14
15 T = TrussAnalyser(B,J,R,F)

```

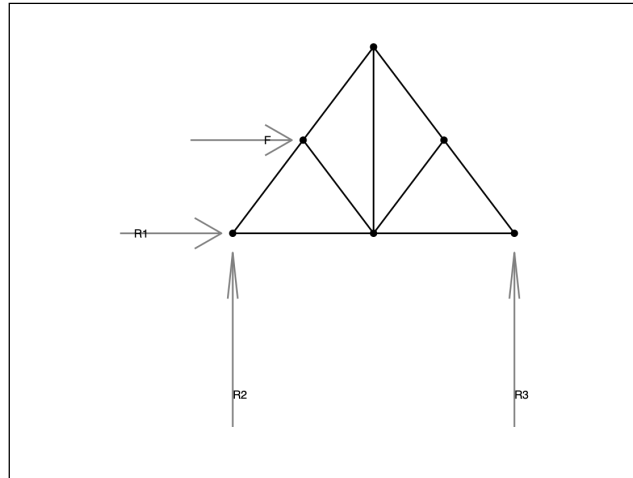
Output :

$$T = \begin{bmatrix} 707.1068 \\ 707.1068 \\ -500 \\ 0 \\ -500 \\ -500 \end{bmatrix}$$

6.3.5 All of the bars in the symmetric truss below are either level or at 30° from the horizontal. Find all the bar forces and reactions.



Problem 6.5



Code:

```

1  J = [1 0 0;
2      2 1/2 1/(2*sqrt(3));
3      3 1 0;
4      4 1 1/sqrt(3);
5      5 1.5 1/(2*sqrt(3));
6      6 2 0];
7
8  B = [1 1 2;
9      2 2 4;
10     3 4 5;
11     4 5 6;
12     5 6 3;
13     6 1 3;
14     7 2 3;
15     8 3 4;
16     9 3 5];
17
18 R = [1 1 1 0;
19     2 1 0 1;
20     3 6 0 1;];
21

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```

22 F = [2 100 0];
23
24 TrussAnalyser(B,J,R,F)

1  function T = TrussAnalyser(B,J,R,F)
2
3      % Receiving Matrix A from mat() %
4      A = TrussMatrix(B,J,R);
5      [sr,g] = size(A);
6
7      % Finding Matrix Containing External Forces %
8      L = zeros(sr,1);
9      [sf, g] = size(F);
10     for i = 1:sf
11         L((F(i)*2)-1,1) = F(i,2);
12         L((F(i)*2),1) = F(i,3);
13     end
14
15     % Finding Tensions on Each Rod Based on External Forces
16     %
17     T = A\L;
18
19     % Checking if values of Tensions are Valid %
20     if MatrixCheck(A,T,L) == 1
21         disp("Values are Valid")
22     else
23         disp("Values are not Valid")
24     end
25 end

1  function A = TrussMatrix(B,J,R)
2      % Finding Number of Rows in Matrix A%
3      [num_reactions, ~] = size(R);
4      [sr,g] = size(B);
5      A = zeros(sr+num_reactions,sr+num_reactions);
6
7      % Finding Matrix A Based on Joint and Rod Data%
8      for i = 1:sr
9          % Finding Joints connecting to ith Rod %
10         jnt1 = B(i,2);

```

```
11     jnt2 = B(i,3);
12
13     % Finding Angle between ith Rod and Horizontal
14     theta = atan((J(jnt1,3)-J(jnt2,3))/(J(jnt1,2)-
J(jnt2,2)));
15     % Adding values of cos and sin theta into A Matrix %
16     A((jnt1*2)-1,i) = cos(theta);
17     A(jnt1*2,i) = sin(theta);
18     A((jnt2*2)-1,i) = -cos(theta);
19     A(jnt2*2,i) = -sin(theta);
20 end
21 for i = 1:num_reactions
22     if R(i,3) == 1
23         A((R(i,2)*2)-1,size(A)-num_reactions+i) = 1;
24     elseif R(i,4) == 1
25         A((R(i,2)*2),size(A)-num_reactions+i) = 1;
26     end
27 end
28 end
```

```

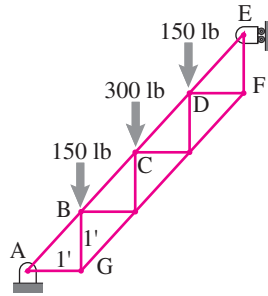
1 function a = MatrixCheck(A,T,L)
2     [sr, g] = size(A);
3     a = 1;
4     tol = 0.0001;    % Setting Tolerance Value
5     for i = 1:sr
6         tmp = A(i,:)*T;
7         disc = abs(tmp-L(i));
8
9         % If the Difference is Greater than the Tolerance %
10        if (disc >= tol)
11            disp(strcat("Not valid at Joint:",num2str(i)))
12            a = 0;
13        end
14    end
15 end

```

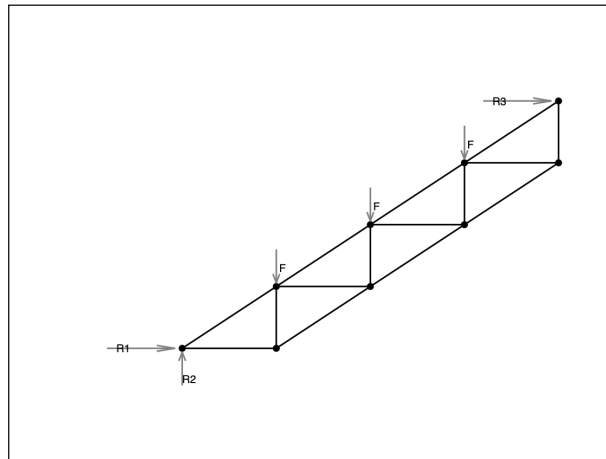
Output:

$$T = \begin{bmatrix} -28.8675 \\ 28.8675 \\ 28.8675 \\ 28.8675 \\ 25 \\ -75 \\ 57.7350 \\ 28.8675 \\ 0 \\ 100 \\ 14.4338 \\ -14.4338 \end{bmatrix}$$

6.3.6 Find the force in each bar of the staircase truss shown in the figure by writing the required number of equilibrium equations and then solving them on a computer.



Problem 6.6



Number of equilibrium Equations : 18

Code :

```

1  J = [1 0 0;
2      2 1 1;
3      3 2 2;
4      4 3 3;
5      5 4 4;
6      6 4 3;
7      7 3 2;
8      8 2 1;
9      9 1 0];
10
11 B = [1 1 2;
12      2 2 3;
13      3 3 4;
14      4 4 5;
15      5 5 6;
16      6 6 7;
17      7 7 8;
18      8 8 9;

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```

19      9 1 9;
20     10 2 9;
21     11 2 8;
22     12 3 8;
23     13 3 7;
24     14 7 4;
25     15 4 6;];
26
27 R = [1 1 1 0;
28      2 1 0 1;
29      3 5 1 0;];
30
31 F = [2 0 -150;
32      3 0 -300;
33      4 0 -150;];
34
35 T = TrussAnalyser(B,J,R,F)

```

```

1  function T = TrussAnalyser(B,J,R,F)
2
3      % Receiving Matrix A from mat() %
4      A = TrussMatrix(B,J,R);
5      [sr,g] = size(A);
6
7      % Finding Matrix Containing External Forces %
8      L = zeros(sr,1);
9      [sf, g] = size(F);
10     for i = 1:sf
11         L((F(i)*2)-1,1) = F(i,2);
12         L((F(i)*2),1) = F(i,3);
13     end
14
15     % Finding Tensions on Each Rod Based on External Forces
16     %
17     T = A\L;
18
19     % Checking if values of Tensions are Valid %
20     if MatrixCheck(A,T,L) == 1
21         disp("Values are Valid")
22     else

```



```

22         disp("Values are not Valid")
23     end
24 end

```

```

1  function A = TrussMatrix(B,J,R)
2      % Finding Number of Rows in Matrix A%
3      [num_reactions, ~] = size(R);
4      [sr,g] = size(B);
5      A = zeros(sr+num_reactions,sr+num_reactions);
6
7      % Finding Matrix A Based on Joint and Rod Data%
8      for i = 1:sr
9          % Finding Joints connecting to ith Rod %
10         jnt1 = B(i,2);
11         jnt2 = B(i,3);
12
13         % Finding Angle between ith Rod and Horizontal
14         theta = atan((J(jnt1,3)-J(jnt2,3))/(J(jnt1,2)-
15         J(jnt2,2)));
16         % Adding values of cos and sin theta into A Matrix %
17         A((jnt1*2)-1,i) = cos(theta);
18         A(jnt1*2,i) = sin(theta);
19         A((jnt2*2)-1,i) = -cos(theta);
20         A(jnt2*2,i) = -sin(theta);
21     end
22     for i = 1:num_reactions
23         if R(i,3) == 1
24             A((R(i,2)*2)-1,size(A)-num_reactions+i) = 1;
25         elseif R(i,4) == 1
26             A((R(i,2)*2),size(A)-num_reactions+i) = 1;
27         end
28     end

```

```

1  function a = MatrixCheck(A,T,L)
2      [sr, g] = size(A);
3      a = 1;
4      tol = 0.0001;      % Setting Tolerance Value
5      for i = 1:sr

```

```

6         tmp = A(i,:)*T;
7         disc = abs(tmp-L(i));
8
9         % If the Difference is Greater than the Tolerance %
10        if (disc >= tol)
11            disp(strcat("Not valid at Joint:",num2str(i)))
12            a = 0;
13        end
14    end
15 end

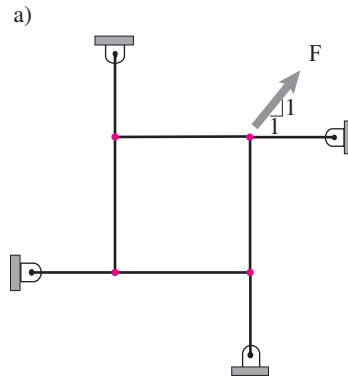
```

Output :

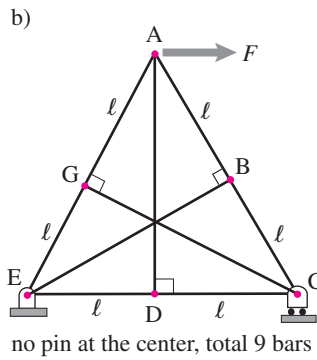
$$T = 10^3 \times \begin{bmatrix} 0.8485 \\ 1.0607 \\ 0.8485 \\ 0.4243 \\ 0.3 \\ 0.4243 \\ 0.6364 \\ 0.4243 \\ -0.3 \\ -0.3 \\ -0.15 \\ -0.15 \\ 0.15 \\ 0.15 \\ 0.3 \\ -0.3 \\ -0.6 \\ 0.3 \end{bmatrix}$$

6.3.7 Find the tensions in all the bars, and all the reactions for these structures.

- a) A square supported by four bars. This is perhaps the simplest rigid structure that has no triangles.

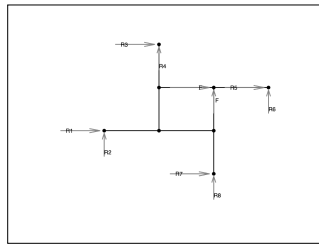


- b) The 9-bar structure shown. This structure also has no triangles in that there is no closed circuit that involves only three bars (for example, from D to A to B to C and back to D involves 4 bars).



Problem 6.7

a)



```

1  J = [1 0 0;
2      2 1 0;
3      3 1 1;
4      4 1 2;
5      5 2 1;
6      6 3 1;
7      7 2 0;
8      8 2 -1];
9
10 B = [1 1 2;
11      2 2 3;
12      3 3 4;
13      4 3 5;
14      5 5 6;
15      6 5 7;
16      7 7 8;
17      8 7 2;];
18
19 R = [1 1 1 0;
20      2 1 0 1;
21      3 4 1 0;
22      4 4 0 1;
23      5 6 1 0;
24      6 6 0 1;

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```

25     7 8 1 0;
26     8 8 0 1;];
27
28 F = [5 1 1];
29
30 T = TrussAnalyser(B,J,R,F)

```

```

1  function T = TrussAnalyser(B,J,R,F)
2
3      % Receiving Matrix A from mat() %
4      A = TrussMatrix(B,J,R);
5      [sr,g] = size(A);
6
7      % Finding Matrix Containing External Forces %
8      L = zeros(sr,1);
9      [sf, g] = size(F);
10     for i = 1:sf
11         L((F(i)*2)-1,1) = F(i,2);
12         L((F(i)*2),1) = F(i,3);
13     end
14
15     % Finding Tensions on Each Rod Based on External Forces
16     %
17     T = A\L;
18
19     % Checking if values of Tensions are Valid %
20     if MatrixCheck(A,T,L) == 1
21         disp("Values are Valid")
22     else
23         disp("Values are not Valid")
24     end
25 end

```

```

1  function A = TrussMatrix(B,J,R)
2      % Finding Number of Rows in Matrix A%
3      [num_reactions, ~] = size(R);
4      [sr,g] = size(B);
5      A = zeros(sr+num_reactions,sr+num_reactions);
6

```

```

7      % Finding Matrix A Based on Joint and Rod Data%
8      for i = 1:sr
9          % Finding Joints connecting to ith Rod %
10         jnt1 = B(i,2);
11         jnt2 = B(i,3);
12
13         % Finding Angle between ith Rod and Horizontal
14         theta = atan((J(jnt1,3)-J(jnt2,3))/(J(jnt1,2)-
J(jnt2,2)));
15         % Adding values of cos and sin theta into A Matrix %
16         A((jnt1*2)-1,i) = cos(theta);
17         A(jnt1*2,i) = sin(theta);
18         A((jnt2*2)-1,i) = -cos(theta);
19         A(jnt2*2,i) = -sin(theta);
20     end
21     for i = 1:num_reactions
22         if R(i,3) == 1
23             A((R(i,2)*2)-1,size(A)-num_reactions+i) = 1;
24         elseif R(i,4) == 1
25             A((R(i,2)*2),size(A)-num_reactions+i) = 1;
26         end
27     end
28 end

```

```

1  function a = MatrixCheck(A,T,L)
2      [sr, g] = size(A);
3      a = 1;
4      tol = 0.0001;      % Setting Tolerance Value
5      for i = 1:sr
6          tmp = A(i,:)*T;
7          disc = abs(tmp-L(i));
8
9          % If the Difference is Greater than the Tolerance %
10         if (disc >= tol)
11             disp(strcat("Not valid at Joint:",num2str(i)))
12             a = 0;
13         end
14     end
15 end

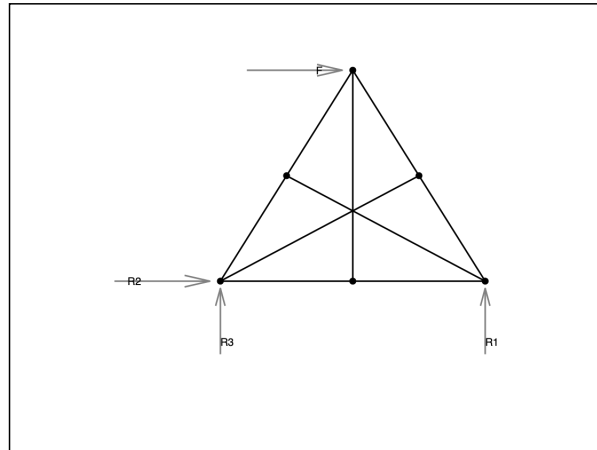
```

Output :

$$T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Disclaimer: We have lost track of the many people who wrote these solutions.
They are not official and have not been reviewed by the authors for correctness.

b)



Code:

```

1  l = 1;
2  F = 1;
3
4  J = [1 2*l*cosd(60) 2*l*sind(60);
5        2 sqrt(5)*l*cosd(30) sqrt(5)*l*sind(30);
6        3 2*l 0;
7        4 l 0;
8        5 0 0;
9        6 l*cosd(60) l*sind(60)];
10
11 B = [1 1 2;
12       2 2 3;
13       3 3 4;
14       4 4 5;
15       5 5 6;
16       6 1 6;
17       7 1 4;
18       8 3 6;
19       9 2 5];
20
21 R = [1 3 0 1;
22       2 5 1 0;
23       3 5 0 1];
24

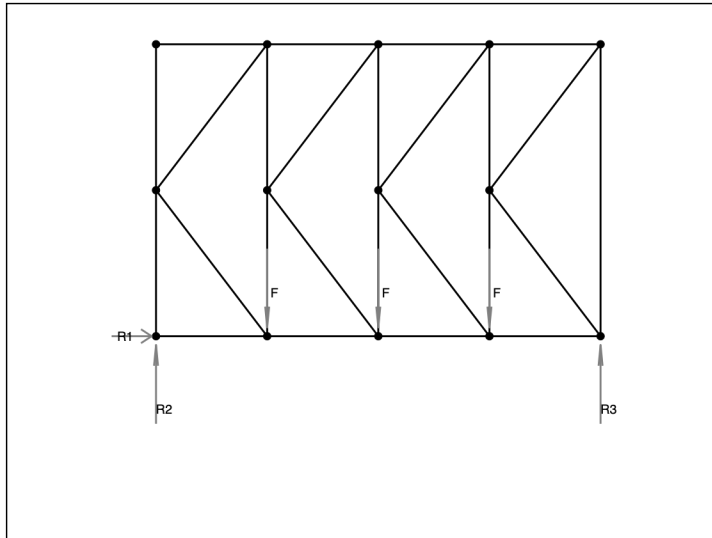
```

```
25 F = [1 F 0];  
26  
27 T = TrussAnalyser(B,J,R,F)
```

Output :

$$T = \begin{bmatrix} 1 \\ 1 \\ 0.5 \\ 0.5 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ -0.866 \\ 1 \\ 0.866 \end{bmatrix}$$

6.3.8 Analyze the truss given in Problem ?? and solve for all bar tensions and support reactions.



Code:

```

1  J = [1 0 0;
2       2 0 1;
3       3 0 2;
4       4 2 2;
5       5 2 1;
6       6 2 0;
7       7 4 2;
8       8 4 1;
9       9 4 0;
10      10 6 2;
11      11 6 1;
12      12 6 0;
13      13 8 2;
14      14 8 0];
15
16  B = [1 1 2;
17       2 2 3;
18       3 3 4;
19       4 4 7;
20       5 7 10;
21       6 10 13;
```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```

22     7 13 14;
23     8 12 14;
24     9 9 12;
25    10 6 9;
26    11 1 6;
27    12 2 6;
28    13 2 4;
29    14 4 5;
30    15 5 6;
31    16 5 9;
32    17 5 7;
33    18 7 8;
34    19 8 9;
35    20 8 12;
36    21 8 10;
37    22 10 11;
38    23 11 12;
39    24 11 14;
40    25 11 13;];
41
42 R = [1 1 1 0;
43      2 1 0 1;
44      3 14 0 1;];
45
46 F = [6 0 -2000;
47      9 0 -2000;
48      12 0 -2000;];
49
50 T = TrussAnalyser(B,J,R,F)

```

```

1 function T = TrussAnalyser(B,J,R,F)
2
3     % Receiving Matrix A from mat() %
4     A = TrussMatrix(B,J,R);
5     [sr,g] = size(A);
6
7     % Finding Matrix Containing External Forces %
8     L = zeros(sr,1);
9     [sf, g] = size(F);
10    for i = 1:sf

```

```

11         L((F(i)*2)-1,1) = F(i,2);
12         L((F(i)*2),1) = F(i,3);
13     end
14
15     % Finding Tensions on Each Rod Based on External Forces
16     %
17     T = A\L;
18     % Checking if values of Tensions are Valid %
19     if MatrixCheck(A,T,L) == 1
20         disp("Values are Valid")
21     else
22         disp("Values are not Valid")
23     end
24 end

```

```

1 function A = TrussMatrix(B,J,R)
2     % Finding Number of Rows in Matrix A%
3     [num_reactions, ~] = size(R);
4     [sr,g] = size(B);
5     A = zeros(sr+num_reactions,sr+num_reactions);
6
7     % Finding Matrix A Based on Joint and Rod Data%
8     for i = 1:sr
9         % Finding Joints connecting to ith Rod %
10        jnt1 = B(i,2);
11        jnt2 = B(i,3);
12
13        % Finding Angle between ith Rod and Horizontal
14        theta = atan((J(jnt1,3)-J(jnt2,3))/(J(jnt1,2)-
15        J(jnt2,2)));
16        % Adding values of cos and sin theta into A Matrix %
17        A((jnt1*2)-1,i) = cos(theta);
18        A(jnt1*2,i) = sin(theta);
19        A((jnt2*2)-1,i) = -cos(theta);
20        A(jnt2*2,i) = -sin(theta);
21    end
22    for i = 1:num_reactions
23        if R(i,3) == 1
24            A((R(i,2)*2)-1,size(A)-num_reactions+i) = 1;

```

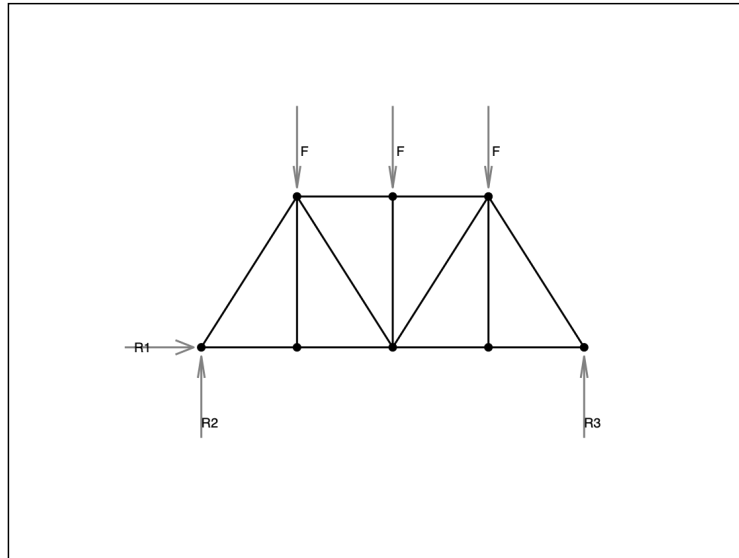
```
24         elseif R(i,4) == 1
25             A((R(i,2)*2),size(A)-num_reactions+i) = 1;
26         end
27     end
28 end
```

```
1  function a = MatrixCheck(A,T,L)
2      [sr, g] = size(A);
3      a = 1;
4      tol = 0.0001;      % Setting Tolerance Value
5      for i = 1:sr
6          tmp = A(i,:)*T;
7          disc = abs(tmp-L(i));
8
9          % If the Difference is Greater than the Tolerance %
10         if (disc >= tol)
11             disp(strcat("Not valid at Joint:",num2str(i)))
12             a = 0;
13         end
14     end
15 end
```

Output :

$$T = 10^3 \times \begin{bmatrix} -3 \\ 0 \\ 0 \\ 3 \\ 4 \\ 3 \\ -1.5 \\ -3 \\ -4 \\ -3 \\ 0 \\ -3.3541 \\ 3.3541 \\ 1.5 \\ 0.5 \\ -1.118 \\ 1.118 \\ 0.5 \\ 1.5 \\ 1.118 \\ -1.118 \\ -0.5 \\ 2.5 \\ 3.3541 \\ -3.3541 \\ 0 \\ -3 \\ -3 \end{bmatrix}$$

6.3.9 Solve Problem ??.



Code:

```

1  J = [1 0 0;
2      2 1 1;
3      3 1 0;
4      4 2 1;
5      5 2 0;
6      6 3 1;
7      7 3 0;
8      8 4 0;];
9
10 B = [1 1 2;
11     2 2 4;
12     3 4 6;
13     4 6 8;
14     5 7 8;
15     6 5 7;
16     7 3 5;
17     8 1 3;
18     9 2 3;
19    10 2 5;
20    11 4 5;
21    12 5 6;

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```

22     13 6 7;];
23
24 R = [1 1 1 0;
25       2 1 0 1;
26       3 8 0 1;];
27
28 F = [2 0 -2000;
29       4 0 -5000;
30       6 0 -2000;];
31
32 T = TrussAnalyser(B,J,R,F)

```

```

1  function T = TrussAnalyser(B,J,R,F)
2
3      % Receiving Matrix A from mat() %
4      A = TrussMatrix(B,J,R);
5      [sr,g] = size(A);
6
7      % Finding Matrix Containing External Forces %
8      L = zeros(sr,1);
9      [sf, g] = size(F);
10     for i = 1:sf
11         L((F(i)*2)-1,1) = F(i,2);
12         L((F(i)*2),1) = F(i,3);
13     end
14
15     % Finding Tensions on Each Rod Based on External Forces
16     %
17     T = A\L;
18
19     % Checking if values of Tensions are Valid %
20     if MatrixCheck(A,T,L) == 1
21         disp("Values are Valid")
22     else
23         disp("Values are not Valid")
24     end
25 end

```

```

1  function A = TrussMatrix(B,J,R)

```

```

2      % Finding Number of Rows in Matrix A%
3      [num_reactions, ~] = size(R);
4      [sr,g] = size(B);
5      A = zeros(sr+num_reactions,sr+num_reactions);
6
7      % Finding Matrix A Based on Joint and Rod Data%
8      for i = 1:sr
9          % Finding Joints connecting to ith Rod %
10         jnt1 = B(i,2);
11         jnt2 = B(i,3);
12
13         % Finding Angle between ith Rod and Horizontal
14         theta = atan((J(jnt1,3)-J(jnt2,3))/(J(jnt1,2)-
J(jnt2,2)));
15         % Adding values of cos and sin theta into A Matrix %
16         A((jnt1*2)-1,i) = cos(theta);
17         A(jnt1*2,i) = sin(theta);
18         A((jnt2*2)-1,i) = -cos(theta);
19         A(jnt2*2,i) = -sin(theta);
20     end
21     for i = 1:num_reactions
22         if R(i,3) == 1
23             A((R(i,2)*2)-1,size(A)-num_reactions+i) = 1;
24         elseif R(i,4) == 1
25             A((R(i,2)*2),size(A)-num_reactions+i) = 1;
26         end
27     end
28 end

```

```

1  function a = MatrixCheck(A,T,L)
2      [sr, g] = size(A);
3      a = 1;
4      tol = 0.0001;      % Setting Tolerance Value
5      for i = 1:sr
6          tmp = A(i,:)*T;
7          disc = abs(tmp-L(i));
8
9          % If the Difference is Greater than the Tolerance %
10         if (disc >= tol)
11             disp(strcat("Not valid at Joint:",num2str(i)))

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.


```

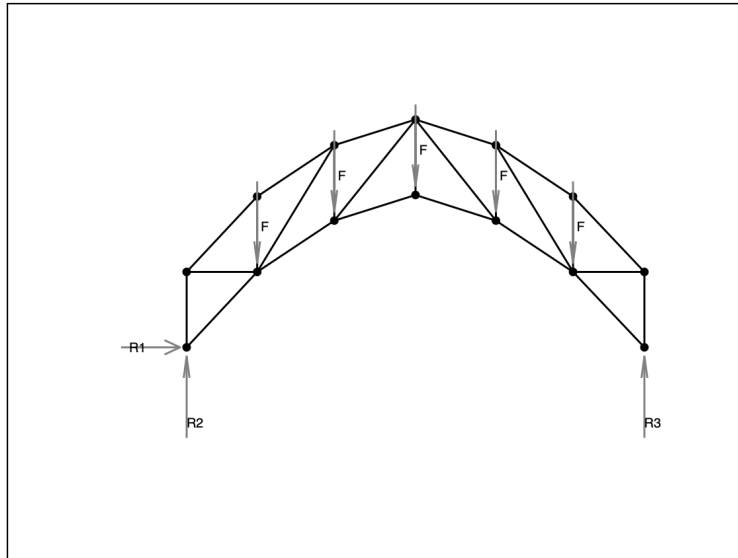
12         a = 0;
13     end
14 end
15 end

```

Output :

$$T = 10^3 \times \begin{bmatrix} 6.364 \\ 7 \\ 7 \\ 6.364 \\ -4.5 \\ -4.5 \\ -4.5 \\ -4.5 \\ 0 \\ -3.5355 \\ -5 \\ -3.5355 \\ 0 \\ 0 \\ -4.5 \\ -4.5 \end{bmatrix}$$

6.3.10 Solve Problem ??.



Code:

```

1  J = [1 0 0;
2      2 0 0.5;
3      3 0.86 1;
4      4 1.8 1.34;
5      5 2.79 1.51;
6      6 3.77 1.34;
7      7 4.71 1;
8      8 5.58 0.5;
9      9 5.58 0;
10     10 4.71 0.5;
11     11 3.77 0.84;
12     12 2.79 1.01;
13     13 1.8 0.84;
14     14 0.86 0.5];
15
16 B = [1 1 2;
17     2 2 3;
18     3 3 4;
19     4 4 5;
20     5 5 6;
21     6 6 7;

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```

22     7 7 8;
23     8 8 9;
24     9 9 10;
25    10 10 11;
26    11 11 12;
27    12 12 13;
28    13 13 14;
29    14 1 14;
30    15 2 14;
31    16 3 14;
32    17 4 13;
33    18 5 12;
34    19 6 11;
35    20 7 10;
36    21 8 10;
37    22 4 14;
38    23 5 13;
39    24 5 11;
40    25 6 10;];
41
42 R = [1 1 1 0;
43      2 1 0 1;
44      3 9 0 1;];
45
46 F = [10 0 -20*1000;
47      11 0 -20*1000;
48      12 0 -20*1000;
49      13 0 -20*1000;
50      14 0 -20*1000;];
51
52 T = TrussAnalyser(B,J,R,F)

```

```

1 function T = TrussAnalyser(B,J,R,F)
2
3     % Receiving Matrix A from mat() %
4     A = TrussMatrix(B,J,R);
5     [sr,g] = size(A);
6
7     % Finding Matrix Containing External Forces %
8     L = zeros(sr,1);

```

```

9      [sf, g] = size(F);
10     for i = 1:sf
11         L((F(i)*2)-1,1) = F(i,2);
12         L((F(i)*2),1) = F(i,3);
13     end
14
15     % Finding Tensions on Each Rod Based on External Forces
16     %
17     T = A\L;
18
19     % Checking if values of Tensions are Valid %
20     if MatrixCheck(A,T,L) == 1
21         disp("Values are Valid")
22     else
23         disp("Values are not Valid")
24     end
25 end

```

```

1  function A = TrussMatrix(B,J,R)
2      % Finding Number of Rows in Matrix A%
3      [num_reactions, ~] = size(R);
4      [sr,g] = size(B);
5      A = zeros(sr+num_reactions,sr+num_reactions);
6
7      % Finding Matrix A Based on Joint and Rod Data%
8      for i = 1:sr
9          % Finding Joints connecting to ith Rod %
10         jnt1 = B(i,2);
11         jnt2 = B(i,3);
12
13         % Finding Angle between ith Rod and Horizontal
14         theta = atan((J(jnt1,3)-J(jnt2,3))/(J(jnt1,2)-
15         J(jnt2,2)));
16         % Adding values of cos and sin theta into A Matrix %
17         A((jnt1*2)-1,i) = cos(theta);
18         A(jnt1*2,i) = sin(theta);
19         A((jnt2*2)-1,i) = -cos(theta);
20         A(jnt2*2,i) = -sin(theta);
21     end
22     for i = 1:num_reactions

```

```
22     if R(i,3) == 1
23         A((R(i,2)*2)-1,size(A)-num_reactions+i) = 1;
24     elseif R(i,4) == 1
25         A((R(i,2)*2),size(A)-num_reactions+i) = 1;
26     end
27 end
28 end
```

```
1  function a = MatrixCheck(A,T,L)
2      [sr, g] = size(A);
3      a = 1;
4      tol = 0.0001;      % Setting Tolerance Value
5      for i = 1:sr
6          tmp = A(i,:)*T;
7          disc = abs(tmp-L(i));
8
9          % If the Difference is Greater than the Tolerance %
10         if (disc >= tol)
11             disp(strcat("Not valid at Joint:",num2str(i)))
12             a = 0;
13         end
14     end
15 end
```

Output :

$$T = 1.0e + 05 \times \begin{bmatrix} -0.5007 \\ 0.9962 \\ 0.9158 \\ 1.4475 \\ 1.4528 \\ 0.9238 \\ 1.0020 \\ -0.4993 \\ 0 \\ 1.5222 \\ 1.6503 \\ 1.6498 \\ 1.5170 \\ 0 \\ -0.8612 \\ 0.1892 \\ 0.5717 \\ 0.7613 \\ 0.5687 \\ 0.1851 \\ 0.8688 \\ -0.7582 \\ -0.2408 \\ 0.2357 \\ 0.7546 \\ 0 \\ -0.5007 \\ -0.4993 \end{bmatrix}$$

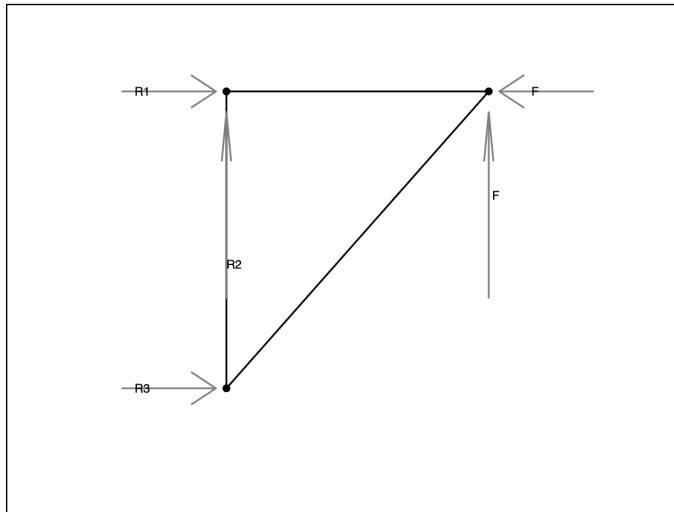
6.3.11 Using your program from problem 6.3.4 solve each of the following problems:

- a) Problem ??
- b) Problem ??

- c) Problem ??
- d) Problem ??
- e) Problem ??
- f) Problem ??

Code:

a)



```

1  J = [1 0 0.86;
2      2 2 0.86;
3      3 0 0;];
4
5  B = [1 1 2;
6      2 2 3;
7      3 1 3;];
8
9  R = [1 1 1 0;
10     2 1 0 1;
11     3 3 1 0;];
12
13 F = [2 173*cosd(120) 173*sind(120)];
14
15 T = TrussAnalyser(B,J,R,F)

```

```

1  function T = TrussAnalyser(B,J,R,F)
2
3      % Receiving Matrix A from mat() %
4      A = TrussMatrix(B,J,R);

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```

5      [sr,g] = size(A);
6
7      % Finding Matrix Containing External Forces %
8      L = zeros(sr,1);
9      [sf, g] = size(F);
10     for i = 1:sf
11         L((F(i)*2)-1,1) = F(i,2);
12         L((F(i)*2),1) = F(i,3);
13     end
14
15     % Finding Tensions on Each Rod Based on External Forces
16     %
17     T = A\L;
18
19     % Checking if values of Tensions are Valid %
20     if MatrixCheck(A,T,L) == 1
21         disp("Values are Valid")
22     else
23         disp("Values are not Valid")
24     end
25 end

```

```

1  function A = TrussMatrix(B,J,R)
2      % Finding Number of Rows in Matrix A%
3      [num_reactions, ~] = size(R);
4      [sr,g] = size(B);
5      A = zeros(sr+num_reactions,sr+num_reactions);
6
7      % Finding Matrix A Based on Joint and Rod Data%
8      for i = 1:sr
9          % Finding Joints connecting to ith Rod %
10         jnt1 = B(i,2);
11         jnt2 = B(i,3);
12
13         % Finding Angle between ith Rod and Horizontal
14         theta = atan((J(jnt1,3)-J(jnt2,3))/(J(jnt1,2)-
15         J(jnt2,2)));
16         % Adding values of cos and sin theta into A Matrix %
17         A((jnt1*2)-1,i) = cos(theta);
18         A(jnt1*2,i) = sin(theta);

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.


```

18     A((jnt2*2)-1,i) = -cos(theta);
19     A(jnt2*2,i) = -sin(theta);
20 end
21 for i = 1:num_reactions
22     if R(i,3) == 1
23         A((R(i,2)*2)-1,size(A)-num_reactions+i) = 1;
24     elseif R(i,4) == 1
25         A((R(i,2)*2),size(A)-num_reactions+i) = 1;
26     end
27 end
28 end

```

```

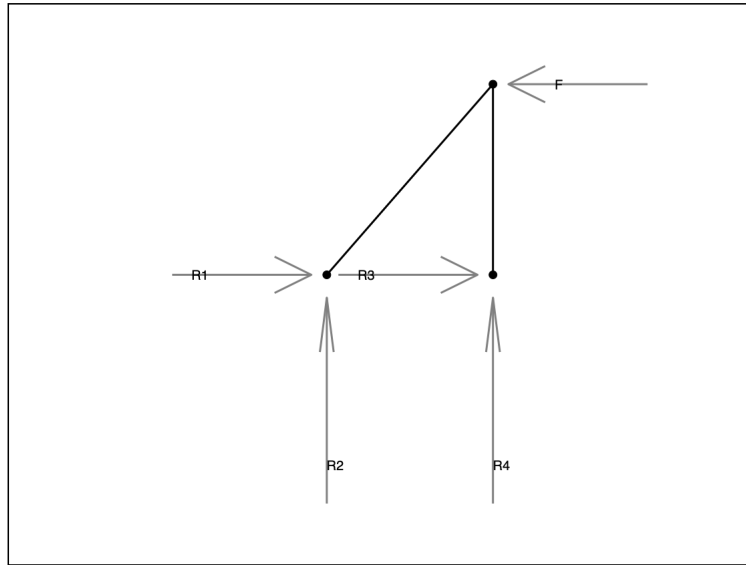
1 function a = MatrixCheck(A,T,L)
2     [sr, g] = size(A);
3     a = 1;
4     tol = 0.0001;    % Setting Tolerance Value
5     for i = 1:sr
6         tmp = A(i,:)*T;
7         disc = abs(tmp-L(i));
8
9         % If the Difference is Greater than the Tolerance %
10        if (disc >= tol)
11            disp(strcat("Not valid at Joint:",num2str(i)))
12            a = 0;
13        end
14    end
15 end

```

Output :

$$T = \begin{bmatrix} 434.9242 \\ 379.2706 \\ -149.8224 \\ -434.9242 \\ 149.8224 \\ 348.4242 \end{bmatrix}$$

b)



```

1  J = [1 -0.86 0;
2       2 0 0.5;
3       3 0 0];
4
5  B = [1 1 2;
6       2 2 3];
7
8  R = [1 1 1 0;
9       2 1 0 1;
10      3 3 1 0;
11      4 3 0 1];
12
13 F = [2 -300 0];
14
15 T = TrussAnalyser(B,J,R,F)

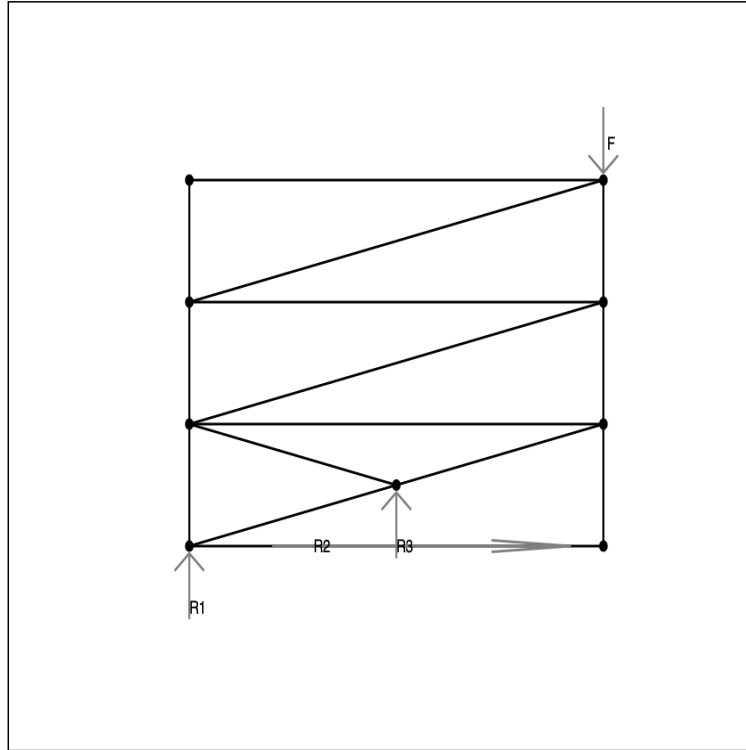
```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

Output :

$$T = \begin{bmatrix} 347.0185 \\ 174.4186 \\ -300 \\ -174.4186 \\ 0 \\ 174.4186 \end{bmatrix}$$

c)



```

1  J = [1 0 3;
2      2 1 3;
3      3 0 2;
4      4 1 2;
5      5 0 1;
6      6 1 1;
7      7 0 0;
8      8 0.5 0.5;
9      9 1 0];
10
11 B = [1 1 2;
12     2 2 4;
13     3 4 6;
14     4 6 9;
15     5 7 9;
16     6 5 7;

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```

17      7 3 5;
18      8 1 3;
19      9 2 3;
20     10 3 4;
21     11 4 5;
22     12 5 6;
23     13 5 8;
24     14 6 8;
25     15 7 8;];
26
27  R = [1 7 0 1;
28      2 9 1 0;
29      3 8 0 1;];
30
31  F = [2 0 -2];
32
33  T = TrussAnalyser(B,J,R,F)

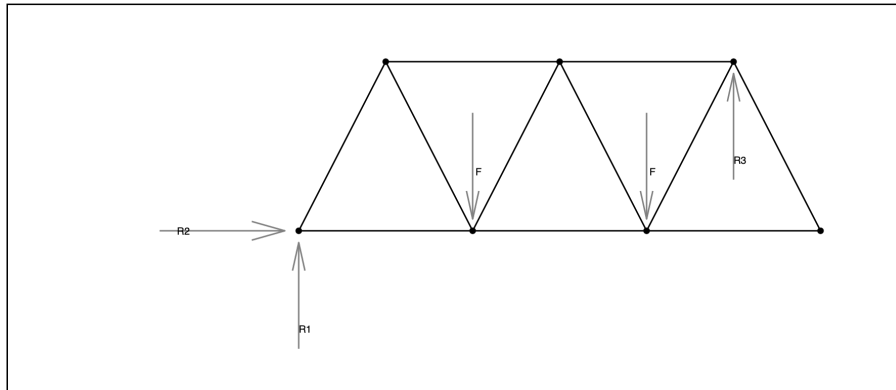
```

Output :

$$T = \begin{bmatrix} 0 \\ -2 \\ -2 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -2 \\ 2.8284 \\ -2.8284 \\ 0 \\ 2 \\ 0 \\ -4 \end{bmatrix}$$

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

d)



```

1  J = [1 0 0;
2      2 0.5 0.86;
3      3 1 0;
4      4 1.5 0.86;
5      5 2 0;
6      6 2.5 0.86;
7      7 3 0;];
8
9  B = [1 1 2;
10     2 2 4;
11     3 4 6;
12     4 6 7;
13     5 5 7;
14     6 3 5;
15     7 1 3;
16     8 2 3;
17     9 3 4;
18    10 4 5;
19    11 5 6;];
20
21 R = [1 1 0 1;
22     2 1 1 0;
23     3 6 0 1;];
24
25 F = [3 0 -4000;
26     5 0 -1000;];

```

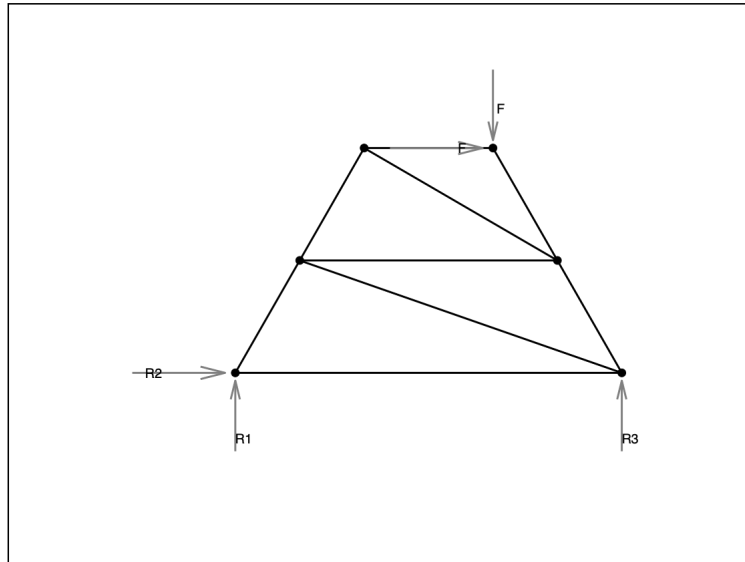
Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```
27  
28 T = TrussAnalyser(B,J,R,F)
```

Output :

$$T = 10^3 \times \begin{bmatrix} 3.0075 \\ 3.0233 \\ 1.3953 \\ 0 \\ 0 \\ -2.2093 \\ -1.5116 \\ -3.0075 \\ -1.6194 \\ 1.6194 \\ -2.7761 \\ -2.6 \\ 0 \\ -2.4 \end{bmatrix}$$

e)



```

1  l = sym('l');
2
3  J = [1 0 0;
4       2 l/2 0.86*l;
5       3 l 2*l*0.86;
6       4 2*l 2*l*0.86;
7       5 (5/2)*l 0.86*l;
8       6 3*l 0;];
9
10 B = [1 1 2;
11       2 2 3;
12       3 3 4;
13       4 4 5;
14       5 5 6;
15       6 1 6;
16       7 2 6;
17       8 2 5;
18       9 3 5;];
19
20 R = [1 1 0 1;
21       2 1 1 0;
22       3 6 0 1;];

```

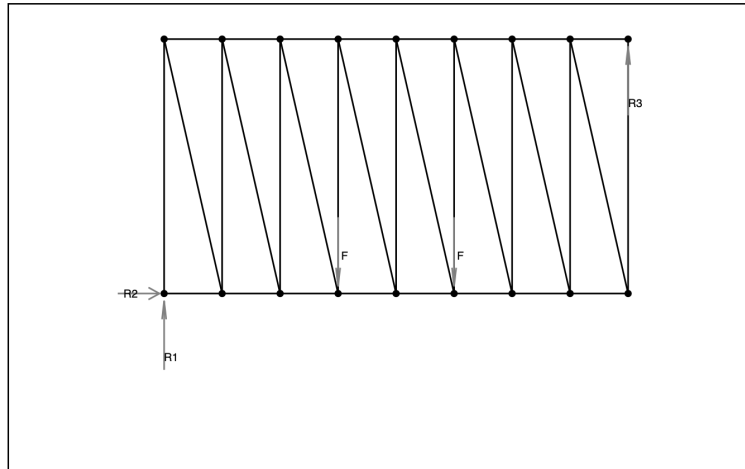
Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.


```
23
24 F = [4 3000*cosd(45) -3000*sind(45)];
25
26 T = TrussAnalyser(B,J,R,F)
```

Output :

$$T = 10^3 \times \begin{bmatrix} -0.5889 \\ -0.4417 \\ -0.8880 \\ 2.4538 \\ 2.8955 \\ -1.8253 \\ 0.3913 \\ -0.4440 \\ 0.7677 \\ 0.5091 \\ 2.1213 \\ -2.6304 \end{bmatrix}$$

f)



```

1  J = [1 0 0;
2      2 0 2;
3      3 1 0;
4      4 1 2;
5      5 2 0;
6      6 2 2;
7      7 3 0;
8      8 3 2;
9      9 4 0;
10     10 4 2;
11     11 5 0;
12     12 5 2;
13     13 6 0;
14     14 6 2;
15     15 7 0;
16     16 7 2;
17     17 8 0;
18     18 8 2;];
19
20  B = [1 1 2;
21      2 2 4;
22      3 4 6;
23      4 6 8;
24      5 8 10;
25      6 10 12;

```

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.

```
26      7 12 14;
27      8 14 16;
28      9 16 18;
29      10 17 18;
30      11 15 17;
31      12 13 15;
32      13 11 13;
33      14 9 11;
34      15 7 9;
35      16 5 7;
36      17 3 5;
37      18 1 3;
38      19 2 3;
39      20 3 4;
40      21 4 5;
41      22 5 6;
42      23 6 7;
43      24 7 8;
44      25 8 9;
45      26 9 10;
46      27 10 11;
47      28 11 12;
48      29 12 13;
49      30 13 14;
50      31 14 15;
51      32 15 16;
52      33 16 17;];
53
54 R = [1 1 0 1;
55      2 1 1 0;
56      3 18 0 1;];
57
58 F = [7 0 -10*1000;
59      11 0 -10*1000;];
60
61 T = TrussAnalyser(B,J,R,F)
```

Output :

$$T = 10^4 \times \begin{bmatrix} -1 \\ 0.5 \\ 1 \\ 1.5 \\ 1.5 \\ 1.5 \\ 1 \\ 0.5 \\ 0 \\ 1 \\ -0.5 \\ -1 \\ -1.5 \\ -1.5 \\ -1.5 \\ -1 \\ -0.5 \\ 0 \\ -1.118 \\ -1 \\ -1.118 \\ -1 \\ -1.118 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1.118 \\ 1 \\ 1.118 \\ 1 \\ 1.118 \\ -1 \\ 0 \\ -1 \end{bmatrix}$$

Disclaimer: We have lost track of the many people who wrote these solutions. They are not official and have not been reviewed by the authors for correctness.