Freie Universität Berlin
Prof. Dr. Max von Kleist
Dr. Alexia Raharinirina

2. Assignment
Complex Systems for Bioinformaticians
SS 2024

Deadline: April 30, 12:00 (**before** the lecture)

The homework should be worked out individually, or in groups of 2 students. Pen & paper excercises should be handed at the designated deadline. Each solution sheet must contain the names and 'Matrikulationnummer' of all group members and the name of the group. Please staple all sheets.
Programming exercises must be submitted via Whiteboard.

Homework 1 (Pen/paper + Implementation (upload via Whiteboard), 1+2 points)
You are given the following ODE-system:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}x_1 &= -k_1 \cdot x_1 \cdot x_2 \\ \frac{\mathrm{d}}{\mathrm{d}t}x_2 &= -k_1 \cdot x_1 \cdot x_2 + k_2 + k_3 \cdot x_3 \cdot x_2 \\ \frac{\mathrm{d}}{\mathrm{d}t}x_3 &= -k_3 \cdot x_3 \cdot x_2 \end{aligned} \tag{1}$$

a) **(pen & paper)** Derive the stoichiometric matrix and rate functions $r_1, \ldots$.
b) **(to be uploaded via Whiteboard)** Write a small program implementing this model.
(i) Your program must write the stoichiometric matrix into a text file named "SMatrix.txt", collumns should be comma-delimited. *E.g., in Python the numpy function 'savetxt' using a comma-delimiters (',').* Write numbers as signed integers, e.g. '-1' instead of '-1.00'.
(ii) Compute the value of the ODEs for parameters $\mathrm{k}_1 = 0.01$, $\mathrm{k}_2 = 0.1$, $\mathrm{k}_3 = 0.01$ and time step $\triangle t = 1$ and the current system's state, which is given in the "Input.txt" file:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 25 \\ 15 \end{pmatrix}$$

Your program must write an output into a text file named "ODEValue.txt" as a comma-delimited textfile. *E.g., in Python the numpy function 'savetxt' using a comma-delimiters (',').* Round the output to the first two digits after the comma, e.g. '1.20' or '0.34'.
Name your program "Ex2_1.py" and submit via Whiteboard.
Tipp: There is a code scaffold provided for you in whiteboard which you can adapt.

Homework 2 (Implementation (upload via Whiteboard), 4 points)
a) (**to be uploaded via Whiteboard**) Write a program implementing the model above and generate trajectories using the stochastic simulation algorithm (SSA; = Gillespie's algorithm). You can hardcode the initial conditions if you prefer. The program reads the input file ("Input2.txt") provided in Whiteboard. The first number in the input file is the 'seed' of the random number generator, and the second is the number of trajectories N to be computed (see explanations below). Using this input, compute the trajectories for N simulations up to time $T = 1$ (this means that the last time point you consider is $t \leq 1$). For each simulation, write the time and the values X_2 into a file "Task2TrajY.txt", where 'Y' $= 1 \ldots N$ is the current simulation (e.g. "Task2Traj3.txt" contains the jump-times and the trajectory of X_2 from the third simulation).

The output textfile should be in the comma-separated text format using 3 digits after the comma (format '%1.3f'), e.g.

$$0.000, 0.010, 0.017, 0.202, ..., 9.763 \tag{2}$$

$$5.000, 6.000, 5.000, 6.000, ..., 12.000 \tag{3}$$

where the first row are the times $t^{(i)} \leftarrow t^{(i-1)} + \Delta t^{(i)}$ and the second row contains the corresponding number of X_2 at that time $X_2(t^{(i)})$. Name your program "Ex2_2.py" and submit via Whiteboard.

Hint:

i) Remember the (extremely wise) hints given to you in the seminar.

ii) For Python users, to import and set the input-variables:

```
import numpy as np
In = np.loadtxt('Input.txt')
np.random.seed(seed=int(In[0]))
NrSimulations = int(In[1])
```

iii) To write the output:

```
states, times = SSA(...)
## save trajectory
Output = np.concatenate((np.array(times,ndmin=2),np.array(states,ndmin=2)), axis=0)
np.savetxt('traj'+str(i+1)+'.txt',Output,delimiter = ',',fmt='%1.3f')
```

Tipp: There is a code scaffold (CodeScaffold2.py) provided for you in whiteboard which you can adapt.

Homework 3 (Implementation (upload via Whiteboard), 4 points)

Implement the EXTRANDE method for the model from Homework 2 with the following modification:

$$r_2(t) = k_2 \cdot 0.5 \cdot (\sin(t \cdot 180) + 2)$$

where 'sin' denotes the sinus function.

a) (**to be uploaded via Whiteboard**) Write a program implementing this model and generate trajectories using the EXTRANDE algorithm. The program reads the input file ("Input2.txt") provided in Whiteboard. The first number in the input file is the 'seed' of the random number generator, and the second is the number of trajectories N to be computed. Compute the trajectories for N simulations up to time $T = 10$. For each simulation, write the time and the values X_2 into a file "Task3TrajY.txt". The output textfile should be in the comma-separated text format using 3 digits after the comma (format '%1.3f'). Name your program "Ex2_3.py" and submit via Whiteboard. **Hint:** First, explore the value of $k_2 \cdot 0.5 \cdot (\sin(t \cdot 180) + 2)$ over the time horizon $t \in [0, \infty]$ and compute its maximum for computing the upper bound B.

Good luck!