Assignment #3

This assignment is due on Monday, November 6th 23:59:59 class via email to christian.wallraven+AMF2023@gmail.com.

Important: You need to name your file properly. If you do not adhere to this naming convention, I may not be able to properly grade you!!!

If you are done with the assignment, make one zip-file of the assignment1 directory and call this zip-file `STUDENTID_A3.zip` (e.g.: `2016010001_A3.zip`). The correctness of the IDs matters! **Please double-check that the name of the file is correct!!**

Also: Please make sure to comment all code, so that I can understand what it does. Uncommented code will reduce your points!

Finally: please read the assignment text carefully and make sure to implement EVERYTHING that is written here – if you forget to address something I wrote, this will also reduce your points! Precision is key ☺!

Part1 Gaussian Elimination (50 points):

Make an ipython notebook `gauss_elimination.ipynb`.

a) Implement a function `GaussSolve` that solves a linear system of equations Ax=b for a square matrix A using forward elimination, backward substitution, **and** partial pivoting as shown during class.

The function should accept two inputs (A and b) and return one output (x).

The function also needs to include error handling to

- check whether the matrix A is square or not
- whether the dimensions of b and A fit
- whether during the forward elimination step any of the leading coefficients after partial pivoting become 0

If any conditions become critical, then the function should abort, telling the user the reason for it.

b) Which part of the code (forward elimination or backward substitution) takes longest to run? Use both the `timeit` function and an analysis of the complexity of the implementation to explain and show why this is!

c) Next, use the `timeit` function to compare how fast your GaussSolve function is compared to `numpy.linalg.solve` for **random** matrices (numpy.randn) and vectors for the following sizes:

n=1,10,20,50,100,500,1000.

Run each matrix size 20 times, and plot the results in a nice plot, comparing your function and numpy's function.

Answer the following questions in your comments:

How much faster is the built-in Matlab function than your function for each

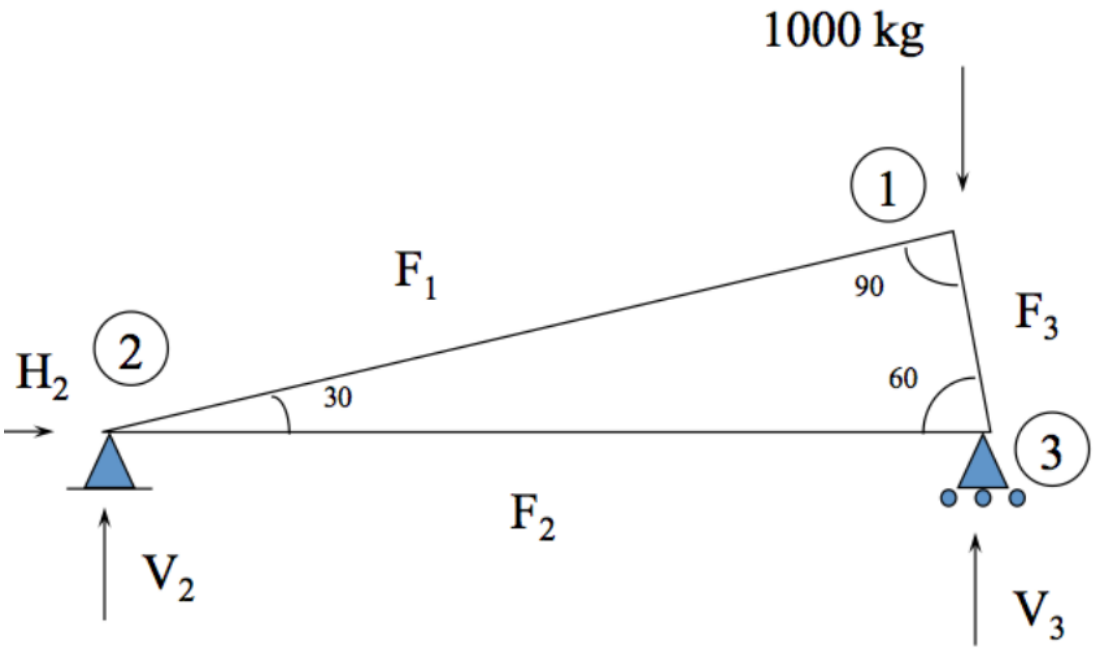
time step?

Why do you think it is faster?

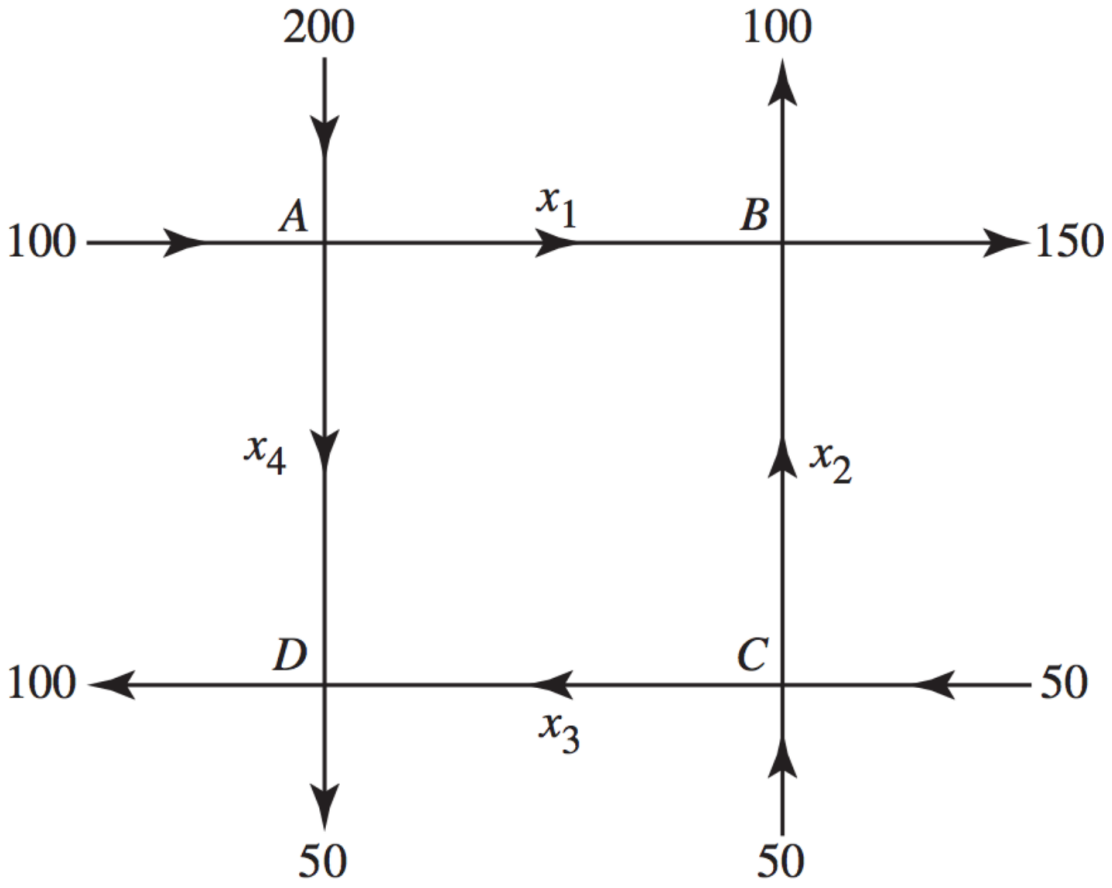
Part2 Gaussian Elimination - applications (10 points):

Insert the necessary code and discussion into `gauss_elimination.ipynb`.

a) Use your code to solve the simple truss problem from class to determine the six unknown forces F1,F2,F3,H2,V2,V3. See the class material for the layout of the problem and for the matrix. You can use "kg" as the force unit (yes, you are officially allowed, but only for this assignment ^^).



b) Use your code to solve a traffic flow problem. In the picture below, you see a cutout of a city with different streets and junctions. Since you have traffic cameras positioned at the exits of the block, you can measure how many cars enter and exit the block. These numbers are given next to the arrows and all streets are ONE-WAY streets. Your task is to find out the missing numbers x1, x2, x3, x4 using a system of linear equations.



To solve this, you obviously need four equations – you can get those by looking at the traffic numbers at intersections! Also, the system will have many solutions, so I would like you to give me two possible traffic flow solutions. This means, you will need to take a look at the final matrix that your code spits out!!