Determining for the forces in the members of a structure often requires the solution to a system of coupled linear equations. These problems naturally lend themselves to matrix solutions of the form

\[ [A] \{x\} = \{b\} \]  

where the system matrix \([A]\) only depends on the geometric layout of the structure, and the right-hand-side vector \(\{b\}\) contains the forcing; \(\{x\}\) is the solution vector for the forces in the members of the structure. This laboratory applies matrix methods to the solution of a statically determinant truss. Section 1 introduces the truss geometry and few review points from statics. Section 2 details the requirements of the programming assignment. The contents of the memorandum are detailed in Section 3, followed by the challenge problem in Section 4.

1 Trusses and Their Naming Convention

This assignment will apply matrix methods to find the reactions at the supports and the forces in the members of the K-truss shown in Figure 1. The truss is supported by a fixed, hinged support at the left side attached to node 1 and a sliding, hinged support at the right side attached to node \(n\), where \(n\) is the total number of nodes in the truss. The truss is symmetric about the center plane. Each of the triangular elements attached to the supports have a base length \(l_b\) and height \(h\). Each of the K-elements have a width \(l_k\) and height \(h\); the central nodes of each K-element are located at a height \(h/2\).

In trusses with only a few members, it is quite easy to build the system and right-hand-side matrices, and no naming conventions are used. For problems such as those in Figure 1, however, we must introduce a naming convention to specify the order of the rows and column of \([A]\).

The first step to the naming convention is the node numbering system. Node 1 will be the left-most node, truss connections are numbered from left to right and top to bottom, and node \(n\) is the right-most node.

The unknown forces we would like to solve for are the reactions at the supports \((V_1, H_1, \text{ and } V_n)\) and the forces in each of the members of the truss \(F_{ij}\). In this notation, the subscript \(i\) is the
lower node number and $j$ is the higher node number at the connections of the member to which $F_i^j$ belongs (see for example $F_1^2$ in the figure). Next, we must specify how these unknown values are ordered in the vector of unknowns $\{x\}$. For this assignment, the first two unknowns are $V_1$ and $H_1$ and the last unknown is $V_n$. The other unknowns are sorted first by the subscript $i$ and then by the subscript $j$, so that the unknown vector would be

$$\{x\} = \begin{bmatrix} V_1 \\ H_1 \\ F_1^1 \\ F_1^2 \\ F_2^1 \\ F_2^2 \\ \vdots \\ F_{n-1}^{n-2} \\ F_n^{n-1} \\ V_n \end{bmatrix}$$

For the truss shown in Figure 1, there would be 32 unknowns in the $\{x\}$ vector.

In your basic statics class, you learned to solve truss problems using the method of nodes. In this method, we apply $\sum F_x = 0$ and $\sum F_y = 0$ to the free-body diagram of each node. Since there are 16 nodes in Figure 1, we would get 32 equations; thus, this truss is statically determinant.

The last step of the naming convention is to specify the row-order of the system matrix $[A]$. Note that the column order is fixed by the definition of $\{x\}$. Here, we will apply the statics equations to each node in ascending order of node number, yielding two equations for each node. Thus, row 1 is the $\sum F_x = 0$ at node 1, row 2 is the $\sum F_y = 0$ at node 1, row 3 is the $\sum F_x = 0$ at node 2, etc. This convention also specifies the order of the rows of the right-hand-side vector $\{b\}$. 
2 Programming Assignment

For this assignment you will solve the truss in Figure 1 under different loading conditions. You should write a Matlab program with additional user-specified functions that does the following

1. Create variables for the dimensions of the truss and the loading (see the next section for the values to use here).

2. Call a user-defined function k_truss to build the system matrix \([A]\). The function call should be

\[
[A] = \text{k\_truss}(h, lt, lk);
\]

- You will need to create the function \text{k\_truss} which will fill in the \([A]\) matrix. Note that this function only needs information on the geometry, and has nothing to do with the loading.
- Since the \([A]\) matrix is very large (32x32), you should make use of the : operator to enter small segments of the \([A]\)-matrix at a time. For instance, you can enter row 1 using the following command

\[
A(1,1:4) = [0 1 \cos(a1) 1];
\]

3. Call another user-defined function \text{k\_loads} to build the right-hand-side vector \([b]\). The function call should be

\[
[b] = \text{k\_loads}(f\_node, F, m)
\]

where \(f\_node\) is the node number where the force is applied, \(F\) is a row vector specifying the force as \(F = (F_x, F_y)\), and \(m\) is the number of nodes in the truss. You will need to create this function; be careful when deciding the sign of \(F\) in the \([b]\) matrix.

4. Solve the system of equations using Gauss elimination with partial pivoting. A sample code for this method is provided in the Chapra textbook in Figure 9.5. Write a function containing this code, but do not turn it in.

5. Search through the solution vector to identify the maximum tension and maximum compression in the truss.

6. Display the values for the reactions at the supports and the maximum tension and compression. Identify the members (i.e. \(F^9_g\)) where the maximum tension and compression occur.
3 Assignment

You should run your code for the following forcing cases:

1. Force of $F = (0, -3000)$ kN applied at node 8.
2. Force of $F = (0, -3000)$ kN applied at node 9.
3. Force of $F = (-1500, 1200)$ kN applied at node 14.
4. Force of $F = (0, -1500)$ kN applied simultaneously at nodes 4, 7, 9, 12, and 15.

using the geometry $h = 4$ m, $l_t = 2$ m, and $l_k = 2$ m. Check your code by comparing the output
for the reactions at the supports to values you obtain by hand calculation for each of these cases.

Your memorandum should summarize the results of your computations, including the comparisons to the hand calculations. Include text describing what you observe, and report your numerical solutions for the reaction forces and the maximum tension and compression for each of the above loading cases in a table of results.

In the last paragraph of the memorandum, answer the following question: In the second-to-last line of the GaussPivot function in the Chapra textbook, the code computes:

$$x(i)=(Aug(i, nb)-Aug(i, i+1:n)*x(i+1:n))/Aug(i, 1);$$

Why does the calculation $Aug(i, i+1:n)*x(i+1:n)$ work without using the .* method?

The appendix should contain a listing of your program code and the two user-defined functions k_truss and k_loads. Be sure to include prose in the appendix introducing each program listing.

To submit the program to the TA, copy your functions k_truss and k_loads to the end of your program file and submit the combined file to the TA. Note that this file will not run in Matlab since you cannot define a function within a program.

4 Challenge Problem

Choose from one of the following challenge problems:

1. Add the programming elements necessary to automatically plot the truss, using red for the elements in maximum tension, blue for the elements in maximum compression, and black for all other elements. Plot the nodes using open circles. To accomplish this task, I suggest the following:

   - Add two more outputs to the k_truss function. The first output is a vector conx specifying the connections for each member in the truss. The rows of conx are in the same order as the \{x\} vector; the columns report the i and j index of the member. The second output is a vector coord specifying the x- and y-coordinates of each of the nodes. The row number corresponds to the node number and the columns contain coordinates, for instance, row $k$ contains $(x_k, y_k)$.

   - Use the new variables defined above in a FOR loop to plot each member one-at-a-time. First, plot the whole truss in black. Then go back and re-plot the members in maximum tension and compression in their respective colors.
2. Create the k.truss program in a general way so that the user can specify the number of left and right K-elements in the truss. Test your program for a truss having four K-elements on each side of the center. To get full credit, your code must be able to solve any truss with two or more K-elements.