VRIJE UNIVERSITEIT BRUSSEL

Second assignment: linear systems

Basic Techniques for Computer Simulations WPO | 18 March 2025

1 Instructions

In order to solve the assignment, you have to answer all questions included in the problem set.

Provide your answers as **one Python file** called `wpo1.py` that has to print all solutions to the questions and produce all plots requested in the problem set. The file has to be uploaded on Canvas before the deadline of the assignment.

Solutions to the questions have to be defined in the file as variables named `sol_x` where `x` is the number of the question. The solution file has to comply with the following format:

```
1
2  import numpy as np
3
4  def main():
5      """ Your names """
6      sol_1 = square(2)
7      print(sol_1)
8      sol_2a, sol_2b = logarithm(1000)
9      print(sol_2a, sol_2b)
10
11 def square(x):
12     """ Ex1 : Square of x """
13     return x**2
14
15 def logarithm(x):
16     """ Ex2 : Calculate log base ten and log base e """
17     return np.log10(x), np.log(x)
18
19 if __name__=='__main__':
20     main()
```

Listing 1: Example solution file format

2 Problem sets

Label B

A system of three reactors is linked by pipes. The rate of transfer of chemicals through each pipe is equal to a flow rate from reactor x to reactor y (Q_{xy} in m^3/s) multiplied by the concentration of the reactor from which the flow originates (C_x in mg/m^3). These concentrations are the unknowns of this problem.. When the system is at steady state, the flow into each reactor will balance the flow out:

$$F_{in} = Q_{13}C_1 + Q_{12}C_1 + Q_{21}C_2$$

$$Q_{12}C_1 = Q_{21}C_2 + Q_{23}C_2$$

$$Q_{13}C_1 + Q_{23}C_2 = Q_{33}C_3$$

It is known that $Q_{13} = 40$, $Q_{12} = 90$, $Q_{21} = 30$, $Q_{23} = 60$, $Q_{33} = 120$ and $F_{in} = 200$ mg/s. After expressing this problem with the usual matrix equation form $Ax = b$:

1. write your own Python function, which performs *forward elimination* and *back substitution* and returns the final A and b matrices, after the elimination procedure is completed;

2. solve the problem by coding a Python function which applies *LU factorization*, using the *scipy.linalg* and *numpy.linalg* Python modules, as described in the WPO presentation;

3. solve the problem by writing your own Python function that implements the *Jacobi* iterative method, until two-figure accuracy after the decimal point is achieved;

4. solve the problem by writing your own Python function for the *Gauss-Seidel* iterative method, until two-figure accuracy after the decimal point is achieved.