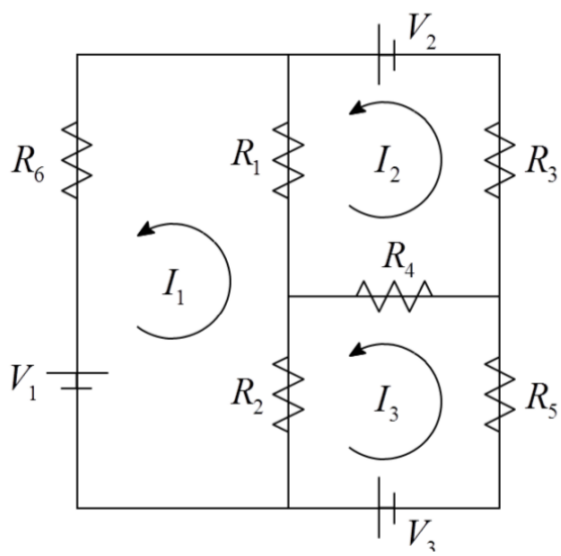
Homework 3

Problem 1: Circuit Diagrams

Consider the circuit diagram below:



Here, each V represents a change in voltage (in volts) at a battery, each R represents a resistance (in ohms) at a resistor and each I represents a current (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 changes in a closed loop is zero.

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

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

Suppose that the resistances are given by $R_1 = 15\Omega$, $R_2 = 20\Omega$, $R_3 = 6\Omega$, $R_4 = 18\Omega$, $R_5 = 25\Omega$ and $R_6 = 30\Omega$ and that we are trying to calculate the currents I_1, I_2 and I_3.

(a) Write the equations in the matrix form $A\mathbf{x} = \mathbf{b}$, where $\mathbf{x}$ is a 3×1 vector of (unknown) currents. (You need to do this by hand, not in Matlab.) Using the `lu` command in Matlab, find matrices L, U and P such that $LU = PA$. 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) Now let $V_1 = 50V$ and $V_3 = 75V$. For every value of V_2 from $1V$ to $100V$ (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`.

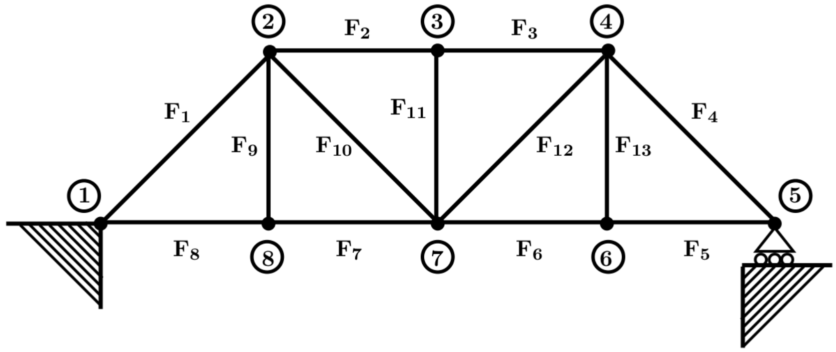
(c) Repeat part (b), but solve each system using the `inv` command (i.e., using the inverse of A) instead of L, U and P. This method should be slightly slower and the answers should be slightly different. Save the resulting values of I_1 as a 1×100 row vector in `A3.dat`.

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 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)?

Problem 2: Bridge Safety

Consider the bridge truss diagrammed below:



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

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

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

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

(b) Now solve for $\mathbf{x}$ using the backslash command directly (i.e., without using L, U or P). Save your answer in `A6.dat`.

(c) Now suppose that we add weight to the truck in position 8 (which corresponds to the 9th 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) before breaking. This means that 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 8 for which the bridge collapses, and save this weight as `A7.dat`. (When the truck has this weight, one of the forces on the bridge should be larger than 30, and if the truck weighs 0.01 tons less then none of the forces on the bridge should be larger than 30.) **Hint:** You may find some combination of the functions `max`, `abs` or `norm` useful.

Things to think about: In a previous quarter, I made all three trucks weigh 5 tons and had you add weight to the middle truck instead. Many people were off by 0.01 for `A7.dat`. What happened?

Problem 3: Pivoting

Consider the matrix
$$A = \begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix}.$$
Find the condition number of A (using `cond`) 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 the LU decomposition of A (without pivoting) is
$$L = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix} \text{ and } U = \begin{pmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{pmatrix}.$$
Multiply LU by hand and confirm that $LU = A$. 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
$$B = \begin{pmatrix} 1 & 1 \\ 10^{-20} & 1 \end{pmatrix}.$$
It is easy to check by hand that the LU decomposition of B (without pivoting) is
$$L = \begin{pmatrix} 1 & 0 \\ 10^{-20} & 1 \end{pmatrix} \text{ and } U = \begin{pmatrix} 1 & 1 \\ 0 & 1 - 10^{-20} \end{pmatrix}.$$

Multiply LU by hand and confirm that $LU = B$. 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 of 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 with A and B. What does Matlab return?