## Assignment #3

This assignment is due on April 22nd via email to christian.wallraven+AMS2019@gmail.com.

If you are done with the assignment, make one zip-file of the `assignment3` directory and call this zip-file `STUDENTID1_STUDENTID2_STUDENTID3_A3.zip` (e.g.: `2016010000_2017010001_A3.zip` for a team consisting of two students or `2016010000_2017010001_2017010002_A3.zip` for a three-student team). The order of the IDs does not matter, but the correctness of the IDs does! **Please double-check that the name of the file is correct!!**

Please make sure to comment the code, so that I can understand what it does. Uncommented code will reduce your points!
**ALSO: I can also surf on the internet for code [ESPECIALLY TRUE FOR THIS ASSIGNMENT!!]. Downloading and copying and pasting other peoples' code is plagiarism and will NOT be tolerated. If you work as a team, the code needs to contain all team members' names!!**

### Part1 (50 points):

In this part you will implement a function `GaussElimination` that uses the Gaussian Elimination technique **with pivoting** as shown during class in the **powerpoint material. Please take a look at the slides I uploaded**. **Please use exactly this algorithm!!!**
a) Implement a Matlab function `GaussElimination` that solves a linear system of equations Ax=b for a square matrix A using forward elimination, backward substitution, and partial pivoting with the max-element **as shown during class**. The function should be defined as
`function [det,x] = GaussSolve(A,b)`
and should include **error handling to**

- **check whether the matrix A is square or not**
- **whether the dimensions of b and A fit**
- **whether following pivoting during the forward elimination step any of the leading coefficients become 0, if yes, abort with an error**

If any conditions become critical, **then the function should abort, telling the user the reason for it**.
The code also needs to return the determinant of the matrix A. We can do this because at the final step of the forward elimination we have created a upper triangular matrix U. It turns out that the determinant of such a matrix can be calculated as – so return it alongside the solution you obtain:

$$\det(U) = \prod_{i=1}^{n} u_{ii}$$

b) Which part of the code (forward elimination or backward substitution) takes longest to run? Insert your observations as a comment into `GaussElimination`. You can either use the built-in Matlab commands `tic` and `toc` to measure this time, or you can think about how the two parts are structured. I would prefer if you did the latter ☺

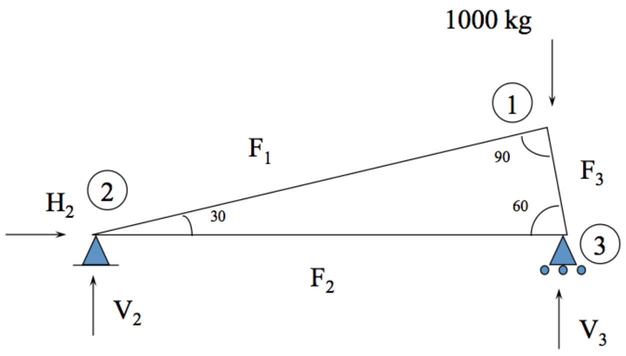
c) Now, please use the Matlab commands `tic` and `toc` to time how fast the whole `GaussElimination` script is for **random** matrices and vectors for the following sizes: n=2,5,10,100,1000.
In order to do this, write a script `TestGaussElimination.m` that runs `GaussElimination` and compares it with the **built-in Matlab backslash** command (**\**) for **random matrices and vectors (use the matlab function `rand(n)` for this)** for the given

problem sizes, records the execution time for each solution method and **plots the results in a nice plot**.
**How much faster is the built-in Matlab function than your function for each time step? Why do you think it is faster?**
Insert the answer and your observations as comments into `TestGaussElimination.m`.

## Part2 (10 points):

Write a script called `TrussAndEconomy.m` in which you set up, solve, and print out the solutions to the following two problems using `GaussElimination:`

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) Here's an example from economy:
Let's say we have three sectors: steel (S), energy (E) and cars (C). Let's also say we know

1) That the production of a dollar's worth of cars requires an input of \$0.6 from the steel sector, \$0.2 from the energy sector, and \$0.05 from the car sector.
2) That the production of a dollar's worth of energy requires an input of \$0.2 from the energy sector and \$0.04 from the steel sector and \$0.1 from the car sector.
3) That the production of a dollar's worth of steel requires an input of \$0.1 from the steel sector, \$0.65 from the energy sector, and \$0.05 from the car sector.

Our task is to find the output from each sector that is needed to satisfy a final demand of \$10 billion for steel, \$15 billion for energy, and \$20 billion for cars. So, this becomes the solution vector b.

Our model now says we need the total output X, which is given by the internal demands M on the output plus the final demands b. M is made from the information about the production values above, so we get X = MX+b.

From this, it should be easy to solve for X for you, which is the total output in billion US\$ for each industry. Do this using `GaussElimination.`

Also answer the following questions as comments in `TrussAndEconomy.m` .

1) Why do the numbers in 1), 2), 3) not add up to US\$1?
2) Does the solution X add up to the sum of the final demand? Why not?