FINAL ASSESSMENT (2 hours)

Exam Instructions:

1. All course documents available on Moodle are authorized. The use of Internet is strictly **forbidden**.
2. At the end of the exam, you must provide one Python program per exercise. Your programs must be **executable** with the command `python3 filename.py` and **display their results**.

I. Mathematical modelling of infectious diseases: the SCIR model

A famous model in epidemiology to describe the spread of a disease in a population is the SCIR model. The total population is decomposed into four categories:

- S represents the fraction of *susceptible* individuals who have not contracted the disease yet;
- C represents the fraction of *contaminated* individuals who have contracted the disease but are not contagious yet;
- I represents the fraction of *infectious* individuals who have contracted the disease and are contagious;
- R represents the fraction of *recovered* individuals who have healed from the disease and are now immune.

These four populations are related by the following set of ordinary differential equations:

$$\begin{cases} \dfrac{\mathrm{d}S}{\mathrm{d}t} = -\beta I S, \\ \dfrac{\mathrm{d}C}{\mathrm{d}t} = \beta I S - \dfrac{C}{\tau}, \\ \dfrac{\mathrm{d}I}{\mathrm{d}t} = \dfrac{C}{\tau} - \gamma I - \mu I, \\ \dfrac{\mathrm{d}R}{\mathrm{d}t} = \gamma I, \end{cases} \tag{1}$$

where the constants involved in the equations are the incidence rate β, the incubation time τ, the recovery rate γ and the death rate μ.

Question 1: Using an explicit time-stepping method (*e.g.,* Euler or RK4) seen in the lecture notes, integrate the above system of differential equations for parameters $\beta = 1/2$, $\tau = 10$, $\gamma = 1/12$, $\mu = 1/50$ up to time $t_\mathrm{f} = 100$, and with initial conditions $S(0) = 1 - C_0$, $C(0) = C_0$, $I(0) = 0$, $R(0) = 0$, with $C_0 = 1/2$. Add a comment in your code to justify your choice of the numerical value of the timestep h, referring to the characteristic timescales of the problem.
Important: Bonus points will be awarded for using NumPy arrays and vectorization.

Question 2: The fraction of alive individuals is defined as $A = S + C + I + R$, while the fraction of dead individuals reads $D = 1 - A$. Plot on the same figure the time evolution of the populations S, C, I, R, A and D. All the populations should be plotted with continuous lines with the following colour code: S in green, C in orange, I in red, R in blue, A in purple and D in black. Ensure the figure includes a grid. The plot should be self-explanatory and thus understandable by someone who did not do the exercise.

Question 3: The previous question shows that $I(t)$ is non-monotonic and reaches its maximum value I_max for a time t_max. Copy and modify your program so that the integration stops automatically once $I(t)$ has passed its maximum and becomes smaller than $I_\mathrm{max}/100$. Print the time $t_{1\%}$ at which this condition is met.
Hint: The value of I_max should be computed on the fly during the integration.

II. Buffon's needle problem

Georges-Louis Leclerc de Buffon asks the following problem (see Fig. 1): what is the probability that a needle of length ℓ thrown on a floor with equally spaced parallel lines separated by a distance d crosses one of the lines? It turns out that there is a simple analytical solution for this probability, which only depends on the ratio $z = \ell/d$:

$$\mathcal{P}(z) = \begin{cases} \dfrac{2z}{\pi} & \text{if} \quad z < 1, \\ \dfrac{2}{\pi}\left[z - \sqrt{z^2-1} + \arccos\left(\dfrac{1}{z}\right)\right] & \text{otherwise.} \end{cases} \tag{2}$$

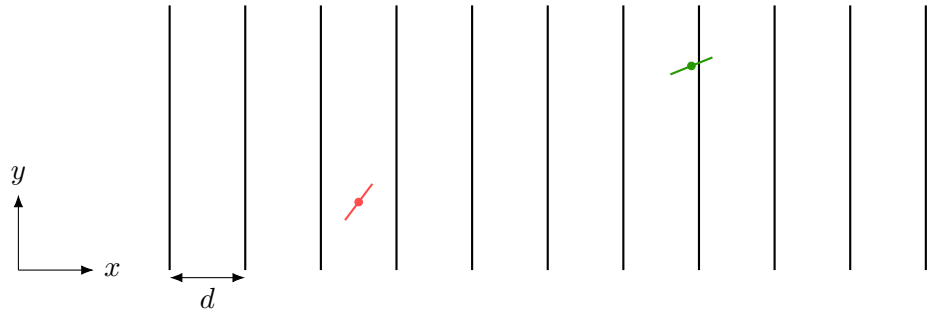

Figure 1: Illustration of Buffon's needle problem. Two needles were thrown on the ground on which equidistant parallel lines separated by a distance d had been drawn. The pink needle does not cross a line, while the green one crosses a line.

Question 1: Define a function `crossing_proba_th(z)` which takes as input a variable `z` which can be a float or a one-dimensional array and returns the value given by Eq. (2). In case the input is an array, the output should be an array of equal size with entries corresponding to the value given by Eq. (2) for each entry of the input. In case the input is a float, the output should be a float.
Hint: The $\arccos$ *function is implemented in NumPy (*`numpy.acos`*).*
Important: Bonus points will be awarded for using vectorization.

Question 2: In the following, we consider the situation shown in Fig. 1. The floor is made of 11 parallel lines with $d = 1$ (without loss of generality). The throwing of one needle is then parametrized by the abscissa $x \in [0,\, 10]$ of the centre of the needle (marked with a dot on the picture above), and by the angle $\theta \in [0,\, 2\pi]$ made by the needle with the x-axis. We assume that the x-coordinate of the needle and the angle made by the needle with the x-axis are real random variables uniformly distributed (the y-coordinate is irrelevant).
Define a function `crossing_proba(ell, N)` which takes as input a float `ell` (the length of the needles), throws `N` needles on the floor (as described before) and returns the fraction of needles which cross a line on the floor.
Hint: For a needle of angle θ *and of centre at abscissa* x*, the abscissae of its two extremities are*

$$x_1 = x - \frac{\ell}{2}\cos\theta, \quad \text{and} \quad x_2 = x + \frac{\ell}{2}\cos\theta. \tag{3}$$

Because the lines on the floor are spaced at integer positions along the x*-axis, there is a very simple criterion which involves the integer part of* x_1 *and* x_2 *to determine whether a needle crosses a line.*
Important: Bonus points will be awarded for using NumPy arrays and vectorization.

Question 3: Plot on the same graph the analytical solution with a black solid line and the result of your Monte Carlo simulations with disconnected red points for values of ℓ between 0 and 2. The plot should be self-explanatory and thus understandable by someone who did not do the exercise.

Question 4: For $\ell = d$, the probability of crossing is equal to $2/\pi$. This relation was used by Buffon to obtain an estimate of π. Compute the estimate of π for different numbers of throws $N = 10^k$ with $k \in \{2, \ldots, 8\}$. With a log-log plot, illustrate how the Monte Carlo estimate of π converges to its exact value as a function of the number of throws N. From your log-log plot, add a comment in your code describing the observed convergence rate as a function of N.