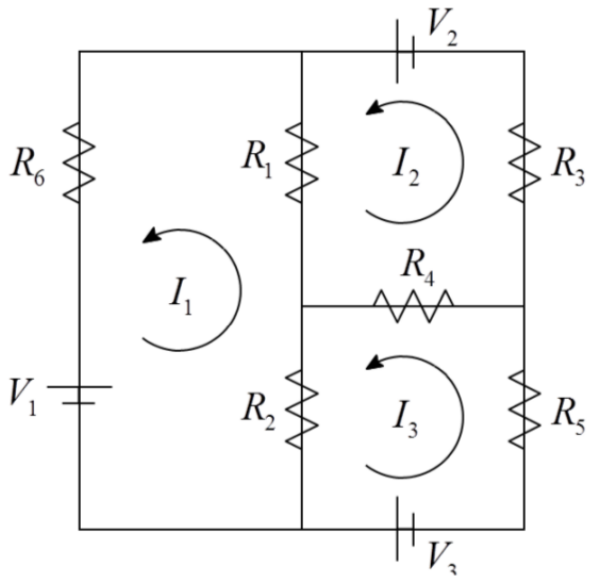
AMATH 301 - Spring 2018

Homework #3

Due on Thursday, April 19, 2018

1. Consider the circuit diagram below:



Here, each V represents a change in voltage (measured in volts) at a battery, each R represents a resistance (measured in ohms) at a resistor, and each I represents a current (measured in amps) through a wire. These quantities obey two simple laws:

(1) **Ohm's Law:** The voltage drop across a resistor is $V = IR$,

(2) **Kirchhoff's Second Law:** The sum of all the voltage drops in a closed loop is zero.

Using these two laws, we can construct the following systems of equations:

$$
\begin{aligned}
R_6 I_1 + R_1 (I_1 - I_2) + R_2 (I_1 - I_3) &= V_1, \\
R_3 I_2 + R_4 (I_2 - I_3) + R_1 (I_2 - I_1) &= V_2, \\
R_5 I_3 + R_4 (I_3 - I_2) + R_2 (I_3 - I_1) &= V_3.
\end{aligned}
$$

Let the resistances be given by $R_1 = 10\Omega$, $R_2 = 20\Omega$, $R_3 = 5\Omega$, $R_4 = 15\Omega$, $R_5 = 30\Omega$ and $R_6 = 25\Omega$. We want to calculate the currents I_1, I_2, and I_3.

(a) Write the equations in matrix form $A\mathbf{x} = \mathbf{b}$, where $\mathbf{x}$ is a 3×1 vector of the unknown current values (you need to do this by hand, not in Matlab). Using the `lu` command in Matlab, find matrices L, U and P such that $PA = LU$. Calculate the product UPL and save the resulting 3×3 matrix in `A1.dat`. (Notice that I haven't told you the voltages yet. Does that matter?)

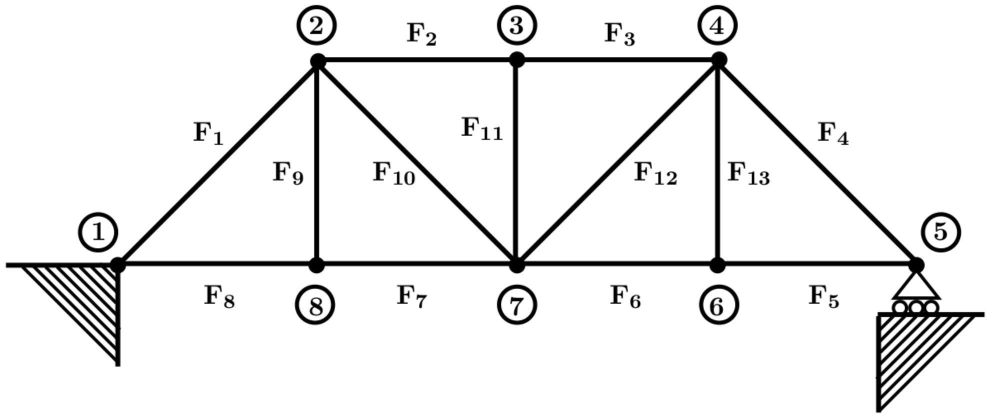
(b) Let $V_1 = 50$ and $V_2 = 0$. For every value of V_3 from 1 to 100 in increments of 1, calculate I_1, I_2 and I_3 using LU decomposition. Save the resulting values of I_2 as a 1×100 row vector in `A2.dat`.

(c) Repeat part (b), but solve each system using the `inv` command (i.e. using the inverse of A) instead of LU decomposition. This method should be slower, and the result should be slightly different. Similar to part (b), make 1×100 row vector of the values of I_2. Subtract your vector from part (c) from the vector from part (b), and save the difference as `A3.dat`. For example, if `x_b` is the name of your vector from part (b) and `x_c` is the name of your vector from part (c), then the vector you should save in `A3.dat` is `x_b - x_c`.

Things to think about: The `inv` command is not very useful in practical applications. In fact, Matlab probably warns you not to use it in your code. However, you probably saw in this code that it does not make much of a difference to the final answer. Can you find a system $A\mathbf{x} = \mathbf{b}$ where the `inv` command causes a lot of rounding error? Can you find a system where the `inv` command is visibly slower than backslash?

As we learned in class, the backslash command uses LU decomposition for most systems. Why was it better to find L, U, and P ourselves before doing part (b)?

2. Consider the bridge truss diagrammed below.



Given a vector of external forces $\mathbf{b}$ on the bridge, we can compute the forces $\mathbf{F_1}$, $\mathbf{F_2}$,...,$\mathbf{F_{13}}$ by solving the system $A\mathbf{x} = \mathbf{b}$, where $\mathbf{x}$ is a vector of unknown forces and A is given by

$$A = \begin{bmatrix} -s & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & s & 0 & 0 & 0 \\ -s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -s & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -s & 0 \\ 0 & 0 & 0 & -s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -s & -1 \\ 0 & 0 & 0 & -s & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -s & 0 & s & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & s & 1 & s & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and $s = \sqrt{2}/2$. We will solve for the vector of forces $\mathbf{x}$ assuming that there are three vehicles at positions 6, 7 and 8 with weights 5 tons, 10 tons, and 4 tons, respectively. This means that $\mathbf{b} = [0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 10, 0, 5]^T$. (Note that the T means transpose so $\mathbf{b}$ is a 13×1 column vector with those entries.)

(a) Solve for $\mathbf{x}$ using LU decomposition. Save the intermediate answer $\mathbf{y}$ in `A4.dat` and the final answer $\mathbf{x}$ in `A5.dat`.

(b) Solve for $\mathbf{x}$ using the backslash command. Save your answer as `A6.dat`. How does this compare with your answer from part (a)?

(c) Now suppose that we add weight to the truck at position 6 (which corresponds to the 13th entry of $\mathbf{b}$) in increments of 0.01 tons until the bridge collapses. Each bridge member can withstand 30 tons of compression or tension (i.e., positive or negative forces.) Therefore, the bridge will collapse when the absolute value of the largest force is larger than 30. Find the smallest weight of the truck at position 6 for which the bridge collapses. Save your answer as `A7.dat`. **Hint:** You may find some of the functions `max`, `abs`, or `norm` useful.

3. Consider the matrix

$$A = \begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix}.$$

Find the condition number of A and save it in `A8.dat`. You should find that the condition number is not particularly large, which suggests that any reasonable algorithm should work well with this matrix. It is easy to check by hand that an LU decomposition of A is

$$L = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{pmatrix}$$

Multiply LU by hand and confirm that $A = LU$. Now use Matlab to multiply LU and save your answer in `A9.dat`. Is this close to A?

If we switch the rows of A, we get a new matrix

$$B = \begin{pmatrix} 1 & 1 \\ 10^{-20} & 1 \end{pmatrix}.$$

An LU decomposition of B is

$$L = \begin{pmatrix} 1 & 0 \\ 10^{-20} & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} 1 & 1 \\ 0 & 1 - 10^{-20} \end{pmatrix}.$$

Multiply LU by hand and confirm that $B = LU$. Now use Matlab to multiply LU and save your answer in `A10.dat`. Is this close to B?

Things to think about: Why do you think one of these answers is so inaccurate? (This requires some knowledge about how floating point numbers are rounded.) What happens if you try to use the L and U matrices from above to solve $A\mathbf{x} = \mathbf{b}$? In particular, try $\mathbf{b} = [1 + 10^{-20}, 2]^T$. The solution to that system is $\mathbf{x} = [1, 1]^T$.) Does the same issue arise with the matrix B? Try using the `lu` command to do the LU decomposition of A. Does Matlab give the same LU decomposition as above? How does Matlab's LU decomposition of A compare with the LU decomposition of B above?