

# SC 2008 Workshop Handout

Daniel D. Warner  
Clemson University  
November 8, 2008

## 1 Elementary Calculations

MATLAB can be used as a calculator by typing expressions into the command window. More importantly, in keeping with its origins as a Matrix Laboratory, it can conveniently do extensive matrix calculations.

### 1.1 Three Bungee Jumpers

Three Bungee jumpers - Alex, Barry, and Carol - plan to form a single long bungee cord by tying themselves together with three short bungee cords and then jump off a bridge. That is, the first jumper, Alex, is tied to the bridge by a bungee cord, the second jumper, Barry, is tied to Alex with a second bungee cord, and the third jumper, Carol, is tied to Barry with a third bungee cord. Our goal is to determine the final positions of the three jumpers.

Let  $z_A$ ,  $z_B$ , and  $z_C$  denote the distance below the bridge for Alex, Barry, and Carol, respectively. The distances will be measured in feet, and the distance will be a positive number as long as the jumper is below the bridge. Assume that the weights of the three jumpers are:  $w_A = 225$  lbs,  $w_B = 160$  lbs, and  $w_C = 105$  lbs, respectively. Assume that each of the short cords is 15 feet long. Furthermore, assume that the behavior of the bungee cords is similar to idealized springs, in that the restoring force is proportional to the amount that the cord has been stretched, and assume that the proportionality constants are  $c_1 = 75$  lbs/ft and  $c_2 = c_3 = 30$  lbs/ft, respectively. Finally, assume that the weights of the bungee cords are negligible.

### 1.2 A Static Equilibrium Model

Start by considering the static equilibrium case. When they reach equilibrium after jumping off the bridge, how far below the bridge will each jumper be? One direct approach to answering this question is to consider the force on each bungee cord separately. The first bungee cord is being stretched by the weight of all three jumpers; the second bungee cord is only being stretched by the weight of the Barry and Carol; and the third bungee cord is only being stretched by the weight of Carol. Applying Hooke's Law, the elongations for the bungee cords must satisfy the following equations.

$$\begin{aligned}c_1 e_1 &= w_A + w_B + w_C \\c_2 e_2 &= w_B + w_C \\c_3 e_3 &= w_C\end{aligned}$$

**Exercise 1.** Using MATLAB as a calculator, solve these equations to find the elongation for each bungee cord. Then, add up the elongations and the original lengths of the cords to determine exactly how far below the bridge Alex, Barry, and Carol end up. If the bridge is 80 feet above the river, will Carol be underwater when all the bouncing stops? If not, do you think that there is a good chance that she might be wet?

### 1.3 Matrix Formulation

We can also write these three equations as a single matrix equation. To do this, we start by letting

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{bmatrix} w_A \\ w_B \\ w_C \end{bmatrix}$$

Using the vectors  $\mathbf{e}$  and  $\mathbf{w}$  we can summarize the three equations as the single matrix equation

$$\mathbf{C}\mathbf{e} = \mathbf{A}\mathbf{w} \tag{1}$$

where  $\mathbf{C}$  is the diagonal matrix

$$\mathbf{C} = \begin{bmatrix} c_1 & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_3 \end{bmatrix} = \begin{bmatrix} 75 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 30 \end{bmatrix}$$

The matrix  $\mathbf{A}$  is a  $3 \times 3$  matrix which you should be able to figure out for yourself. Hint, every entry in  $\mathbf{A}$  is either 0 or 1. We can solve this matrix equation for the vector of elongations by multiplying both sides of Equation 1 by  $\mathbf{C}^{-1}$ . In other words

$$\mathbf{e} = \mathbf{C}^{-1}\mathbf{A}\mathbf{w}$$

Given,  $\mathbf{A}$  and  $\mathbf{e}$  we can find the positions

$$\mathbf{z} = \begin{bmatrix} z_A \\ z_B \\ z_C \end{bmatrix}$$

from the equation

$$\mathbf{z} = \mathbf{A}^T(\mathbf{e} + \mathbf{L})$$

which in MATLAB is written as

$$\mathbf{z} = \mathbf{A}' * (\mathbf{e} + \mathbf{L})$$

since MATLAB uses the apostrophe to indicate the transpose of a matrix.

**Exercise 2.** Using MATLAB, calculate the matrices  $\mathbf{C}$  and  $\mathbf{A}$  and solve the matrix equation, Equation 1, for the vector of elongations. In MATLAB we can create a diagonal matrix  $\mathbf{C}$  from a vector  $\mathbf{c}$  with the statement

$$\mathbf{C} = \text{diag}(\mathbf{c})$$

We can also solve Equation 1 by writing either

$$\mathbf{e} = \text{inv}(\mathbf{C}) * \mathbf{A} * \mathbf{w}$$

or

$$\mathbf{e} = \mathbf{C}^{-1} * \mathbf{A} * \mathbf{w}$$

or

$$\mathbf{e} = \mathbf{C} \setminus (\mathbf{A} * \mathbf{w})$$

Calculate  $\mathbf{e}$ , the vector of elongations, all three ways and compare the results with your previous calculations.

## 2 Elementary Plotting

One of the most important pedagogical features of MATLAB is its simple but powerful plot commands. We will first illustrate this with a Proper Plot suitable for engineering student reports. Then we will illustrate its power in illustrating truss diagrams.

### 2.1 Proper Plot

Workshop Example 3 plots a simple oscillating function and its derivative on the interval  $[0, 4]$ . It includes a title, labels for the  $x$  and  $y$  axes, a legend, and background grid. In addition, we have used the MATLAB built-in function, `fzero`, to find the zero of the derivative near  $x = 2.5$ . Then we plotted the corresponding local minimum of the function using a red circle, and also labeled it.

**Exercise 3.** Modify the parameters in `f321` and `fp321`, then adjust the commands in `WorkshopEx03.m` so that the display of the new plot is appropriate.

One of the more important efficiency issues is to avoid unnecessary loops. We do this by using MATLAB's vector, matrix, and array operations as much as possible. Several examples of this are shown in this Proper Plot example. The statement

```
x = 0:0.05:4;
```

generates a row vector of 81  $x$  values. This vector is passed to the functions `f321` and `fp321` as a single argument. These functions, in turn, return the vectors `y` and `yp`. The printout of these vectors can be suppressed by appending semicolons to the end of those three statements.

The line that does the actual calculation contains an important operator that distinguishes between matrices and arrays. In MATLAB all operations are assumed to be matrix operations. In the functions `f321` and `fp321` we simply want to evaluate the function

$$(x - a)(\cos(bx) + \sin(cx))$$

on each  $x$  value separately. This means that we really want to view `x` as an array and not a vector. Multiplication by a scalar means the same thing in either case, so `b*x` and `c*x` are no problem. When applying an elementary function to a vector, MATLAB always interprets the vector as an array and applies the function to each component of the vector. Addition of vectors, matrices, and arrays means the same thing in all cases, so the calculation of the expression `(cos(b*x) + sin(c*x))` results in a vector. Multiplying this by `(x-a)` raises two issues. First, subtracting a scalar from a vector only makes sense as an array calculation, so that is how MATLAB interprets it. Thus we have two row vectors `(x-a)` and `(cos(b*x) + sin(c*x))`. On the one hand matrix and vector multiplication is a legitimate possibility, and hence that will be MATLAB's interpretation. But multiplying a  $1 \times n$  vector by another  $1 \times n$  vector is not defined, and MATLAB will generate an error message. We indicate that the multiplication should be interpreted as multiplying corresponding elements of the two arrays by using the special operator `.*`.

## 2.2 Quick Review On Trusses

This Presentation will focus in part on developing a MATLAB program to provide a static analysis of a simple structure, called a truss. A truss is an idealized structure that is built from elastic bars. Each bar resists a change in its length, and that resisting force permits the structure to withstand a load. The bars are connected at points called nodes with pin joints, which can only transmit forces *along* the bars. The pin joints allow the bars to point in any direction, and each bar as a whole can turn. Bars are simpler than beams, plates, and shells. Unlike beams, they do not bend; unlike plates, they are one-dimensional; and unlike shells, they are simple and straight.

Figure 1: Three Trusses

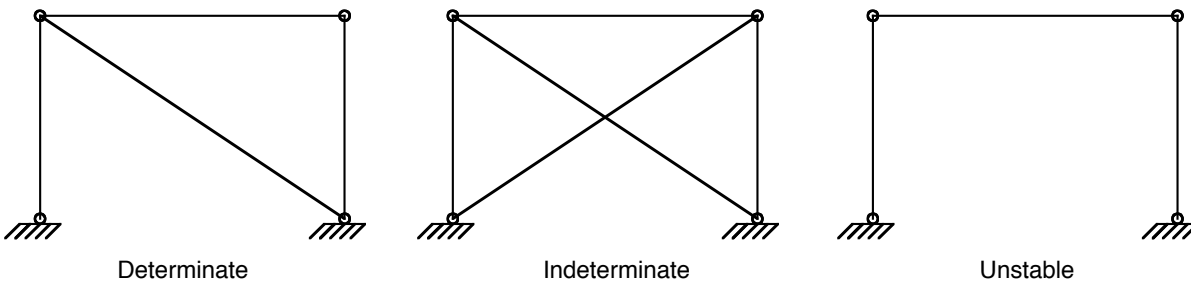


Figure 1 shows three trusses. All three have 4 nodes. The first has 4 bars; the second has 5 bars; and the third has only 3 bars. The diagram indicates that the two nodes on the bottom of each truss are fixed. That is, they are assumed to be immovable. This means that the two nodes at the top of each truss can each move in the  $x$  and  $y$  direction, which means that each of these trusses has two nodes each of which can move in two directions for a total of 4 degrees of freedom in terms of displacements. In addition each truss has an internal force acting along each bar. The first truss is called *Determinate* because it has 4 degrees of freedom in terms of displacements and exactly 4 internal forces. Determinate trusses are relatively easy to analyze by hand. The second truss is called *Indeterminate* because while it has exactly 4 degrees of freedom in terms of displacements, it has more than 4 internal forces. Completing its analysis means that we must determine the forces and displacements that minimize the potential energy of the system. The third truss is unstable. Fortunately, the linear algebra approach that we are going to explore will determine the unique set of displacements and internal forces for both determinate and indeterminate trusses and will also indicate when we have an unstable truss.

## 2.3 Specifying and Drawing the Truss

At this point we want to get things started by considering how to concisely describe the nodes and bars of a truss. Then we want to develop a computer program to evaluate this description and draw the corresponding diagram. To describe the nodes we need to describe the  $x$ - $y$  coordinates for each node. In MATLAB the most direct approach is to simply describe a matrix with  $n$  rows and 2 columns, where  $n$  is the number of nodes. If we assume that the trusses in Figure 1 are 3 units wide and 2 units high and that the bottom left node is at the origin, then the MATLAB code would simply be

```
nodes = [ 0.0  0.0
          3.0  0.0
          0.0  2.0
          3.0  2.0 ]
```

In a similar fashion we can describe the bars using a matrix with  $nb$  rows and 2 columns. The first column contains the number of the node at which the bar starts and the second column contains the number of the node at which the bar terminates. Sometimes we refer to these as the source and target nodes. The MATLAB code to describe the bars for the determinant truss in Figure 1 is simply

```
bars = [ 1  3
         2  3
         2  4
         3  4 ]
```

The MATLAB script file, `truss01.m`, includes these two blocks of code as well as ample comments to clarify the specification of the truss.

A straightforward MATLAB code to input and draw the truss described in the script file is following:

```
name = input('Enter the name of the file defining the truss: ', 's');
eval(name);
```

At this point, we have all the information we need to draw the truss. First we plot the nodes. Since the first column of the nodes matrix contains the  $x$  coordinates and the second column contains the  $y$  coordinates we can plot all the nodes as small black circles with the following code

```
% Plot the nodes
xn = nodes(:,1);
yn = nodes(:,2);
plot(xn,yn,'ok')
```

Then we can add the file name as the title of the plot and simply walk through the list of bars, and for each bar we draw a straight line from the first node of the bar to the second node of the bar. The  $x$  and  $y$  coordinates of these nodes are given in the corresponding columns of the `nodes` matrix. However, since we had already copied the first and second rows of the `nodes` matrix into `xn` and `yn` arrays, using these arrays keeps the code a little shorter and a little clearer.

```
[nb,two] = size(bars);
% Plot the bars
for k=1:nb
```

```

    % Get the index for the first node (or source node) of bar k
    s = bars(k,1);
    % Get the index for the second node (or target node) of bar k
    t = bars(k,2);
    plot([xn(s) xn(t)], [yn(s) yn(t)])
end

```

Note that the default behavior of the `plot` command is to erase the window and draw the new plot. If we want to draw on top of the original plot then we need to bracket the the additional plot commands with `hold on` and `hold off` statements, `WorkshopEx04.m` implements these steps.

## 2.4 Adding A Little Splash

The figure drawn by `WorkshopEx04.m` is not entirely satisfactory. First, we need a small margin around the figure, and second, we really need to label the nodes. `WorkshopEx05.m` includes two additional blocks of code to accomplish this.

This first block determines a horizontal margin and a vertical margin and then uses the `axis` command to set the window to this slightly larger view.

```

    % Set the view, include a small margin around the truss
    xmargin = 0.1*max(abs(xn));
    ymargin = 0.1*max(abs(yn));
    xmin = min(xn) - xmargin;
    xmax = max(xn) + xmargin;
    ymin = min(yn) - ymargin;
    ymax = max(yn) + ymargin;
    axis([xmin xmax ymin ymax]);

```

This second block of code uses the `text` command and the previous margins to label each of the nodes.

```

% label the nodes
xoffset = xmargin/4;
yoffset = ymargin/4;
for k=1:nn
    text(xn(k)+xoffset, yn(k)+yoffset, num2str(k))
end

```

**Exercise 4.** Construct a file, `truss02.m`, which specifies a triangular truss with three nodes and three bars. The nodes should be located at  $(0, 0)$ ,  $(5, 0)$ , and  $(10, 5)$ . Generate a plot of this truss using `WorkshopEx05.m`.

**Exercise 5.** Construct a file, `truss03.m`, which specifies a truss of your own design with five or more nodes. Generate a plot of this truss using `WorkshopEx05.m`.

## 2.5 A Short Mathematical Detour

Before we delve into engineering questions there are a few things we can learn from the information we already have in hand. First, the diagram can also represent a mathematical object called a *graph*. This is not the same object as the graph of a function. This graph is an abstraction of a truss that simply represents a set of points and connections between pairs of points. The points are usually called nodes and the connection between a pair of nodes is called an edge. The nodes of the graph correspond to the nodes of the truss, and the edges of the graph correspond to the bars in the truss. When we think of the diagram as representing a graph, then a line connecting two nodes represents an edge in the graph. On the other hand, when we think of the diagram as representing a truss, then a line connecting two nodes represents a bar of the truss.

One of the major hurdles for engineering students who are new to programming is the insight in how to extract the appropriate information from the **nodes** and **bars** matrices that will be needed in order to build the structural matrices needed to analyze a truss. Building the following edge-node matrix is an excellent first step down that path.

The edge-node matrix represents the structure of the graph. It is a “structural” matrix that contains one row for each edge and one column for each node. The row corresponding to a given edge will have all zero entries except in the columns corresponding to the two nodes. The entry for the first node will be  $-1$  and the entry for the second node will be  $+1$ . If we consider the first diagram in Figure 1, then the edge-node matrix is

$$\mathbf{E}_1 = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

If we consider the second diagram in Figure 1, then there is a fifth edge, and the edge-node matrix has an additional row.

$$\mathbf{E}_2 = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

These two edge-node matrices don't really contain any more information than we already had with the list containing the pair of nodes for each edge. However, this matrix structure leads to some interesting and useful calculations.

If we take the transpose of the edge-node matrix and multiply it times the edge-node matrix itself, we get a square matrix with the same number of rows and columns as the number of nodes. This matrix is called the *Laplacian matrix* of the graph. For the first diagram of Figure 1 we get

$$\mathbf{L}_1 = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

For the second diagram of Figure 1 we get

$$\mathbf{L}_2 = \begin{bmatrix} 2 & 0 & -1 & -1 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

These Laplacian matrices have two particularly interesting properties. First, the entries on the diagonal count how many edges are connected to the corresponding node. For example, in the first diagram node 1 only has one edge connected to it, while in the second diagram it has two edges connected to it. Similarly, in the first diagram node 4 has two edges connected to it, while in the second diagram it has three edges connected to it. On the other hand, nodes 2 and 3 have the same number of edges connected to them in both diagrams – namely, two edges and three edges, respectively.

The second interesting property is that all the off-diagonal elements are either 0 or  $-1$ , and there is an edge connecting node  $i$  and node  $j$  if and only if the  $(i, j)$  entry is not zero. Because of these two properties, it is often a good idea to build the structural edge-node matrix and then compute the Laplacian matrix.

A final piece of easily computed information (with MATLAB) is the determinant of the Laplacian matrix after we have excluded the last row and column. This number will always be an integer that represents something both surprising and useful. A spanning tree of a graph is a set of edges that has two properties

1. given any node, it is possible to find a path from that node to any other node; and
2. given any node, it is **not** possible to find a path that loops back to that node.

This means that if you include any other edge of the graph you will be able to find a loop. It is fairly easy to show that for a graph with  $n$  nodes a spanning tree will always have exactly  $(n - 1)$  edges.

The surprising and useful result, called the Matrix-Tree Theorem, states that the determinant of the Laplacian matrix with the last row and column removed is exactly the number of possible spanning trees for the graph. In other words it is the number of ways that you can pick  $(n - 1)$  edges from the graph so that it is possible to find a path from any node to any other node.

**Exercise 6.** Add the two lines of code to the file WorkshopEx06.m that are needed to complete the calculation of the edge-node matrix.

**Exercise 7.** Run your modified version of WorkshopEx06.m on the files truss01.m, truss02.m, and truss03.m. The number of spanning trees for truss01.m and truss02.m should be 3. Depending on how creative your truss was, the number of spanning trees for truss03.m could be quite large and finding all of them could be a challenge. Take a stab at that challenge.

### 3 The Analysis Of Determinate Trusses

If we have a determinate truss, then we can analyze it by assuming that the forces on each node are balanced. The question is how do we describe this process so that a computer program can systematically set up the system of equations. We start by remembering that we will want two equations for each node, namely the sum of the horizontal forces and the sum of the vertical forces. Since we have  $n$  nodes, we will have  $2n$  equations in  $2n$  unknowns. The internal forces on each of the bars constitute  $nb$  of the unknowns. The remaining unknowns correspond to the reactive forces applied to the fixed nodes. Setting up these equations is similar, in many respects, to setting up the edge-node matrix. We walk through the bars and determine which equations each bar contributes to and add in that contribution.

Suppose that bar  $k$  connects node 1 with node 3. If the bar is under tension, then the bar exerts a force,  $F_k$ , pulling nodes 1 and 3 towards each other. If the coordinates of node 1 are  $(x_1, y_1)$  and the coordinates of node 3 are  $(x_3, y_3)$ , then we can calculate

$$c_k = \frac{x_3 - x_1}{\sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2}} \quad \text{and} \quad s_k = \frac{y_3 - y_1}{\sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2}}$$

We use the variable names  $c_k$  and  $s_k$  because these are actually the values of  $\cos(\theta)$  and  $\sin(\theta)$ , where  $\theta$  is the angle of bar  $k$  relative to the normal coordinate system centered at node 1. A practical point here is that we don't need to know  $\theta$ . Thus, the contribution of the internal force of this bar is  $+c_k F_k$  in the horizontal direction and  $+s_k F_k$  in the vertical direction for node 1.

When we consider node 3, the angle of bar  $k$ , relative to the normal coordinate system centered at node 3, is  $\theta + \pi$ . Since  $\sin(\theta + \pi) = -\sin(\theta)$  and  $\cos(\theta + \pi) = -\cos(\theta)$ , the contribution of the internal force of this bar is  $-c_k F_k$  in the horizontal direction and  $-s_k F_k$  in the vertical direction for node 3.

Consider the system of 8 equations arising from the determinate truss of Figure 1, and assume that we have vertical loads placed on nodes 3 and 4. We proceed from node to node first considering the sum of the horizontal forces and then considering the sum of the vertical forces. Since nodes 1 and 2 are fixed, we include the reactive forces  $H_1$ ,  $V_1$ ,  $H_2$ , and  $V_2$ . These are the forces that the foundation must provide in order to keep nodes 1 and 2 from moving.

$$\begin{aligned} F_1 \cdot 0 + H_1 &= 0 \\ F_1 \cdot 1 + V_1 &= 0 \\ F_2 \cdot c_2 + F_3 \cdot 0 + H_2 &= 0 \\ F_2 \cdot s_2 + F_3 \cdot 1 + V_2 &= 0 \\ -F_1 \cdot 0 - F_2 \cdot c_2 + F_4 \cdot 1 &= 0 \\ -F_1 \cdot 1 - F_2 \cdot s_2 + F_4 \cdot 0 &= -L_3 \\ -F_3 \cdot 0 - F_4 \cdot 1 &= 0 \\ -F_3 \cdot 1 - F_4 \cdot 0 &= -L_4 \end{aligned}$$

where  $c_2 = (0 - 3)/\sqrt{3^2 + 2^2} \approx -0.8321$  and  $s_2 = (2 - 0)/\sqrt{13} \approx 0.5547$ . For bars 1 and 3, which are vertical,  $c_k = 0$  and  $s_k = 1$ , and for bar 4, which is horizontal, we get  $c_k = 1$  and  $s_k = 0$ .

We can write this system of equations as the matrix equation

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & c_2 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & s_2 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -c_2 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & -s_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ H_1 \\ V_1 \\ H_2 \\ V_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -L_3 \\ 0 \\ -L_4 \end{bmatrix}$$

Let  $\mathbf{A}$  denote the  $8 \times 8$  matrix; let  $\mathbf{f}$  denote the vector of forces (internal and reactive); and let  $\mathbf{y}$  denote the vector of loads. This gives us the equation

$$\mathbf{A}\mathbf{f} = \mathbf{y}$$

Then once we have set up the matrix  $\mathbf{A}$  and the vector of loads,  $\mathbf{y}$ , we can easily solve for the forces.

$$\mathbf{f} = \mathbf{A}^{-1}\mathbf{y}$$

A relatively easy way to specify which nodes are fixed is to indicate which nodes have reactive forces. We can enter an array with  $n$  rows, one for each node, and two columns. In this case the first column indicates whether there is a horizontal reactive force and the second column indicates whether there is a vertical reactive force. This can be done with the following MATLAB code

```
fixed = [ 1  1
         1  1
         0  0
         0  0 ]
```

We can also specify the loads on different nodes in a similar fashion

```
loads = [ 0  0
          0  0
          0 -50
          0 -100 ]
```

The program that performs the analysis of the determinate truss now has all the information to insert into the matrix  $\mathbf{A}$  and the vector  $\mathbf{y}$ . The challenge is to put the correct numbers into the correct slots.

**Exercise 8.** Add the nine lines of code to the file WorkshopEx07.m that are needed to complete the calculation of the matrix  $\mathbf{A}$ . Run your modified version of WorkshopEx07.m on the file truss04.m. This includes the truss specification from truss01.m as well as the loading information.

**Exercise 9.** Modify your truss02.m file, call it truss05.m, so that it includes the following information. Node 1 is fixed both horizontally and vertically. Node 2 is only fixed vertically. Node 3 is not fixed but has a load of -100 pounds. Run your modified version of WorkshopEx07.m on the file truss05.m.

**Exercise 10.** Copy lines 74 through 80 of WorkshopEx08.m (or your modified version of WorkshopEx07.m) and adjust them so that the truss is redrawn with bars under tension plotted in blue and bars under compression plotted in red.

## 4 Hooke's Law and Potential Energy

Consider an undamped spring-mass system consisting of a single mass,  $m$ , and a single spring with spring constant,  $c$ . Newton's Second Law states that force equals mass times acceleration. If  $x(t)$  denotes the distance that the mass has been moved from its equilibrium point at time,  $t$ , then Newton's Second Law yields the differential equation

$$m \frac{d^2 x}{dt^2} = -cx \quad (2)$$

where  $-cx$  is spring force acting on the mass according to Hooke's Law. Since the velocity of the mass is  $v(t) = dx/dt$ , we can write Equation 1 as

$$m \frac{dv}{dt} = -cx. \quad (3)$$

If we multiply both sides by  $v$ , using  $v$  on the left and  $dx/dt$  on the right, we get

$$mv \frac{dv}{dt} = -cx \frac{dx}{dt}.$$

Integrating from 0 to  $T$  with respect to  $t$ , we get

$$m \int_0^T v \frac{dv}{dt} dt = -c \int_0^T x \frac{dx}{dt} dt,$$

and after applying the  $u$ -substitution rule we have

$$m \int_{v(0)}^{v(T)} v dv = -c \int_{x(0)}^{x(T)} x dx.$$

Since the two integrals are equal, we have obtained the conservation of energy property for the spring-mass system.

$$\frac{1}{2}mv^2(T) + \frac{1}{2}cx^2(T) = \frac{1}{2}mv^2(0) + \frac{1}{2}cx^2(0) \quad (4)$$

The first term is the kinetic energy of the moving mass and the second term is the potential energy (or work) stored in the elongated spring. The equation states the total energy at time,  $t = T$ , remains the same as the total energy at time  $t = 0$ . The important point in the context of a spring in equilibrium is that the potential energy of the spring, (that is, the amount of energy stored in the spring that can be converted into kinetic energy) is  $cx^2/2$ , where  $x$  is the amount that the spring has been elongated (stretched or compressed).

## 5 The Bungee Jumpers Bounce Back

Now consider the three bungee jumpers, Alex, Barry, and Carol, with masses ( $m_A$ ,  $m_B$ , and  $m_C$ ). At equilibrium they are suspended by three stretched cords, which have spring constants  $c_1 = 75$  lbs/ft and  $c_2 = c_3 = 30$  lbs/ft and which were initially 15 feet in length. Their positions below the bridge are  $z_A$ ,  $z_B$ , and  $z_C$ . Let  $x_1$ ,  $x_2$ , and  $x_3$  denote the displacements of the jumpers from the position they would be in if gravity had been turned off and the cords were not stretched. In other words  $x_1$ ,  $x_2$ , and  $x_3$  denote the distance each jumper has moved under the force of gravity. Then the elongation of each cord is given by

$$\begin{aligned}e_1 &= x_1 \\e_2 &= x_2 - x_1 \\e_3 &= x_3 - x_2\end{aligned}$$

This can be written as the matrix equation

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathbf{A}\mathbf{x}$$

The force acting on each spring is proportional to the spring constant times the elongation, so we can let  $\mathbf{y}$  denote the force acting on each spring and write

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} c_1 & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_3 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \mathbf{C}\mathbf{e}$$

The external forces,  $\mathbf{f}$ , on the masses are their weights, so

$$\mathbf{f} = \begin{bmatrix} gm_A \\ gm_B \\ gm_C \end{bmatrix} = \begin{bmatrix} w_A \\ w_B \\ w_C \end{bmatrix} = \begin{bmatrix} 225 \\ 160 \\ 105 \end{bmatrix}$$

Since this spring-mass system is in equilibrium, the forces acting on each node must balance.

$$\begin{aligned}f_1 - (y_1 - y_2) &= 0 \\f_2 - (y_2 - y_3) &= 0 \\f_3 - y_3 &= 0\end{aligned}$$

Writing this condition as a matrix equation we get

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \mathbf{A}^T\mathbf{y}$$

It is important to notice that the structural matrix,  $\mathbf{A}$ , relates the displacements,  $\mathbf{x}$ , to the elongations of each cord,  $\mathbf{e}$ , while its transpose,  $\mathbf{A}^T$  relates the forces acting on the cords to the external forces at the nodes.

The spring mass system can be summarized by the three equations

$$\mathbf{e} = \mathbf{A}\mathbf{x} \quad (5)$$

$$\mathbf{y} = \mathbf{C}\mathbf{e} \quad (6)$$

$$\mathbf{f} = \mathbf{A}^T\mathbf{y} \quad (7)$$

It is an easy matter to see that for the problem with three springs and three masses we can solve Equation 7 for  $\mathbf{y}$ , then solve Equation 6 for  $\mathbf{e}$ , and then finally solve Equation 5 for  $\mathbf{x}$ .

However, a more general approach is to determine the potential energy of the system. There are two components. The first component is the work needed to restore the masses to their original positions. It is

$$-(x_1f_1 + x_2f_2 + x_3f_3) = -\mathbf{x}^T\mathbf{f}.$$

The second component is the potential energy stored in the stretched cords. From Equation 4, this potential energy is given by

$$\begin{aligned} \frac{1}{2}c_1e_1^2 + \frac{1}{2}c_2e_2^2 + \frac{1}{2}c_3e_3^2 &= \frac{1}{2}\mathbf{e}^T\mathbf{C}\mathbf{e} \\ &= \frac{1}{2}\mathbf{x}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} \end{aligned}$$

Consequently the total potential energy of the system, as a function of the displacements, is

$$P(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{f} \quad (8)$$

A simple examination of the matrix  $\mathbf{A}$  shows that we can only have  $\mathbf{A}\mathbf{x} = \mathbf{0}$  when  $\mathbf{x} = \mathbf{0}$ . Since  $\mathbf{C}$  is a diagonal matrix with positive entries on the diagonal, it follows that

$$(\mathbf{A}\mathbf{x})^T\mathbf{C}(\mathbf{A}\mathbf{x}) > 0$$

as long as  $\mathbf{x} \neq \mathbf{0}$ . Since  $\mathbf{C}$  is a diagonal matrix, it is also the case that

$$(\mathbf{A}^T\mathbf{C}\mathbf{A})^T = \mathbf{A}^T\mathbf{C}\mathbf{A}$$

Consequently, the matrix  $\mathbf{A}^T\mathbf{C}\mathbf{A}$  is symmetric and positive definite. This means that the Potential Energy function,  $P(\mathbf{x})$ , is minimized when  $\mathbf{x}$  satisfies

$$\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} = \mathbf{f} \quad (9)$$

This minimization argument is worth a brief overview. Suppose that  $\mathbf{u}$  is any vector, and let  $\mathbf{v} = \mathbf{u} - \mathbf{x}$  then

$$\mathbf{u} = \mathbf{v} + \mathbf{x}$$

and

$$\begin{aligned} P(\mathbf{u}) &= \frac{1}{2}\mathbf{u}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{u} - \mathbf{u}^T\mathbf{f} \\ &= \frac{1}{2}(\mathbf{v} + \mathbf{x})^T\mathbf{A}^T\mathbf{C}\mathbf{A}(\mathbf{v} + \mathbf{x}) - (\mathbf{v} + \mathbf{x})^T\mathbf{f} \\ &= \frac{1}{2}\mathbf{v}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{v} + \frac{1}{2}\mathbf{v}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{v} + \frac{1}{2}\mathbf{x}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} - \mathbf{v}^T\mathbf{f} - \mathbf{x}^T\mathbf{f} \\ &= \frac{1}{2}\mathbf{v}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{v} + \mathbf{v}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} - \mathbf{v}^T\mathbf{f} + \frac{1}{2}\mathbf{x}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{f} \\ &= \frac{1}{2}\mathbf{v}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{v} + \mathbf{v}^T(\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} - \mathbf{f}) + P(\mathbf{x}) \end{aligned}$$

Since  $\mathbf{x}$  satisfies the Equation 9,  $\mathbf{A}^T \mathbf{C} \mathbf{A} \mathbf{x} - \mathbf{f} = 0$ , and we get

$$P(\mathbf{u}) = \frac{1}{2} \mathbf{v}^T \mathbf{A}^T \mathbf{C} \mathbf{A} \mathbf{v} + P(\mathbf{x})$$

If we let  $\mathbf{w} = \mathbf{A} \mathbf{v}$ , then

$$\mathbf{v}^T \mathbf{A}^T \mathbf{C} \mathbf{A} \mathbf{v} = \mathbf{w}^T \mathbf{C} \mathbf{w} = c_1 w_1^2 + c_2 w_2^2 + c_3 w_3^2 + c_4 w_4^2$$

Since all the  $c_i$ 's are positive this is a positive number, and we can conclude that  $P(\mathbf{x})$  must be the minimum. In other words, solving Equation 9 is the same as finding the displacements of the bungee jumpers which minimize the potential energy of the system.

**Exercise 11.** The file WorkshopEx09.m finds the equilibrium positions of the bungee jumpers by minimizing the potential energy of the system. Run it and verify that it arrives at the same positions for the jumpers that we obtained earlier by balancing the forces.

**Exercise 12.** Run WorkshopEx09.m so that  $\mathbf{K}$ , the stiffness matrix, is available. Using displacement vectors  $\mathbf{u}$  of your own choosing, calculate the potential energy associated with that displacement using the statement

$$p = 0.5 * \mathbf{u}' * \mathbf{K} * \mathbf{x} - \mathbf{x}' * \mathbf{w}$$

## 6 Indeterminate Trusses

The Analysis of Determinate Trusses is based on the premise that the sum of the forces is zero at each node. After we have decomposed the sum at each node into the sum of the horizontal components of the forces and the sum of the vertical components of the forces we have two equations at each of the  $n$  nodes. This is exactly the number we need if we have a determinate truss, but this approach will no longer work if our truss includes additional bars for extra strength. In this section we will develop an analysis based on the assumption that the bars will be compressed or stretched slightly. Then the internal force of the bar will be proportional to the elongation of the bar (the amount that it is compressed or stretched). We start by developing a structural matrix which will give us an approximation to these elongations given small displacements in the nodes.

### 6.1 Approximate Displacements

Suppose that we have a bar connecting nodes 1 and 2 with node 1 at  $(x_1, y_1)$  and node 2 at  $(x_2, y_2)$ . We assume that the bar is very strong, but that under a load it gets displaced slightly so that nodes 1 and 2 are now at  $(x_1 + \Delta x_1, y_1 + \Delta y_1)$  and  $(x_2 + \Delta x_2, y_2 + \Delta y_2)$ . Since the internal force of the bar is determined by how much the bar is stretched or compressed, we must determine a formula for the new length of the bar. The length of the bar before it is displaced is simply

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The length of the bar after it is displaced is

$$\begin{aligned}\bar{L} &= \sqrt{[(x_2 + \Delta x_2) - (x_1 + \Delta x_1)]^2 + [(y_2 + \Delta y_2) - (y_1 + \Delta y_1)]^2} \\ &= \sqrt{[(x_2 - x_1) + (\Delta x_2 - \Delta x_1)]^2 + [(y_2 - y_1) + (\Delta y_2 - \Delta y_1)]^2}\end{aligned}$$

For a moment let's consider what happens if  $\Delta x_1$  is nonzero and the other three displacements are zero. Then we have

$$\begin{aligned}\bar{L} &= \sqrt{[(x_2 - x_1) + (-\Delta x_1)]^2 + (y_2 - y_1)^2} \\ &= \sqrt{(x_2 - x_1)^2 - 2(x_2 - x_1)\Delta x_1 + \Delta x_1^2 + (y_2 - y_1)^2} \\ &= \sqrt{L^2 - 2(x_2 - x_1)\Delta x_1 + \Delta x_1^2} \\ &= \sqrt{L^2 - 2L \frac{(x_2 - x_1)}{L} \Delta x_1 + \Delta x_1^2} \\ &= \sqrt{L^2 - 2L \cos(\theta) \Delta x_1 + \Delta x_1^2}\end{aligned}$$

where  $\theta$  is the angle the bar originally made with the  $x$ -axis and  $\cos(\theta) = (x_2 - x_1)/L$ .

We can approximate this function of the displacement,  $\bar{L}(\Delta x_1)$ , with its first order Taylor polynomial. Recall that Taylor's Theorem states that the function  $f(x + \Delta x)$  is equal to

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + f''(x + \zeta) \frac{(\Delta x)^2}{2}$$

where  $|\zeta| \leq |\Delta x|$ . Since we are assuming that the displacements are very small, the term involving  $\Delta x_1^2$  can be neglected and we can approximate  $\bar{L}(\Delta x_1)$  by its Taylor polynomial of degree 1 (more commonly referred to as the tangent line).

$$\begin{aligned}\bar{L}(\Delta x_1) &\approx L - \frac{1}{2L} (2L \cos(\theta)) \Delta x_1 \\ &\approx L - \cos(\theta) \Delta x_1\end{aligned}$$

If we repeat this process for the displacements  $\Delta x_2$ ,  $\Delta y_1$ , and  $\Delta y_2$ , we can conclude that

$$\bar{L} \approx L - \cos(\theta) \Delta x_1 + \cos(\theta) \Delta x_2 - \sin(\theta) \Delta y_1 + \sin(\theta) \Delta y_2$$

Let  $e$  denote the elongation of the bar connecting nodes 1 and 2. Then we have

$$e = \bar{L} - L \approx -\cos(\theta) \Delta x_1 + \cos(\theta) \Delta x_2 - \sin(\theta) \Delta y_1 + \sin(\theta) \Delta y_2$$

Suppose that we have  $n$  nodes and  $m$  bars in our truss, then we can construct the following structural matrix,  $\mathbf{A}_0$ . This matrix will have  $m$  rows, one for each bar, and it will have  $2n$  columns, two for each node, the first column in the pair corresponding to the horizontal displacement and the second corresponding to the vertical displacement. This is similar to the edge-node matrix, except that each node now has two columns.

For each bar we will compute  $c_k = \cos(\theta_k)$  and  $s_k = \sin(\theta_k)$ , where  $\theta_k$  is the angle bar  $k$  makes with the  $x$ -axis. Of course, we really don't need to determine  $\theta_k$  or compute any trig functions. For  $c_k$  we simply compute the difference in the  $x$ -coordinates of the two nodes connected by the bar and divide that number by the length of the bar. We compute  $s_k$  in a similar way using the  $y$ -coordinates of the two nodes connected by the bar. This is the same calculation that we did in determining the force acting on the bar in the case of the determinate truss.

Let  $\mathbf{u}_0$  denote the vector of displacements with  $u_1 = \Delta x_1$ ,  $u_2 = \Delta y_1$ ,  $u_3 = \Delta x_2$ ,  $u_4 = \Delta y_2$ , etc. Similarly, let  $\mathbf{e}$  denote the vector of elongations. Then for small displacements the vector of elongations can be well approximated by the equation

$$\mathbf{e} = \mathbf{A}_0 \mathbf{u}_0$$

For the trusses in Figure 1, the coordinates of the nodes are  $\{(0, 0), (3, 0), (0, 2), (3, 2)\}$ . For the first truss in Figure 1, bar 1 connects node 1, with  $(x_1, y_1) = (0, 0)$ , to node 3, with  $(x_3, y_3) = (0, 2)$ . This gives us  $c_1 = 0$  and  $s_1 = 1$ . Bar 2 connects node 2, with  $(x_2, y_2) = (3, 0)$ , to node 3, with  $(x_3, y_3) = (0, 2)$ . This gives us

$$c_2 = \frac{x_3 - x_2}{\sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}} = \frac{-3}{\sqrt{13}} \quad \text{and} \quad s_2 = \frac{y_3 - y_2}{\sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}} = \frac{2}{\sqrt{13}}.$$

Geometrically,  $c_2 = \cos \theta$  and  $s_2 = \sin \theta$  where  $\tan \theta = -2/3$ .

After calculating  $c_k$  and  $s_k$  for each bar, and noting that  $c_1 = 0$ ,  $s_1 = 1$ ,  $c_3 = 1$ ,  $s_3 = 0$ ,  $c_4 = 0$ , and  $s_4 = 1$ , we construct the following preliminary structural matrix

$$\mathbf{A}_0 = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -c_2 & -s_2 & c_2 & s_2 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \end{bmatrix}.$$

For the second truss in Figure 1 we get the following structural matrix, which is the previous matrix with one additional row to account for the bar connecting node 1 with node 4. For this fifth bar we have

$$c_5 = \frac{x_4 - x_1}{\sqrt{(x_4 - x_1)^2 + (y_4 - y_1)^2}} = \frac{3}{\sqrt{13}} \quad \text{and} \quad s_5 = \frac{y_4 - y_1}{\sqrt{(x_4 - x_1)^2 + (y_4 - y_1)^2}} = \frac{2}{\sqrt{13}}.$$

Geometrically,  $c_5 = \cos \phi$  and  $s_5 = \sin \phi$  where  $\tan \phi = +2/3$ .

$$\mathbf{A}_0 = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -c_2 & -s_2 & c_2 & s_2 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ -c_5 & -s_5 & 0 & 0 & 0 & 0 & c_5 & s_5 \end{bmatrix}.$$

In Figure 1, nodes 1 and 2 are fixed, which means that their displacements,  $u_1 = \Delta x_1$ ,  $u_2 = \Delta y_1$ ,  $u_3 = \Delta x_2$ , and  $u_4 = \Delta y_2$ , can only be zero. This means that the first four columns of these two matrices can not contribute to the calculation of  $\mathbf{e}$ , the vector of elongations. Consequently, we can write the matrix equations as

$$\mathbf{e} = \mathbf{A}\mathbf{u} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ c_2 & s_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 \end{bmatrix} \mathbf{u}$$

and

$$\mathbf{e} = \mathbf{A}\mathbf{u} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ c_2 & s_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & c_5 & s_5 \end{bmatrix} \mathbf{u}$$

where  $\mathbf{u}$  is the vector that only contains the displacements corresponding to nodes 3 and 4 — the only possible nonzero displacements. One important thing to notice is that for the determinate truss, the first truss in Figure 1, the matrix  $\mathbf{A}$  is square with 4 rows and 4 columns. However, for the second truss, the matrix  $\mathbf{A}$  has 5 rows and 4 columns. This reflects the fact that this truss distributes its load over 5 bars. If the loaded truss is in equilibrium, then the displacements of the nodes will be such that the potential energy of the truss is minimized.

## 6.2 The Potential Energy For A Truss

Earlier we determined the potential energy for the system with the three bungee jumpers. The same analysis applies to trusses. The only difference is that the structural matrix,  $\mathbf{A}$ , is more complicated because it also reflects the two dimensional geometry of a truss. However,  $\mathbf{u}$  is the vector of possible displacements and  $\mathbf{e}$  is the vector of elongations. These two quantities are related by the structural matrix

$$\mathbf{e} = \mathbf{A}\mathbf{u}$$

The internal forces acting on the bars are then given by

$$\mathbf{y} = \mathbf{C}\mathbf{e}$$

where  $\mathbf{C}$  is the diagonal matrix of material constants for the bars. Finally, the external forces,  $\mathbf{f}$ , (the loads on the nodes) must balance the internal forces so we have

$$\mathbf{f} = \mathbf{A}^T\mathbf{y}$$

The work to restore the bars to their original positions is  $-\mathbf{u}^T\mathbf{f}$  and the potential energy stored in the elongated bars is  $(1/2)\mathbf{e}^T\mathbf{C}\mathbf{e}$ . So in equilibrium, the truss will minimize the total potential energy

$$P(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{u} - \mathbf{u}^T\mathbf{f} \quad (10)$$

and as we saw before, this function is minimized when  $\mathbf{u}$  satisfies the matrix equation

$$\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{u} = \mathbf{f} \quad (11)$$

Thus given the external forces, we can find the displacements of the nodes that are allowed to move. From these displacements we can calculate the forces acting on the bars using the equation.

$$\mathbf{y} = \mathbf{C}\mathbf{A}\mathbf{u} \quad (12)$$

Finally we can compute the vector,  $\mathbf{f}_0$ , of all external forces, including the reactive forces from

$$\mathbf{f}_0 = \mathbf{A}_0^T\mathbf{y} \quad (13)$$

**Exercise 13.** Add the nine lines of code to the file WorkshopEx10.m that are needed to complete the calculation of the matrix  $\mathbf{A}_0$ . Run your modified version of WorkshopEx10.m on the file truss06.m. This includes the truss specification from truss01.m as well as the loading information and Hooke's constants for the bars.

**Exercise 14.** Modify your truss05.m file, call it truss07.m, to include Hooke's constants for the bars and run your modified WorkshopEx10.m on truss07.m

**Exercise 15.** Modify your truss03.m file, call it truss08.m, to include loading information and Hooke's constants for the bars and run your modified WorkshopEx10.m on truss08.m



### 6.3 Project Assignment

1. Sketch a planar truss diagram for the near side of the Truss Bridge in the picture.
2. Determine appropriate dimensions for the bridge and use them to define the coordinates for your truss.
3. Determine appropriate materials for the bars of your truss and determine their approximate material constants.
4. Prepare a descriptor file that describes the truss.
5. Modify the BuildTruss3Load program so that it solves for the displacements given three applied loads, calculates and prints out the internal force on each bar, and redraws the truss diagram so that the bars are color coded. The bars which are under compression should be drawn in blue, the bars which are under tension should be drawn in red, and any bar for which the internal force is less than 0.001 of the total applied load should be drawn in green.
6. Consider an object crossing the bridge. First decide on whether you want to consider a human, a horse, or a car, and pick an appropriate mass. Then decide on three positions for the mass and determine appropriate applied loads. Finally analyze each of these three situations using your program.
7. Turn in a report which includes discussions of all the decisions made above, the color coded diagrams, the results of your analysis, and any relevant conclusions.