Student names: . . . (please update)

Instructions: Update this file (or recreate a similar one, e.g. in Word) to prepare your answers to the questions. Feel free to add text, equations and figures as needed. Hand-written notes, e.g. for the development of equations, can also be included e.g. as pictures (from your cell phone or from a scanner). ***This lab is not graded. However, the lab exercises are meant as a way to familiarise with dynamical systems and to study them using Python to prepare you for the final project.*** *This file does not need to be submitted and is provided for your own benefit. The graded exercises will have a similar format.*

In this exercise, you will familiarise with ODE integration methods, how to plot results and study integration error. The file `lab#.py` is provided to run all exercises in Python. Each `exercise#.py` can be run to run an exercise individually. The list of exercises and their dependencies are shown in Figure 1. When a file is run, message logs will be printed to indicate information such as what is currently being run and and what is left to be implemented. All warning messages are only present to guide you in the implementation, and can be deleted whenever the corresponding code has been implemented correctly.

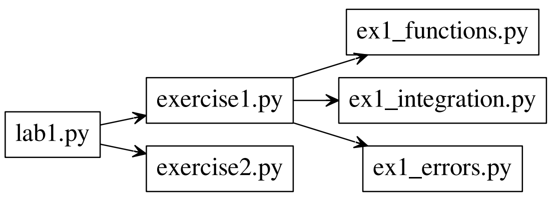


Figure 1: Exercise files dependencies. In this lab, you will be modifying `exercise1.py`, `ex1_functions.py`, `ex1_integration.py`,`ex1_errors.py` and `exercise2.py`. It is recommended to check out `exercise1.py` before looking into the other `ex1_*.py` files.

Question 1: Numerical integration

1.a Compute the analytical solution x(t) for the following linear dynamical system. Provide here the calculation steps, then implement the solution in `ex1_functions.py::analytic_function()` and run `exercise1.py` to plot the result.

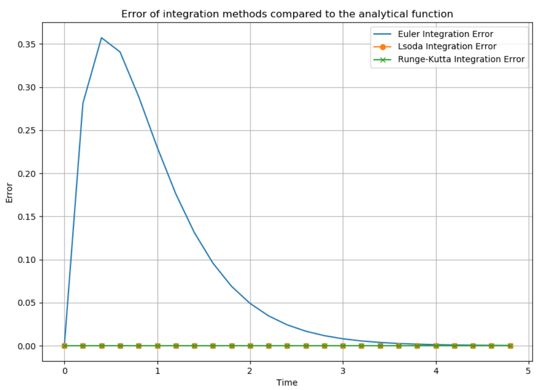
$$\dot{x} = 2 \cdot (5 - x), \quad x(t = 0) = 1 \tag{1}$$

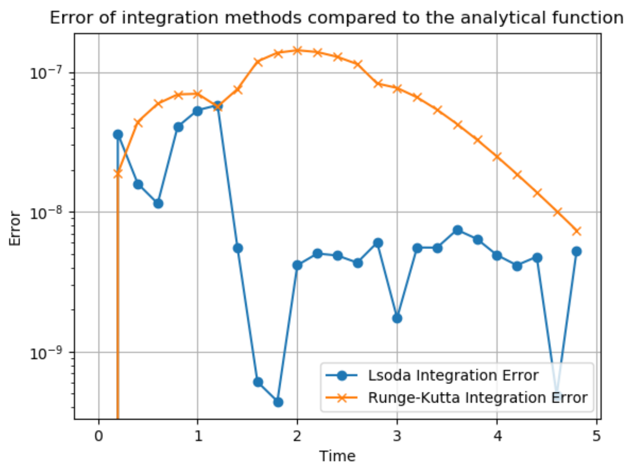
1.b In some cases, an ODE system may not have an analytical solution or it may be difficult to compute. Implement Euler integration in `ex1_integration.py::euler_integrate()`, then run `exercise1.py` again to compare the solution of `euler_integrate()` (with 0.2 timestep) to the analytical solution obtained previously and include a figure of the result here. Make sure to also implement `ex1_functions.py::function()` so that the code may be run correctly.
As a code template, check out `ex1_integration.py::euler_example()`.

1.c Various efficient libraries are available to facilitate ODE integration. Compare the Euler method with Lsoda (`ex1_integration.py::ode_intgrate()`) and Runge-Kutta 4th order (`ex1_integration.py::ode_intgrate_rk()`) integration methods by completing the corresponding functions using the scipy library in Python. See `exercise1.py::exercise1()` for the function calls. Provide the error values (there are multiple ways you could do this, choose an appropriate method and explain why), number of time steps, and include a figure comparing the integration methods.

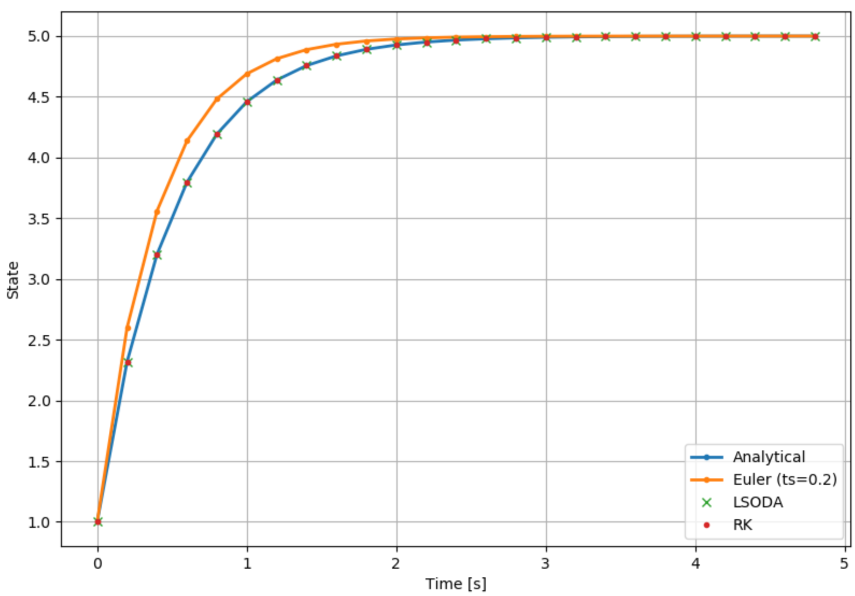
Erreur entre "analytical solution" calculée avant, en comparaison avec Euler, RK et LSODA

```
The mean and standard deviation of the error between the analytical result and the
euler method is: 0.0852982688307, 0.117417191585
The mean and standard deviation of the error between the analytical result and the
Lsoda method is: 1.17034183056e-08, 1.61493481034e-08
The mean and standard deviation of the error between the analytical result and the RK
method is: 6.41163343751e-08, 4.38020893657e-08
```





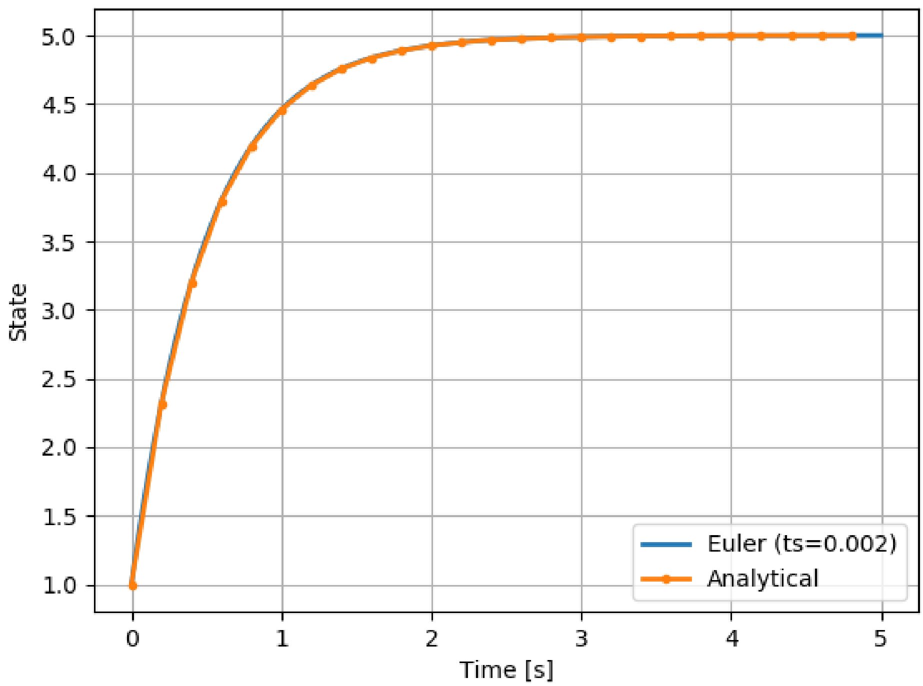
Comparison of the three integration method with the analytical solution



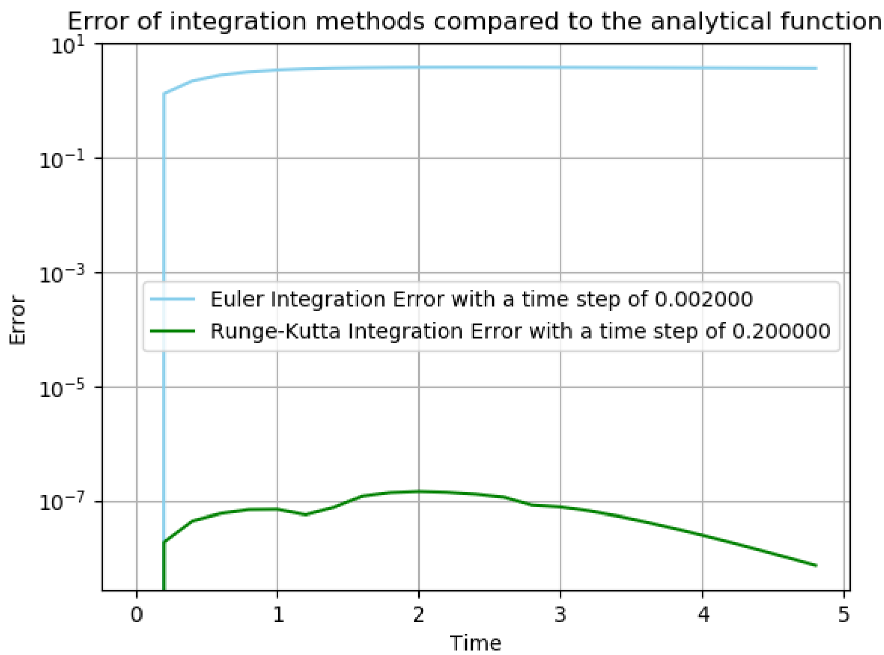
1.d As mentioned in the course, the comparison of question 1.c is not fair for the Euler method. Briefly say why. Choose another number of time steps for the Euler method that is fairer when compared to Runge-Kutta. Provide the error values, number of time steps, and include a figure comparing the two integration methods. Briefly discuss which integration method is best.

Euler's method uses the line tangent to the function at the beginning of the interval as an estimate of the slope of the function over the interval, assuming that if the step size is small, the error will be small. However, even when extremely small step sizes are used, over a large number of steps the error starts to accumulate and the estimate diverges from the actual functional value.

Comparison between euler with a time step of a hundredth of that of Runge Kutta

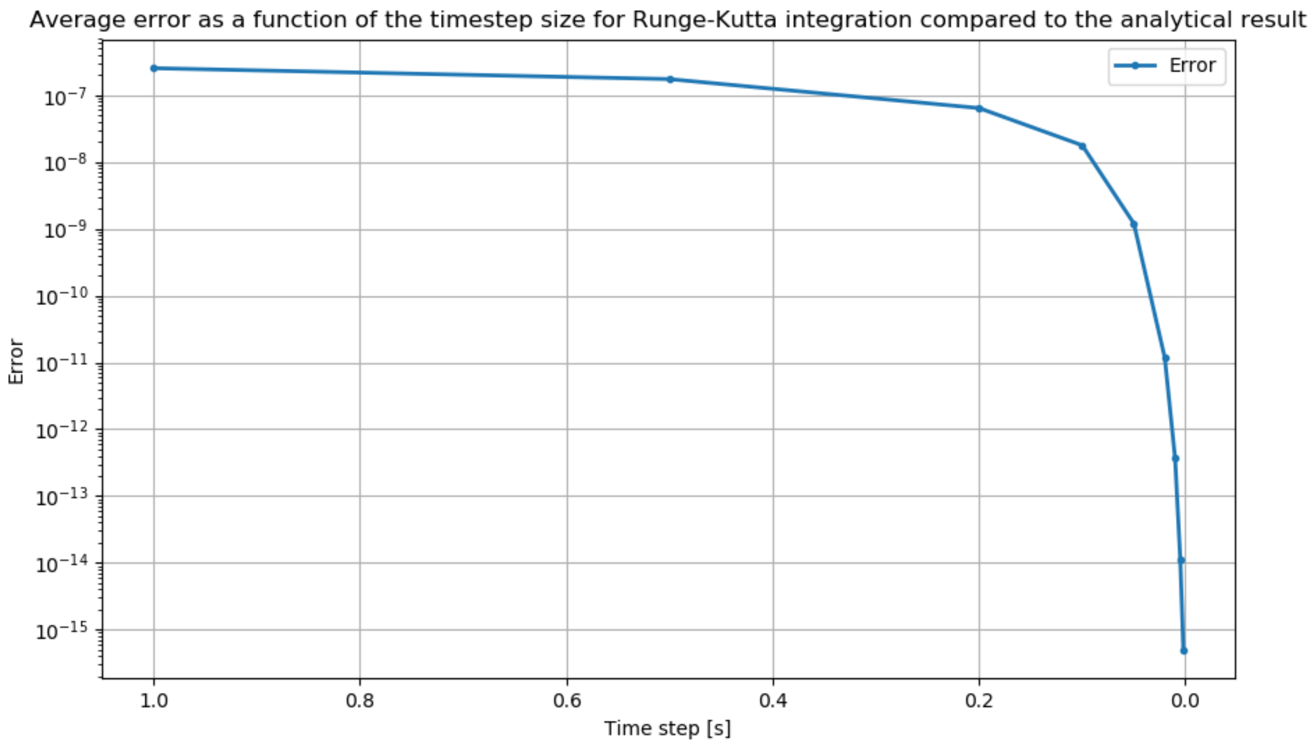


We see that even for a timestep 100 times bigger than for euler, RK still exhibits a smaller error rate. Therefore, we can safely assume that for a given timestep, RK will always be more accurate than Euler



1.e Test the role of the step size by plotting the integration error as a function of step size. You can use `ex1_errors.py::compute_error()` to do this by completing the code in `ex1_errors.py::error()`. How accurate is the solution compared to the analytical solution for different step sizes? Include here a graph showing the error against the step size. Explain which error measure you used (there are several options)

I included the absolute average (mean) error between the Runge-Kutta method and the analytical result, since it is the most stable and accurate method, it was relevant to assess the timestep role for this method, that is more representative of the true value. We observe, quite logically, that the smaller the timestep, the smaller the approximation error.



Question 2: Stability analysis

2.a Find the fixed points of the following linear dynamical system, and analyze their stability (briefly describe the calculation steps).

$$\dot{x} = Ax, \qquad A = \begin{pmatrix} 1 & 4 \\ -4 & -2 \end{pmatrix} \tag{2}$$

2.b Perform numerical integration from different initial conditions to verify the stability properties. See `exercise2.py::exercise2()` for implementation. Include some figures with these different time evolutions and their corresponding phase portrait and explain their roles.

2.c Change one value in matrix A such that the time evolution becomes periodic for some initial conditions. Say which value and include a time evolution figure.