Prof. Dr. Jörg Müller
Arthur Fleig

AI VIII – Serious Games

Exercises for **Programming, Data Analysis, and Deep Learning in Python** (SoSe 2021)

Exercise Sheet no. 6, *Deadline*: Monday, May 31, 10:15

Notes

- Pay attention to the notes on the previous sheet.

Exercise 19 Numpy Middleschool (programming exercise) (8 points)

At the beginning of your file (after imports) set the seed to 0 using `numpy.random.seed(0)`. This makes the generated random numbers predictable.

a) Create an array `X` of integers from 0 to 15. Calculate the square root of each element of `X` and store the result in an array `X_sqrt`. Calculate x^2 for each $x \in$ `X` and store the result in an array `X_2`. Use `hstack` or `vstack` to create a matrix `XX` with `X_sqrt` as the first row (or column) and `X_2` as the second.

b) The command `numpy.random.randn` returns a sample (or samples) from the "standard normal" distribution (mean $\mu = 0$, variance $\sigma^2 = 1$). Use the command to create a 1000×5 matrix `M` that consists of samples from the normal distribution with mean $\mu = 4$ and variance $\sigma^2 = 5$.

c) Save the matrix `M` to a file and load the data from it afterwards
   i) using `numpy.save` and `numpy.load`,
   ii) using `numpy.savetxt` and `numpy.loadtxt`.

   What is the size difference of the two created files?

d) For each column of `M`, calculate the minimum value, the maximum value, the mean, and the variance using `numpy` methods. Compare the mean and the variance of each column to the parameters in b).

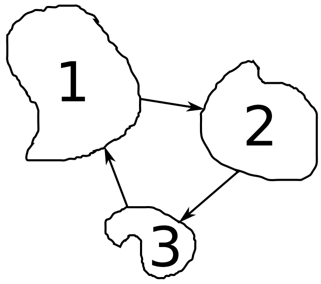
e) Create a matrix `M2` that consists of all entries of `M` except the third column.

f) Get all negative values from `M` and store them in a rank 1 array `M3`. Replace all positive values in `M` by 0.

Exercise 20 Linear Equations and Pollution (programming exercise) (7 points)

Note: You do NOT need to have a detailed understanding of differential equations. The knowledge from the lecture regarding linear equations is enough to complete this exercise!

Consider three lakes connected by streams. A pollutant is dumped into the first lake, which then spreads to the other lakes. The aim is to determine the amount of pollutant in each lake. The symbol $x_i(t)$ denotes the amount of pollutant at some time t in lake i, $i = 1, 2, 3$. For example, $x_1(60)$ denotes the amount of pollutant in lake 1 after 60 minutes. Similarly, the derivative $x_i'(t)$ denotes the *change* of pollutant. Initially (at time $t = 0$), the lakes are pollutant-free. To determine the amount of pollutant in each lake, one has to solve the following system of (differential) equations:



$$x_1'(t) = (x_3(t) - x_1(t))/1000 + 0.125, \tag{1}$$
$$x_2'(t) = (x_1(t) - x_2(t))/1000, \tag{2}$$
$$x_3'(t) = (x_2(t) - x_3(t))/1000, \tag{3}$$
$$x_1(0) = 0, \quad x_2(0) = 0, \quad x_3(0) = 0 \tag{4}$$

a) In the code, define a matrix A and a vector b with which you can rewrite the above system (1)-(3) as
$$x'(t) = Ax(t) + b,$$
where $x(t) = [x_1(t), x_2(t), x_3(t)]^\top$ and $x'(t) = [x_1'(t), x_2'(t), x_3'(t)]^\top$ are vectors.

On E-Learning you find a Python script that holds the exact solution to (1)-(4). However, in some cases it is not easy to calculate an exact solution. Hence, we will approximate it at *discrete* points in time. Given a stepsize h, we denote the k-th point in time by $t_k = k \cdot h$. For example, $t_0 = 0 \cdot h$ denotes the initial time $t = 0$; t_5 denotes the time $t = 5 \cdot h$, and so on. Set $h = 120$.

b) An approximation of the solution of (1)-(4) can be calculated with the formula:
$$x(t_{k+1}) = x(t_k) + h \cdot (Ax(t_k) + b), \quad x(t_0) = [0, 0, 0]^\top.$$
Use this formula to fill a matrix
$$\texttt{X\_expl} = \begin{bmatrix} x_1(t_0) & x_1(t_1) & x_1(t_2) & \dots \\ x_2(t_0) & x_2(t_1) & x_2(t_2) & \dots \\ x_3(t_0) & x_3(t_1) & x_3(t_2) & \dots \end{bmatrix},$$
in which you store $x(t_0), x(t_1), \dots$ until $t = 1440$ (this corresponds to one day). (How many steps k do you need to reach that time with the above stepsize h?)

c) Another approximation of the solution of (1)-(4) can be calculated by solving the following system of linear equations for $x(t_{k+1})$:
$$\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - h \cdot A \right) x(t_{k+1}) = x(t_k) + h \cdot b, \quad x(t_0) = [0, 0, 0]^\top.$$
Use this formula to fill a matrix `X_impl` analogous to b) (until time $t = 1440$).

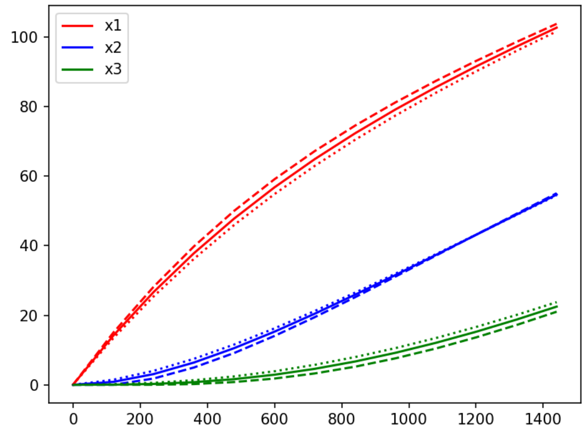


Figure 1: Sample plot of the explicit solution (solid), the approximation b) (dashed), and the approximation c) (dotted).

d) If you follow these instructions, then a plot as in Figure 1 will appear. Describe how the plot changes if you modify the stepsize h to
   - $h = 60$,
   - $h = 240$,

   but still simulate until $t = 1440$.