Obligatorisk innleveringsoppgave IN1900

Versjon 2 (med rettelse av en liten feil i oppgave 5a)

Innleveringsfrist: 11. november kl 17.00

Introduction

This is a compulsory home assignment for all students in IN1900 in the fall of 2023. The first announcement of this assignment referred to Problems E.1, E.2, E.3, E.5 and E.7 in the document titled "Exercises for IN1900" (also called "oppgaveheftet"). Unfortunately, there were some typographical errors in those problems and you should therefore ignore those problems and instead relate only to the text below when you solve this home assignment. The problems are the same as before, but errors have been corrected and more details are provided.

Your solution must satisfy the following three requirements in order to be considered:

- It must be submitted through Devilry before 17.00 on 11 November
- It must satisfy the requirements given in the lecture on 31 October. Note: this requirement only concerns Problem 5 and only the class called **Cooling** and the function called **estimate_h**.
- It must be your own solution, and you must be able to explain it to a teacher if so requested.

Preparing for the assignment

As a preparation for this assignment, we strongly recommend that you follow the lecture on 2 November to learn the basics of ODE solving in Python. If you cannot attend the lecture, then watch the video from the lecture or go through the slides. We next recommend that you read the document *Solving Ordinary Differential Equations in Python* and look at the files on the accompanying web site.

ODE's

The general form of an ODE is

$$u'(t) = f(t, u(t)) \tag{1}$$

where $u(t)$ is an unknown function and where $f(t, u)$ is a known function. We also know the value of the unknown function $u(t)$ at $t = 0$, i.e. we have an initial condition $u(0) = u_0$. The formula above is a very convenient way of explaining what an ODE looks like in general; however, in concrete equations the function $f(t, u)$ is usually only implicitly given. For example, a real equation might look like

$$u'(t) = t + u(t) \tag{2}$$

and there is no mention of any function $f(t, u)$ there. However, we easily see that if we let $f(t, u) = t + u$ the equations (1) and (2) are identical. Another example is the equation

$$u'(t) = u(t)(1 - u(t)) \tag{3}$$

Here, if we let $f(t, u) = u(1 - u)$ the equations (1) and (3) are identical. Notice that in this case, the value of $f(t, u)$ does not depend on t. Nevertheless, we should still write the function as $f(t, u)$, i.e. with two input variables.

Your task

In this assignment you are going to solve some ordinary differential equations (ODE's) in Python. It is not particularly hard to write from scratch a Python program that solves ODE's. In fact, writing such a program is the main topic of the lecture on 2 November and the above mentioned document Solving Ordinary Differential Equations in Python. However, this is not what you are going to do below. Instead, your task is to use already existing "equation solvers" (described in detail in the above document) to solve some concrete equations. Thus your answers will consist of a mixture of code from *Solving Ordinary Differential Equations in Python* and code that you write yourself in order to make use of the equation solvers and plot the results. You can download all relevant code from the above book here. Now to the problems!

Problem 1: Solve a simple ODE with function-based code

This exercise aims to solve the ODE problem $u(t) - 5u'(t) = 0$ with the initial condition $u(0) = 0.1$ and for $t \in [0, 20]$.

a) Identify the mathematical function $f(t,u)$ in the generic ODE form $u' = f(t,u)$, and implement it as a Python function.

b) Use the **forward_euler** function from Section 1.1 of the document *Solving Ordinary Differential Equations in Python* to compute a numerical solution of the ODE problem. Use a time step of $\Delta t = 5$.

c) Plot together the numerical solution and the exact solution $u(t) = 0.1e^{0.2t}$.

d) Try successively smaller Δt values and demonstrate visually that the numerical solution approaches the exact solution.

Filename: **simple_ODE_func.py**

Problem 2: Solve a simple ODE with class-based code

Solve the same ODE problem as in Problem 1, but this time use the class **ForwardEuler** described in Section 1.4 of the document *Solving Ordinary Differential Equations in Python*, and implement the right-hand side function $f(t,u)$ as a class.

Filename: **simple_ODE_class.py**

Problem 3: Solve a simple ODE with the ODEsolver hierarchy

Solve the same ODE problem as in Problem 1, but this time use the class **ForwardEuler** in the **ODESolver** hierarchy from Section 2.2 of the document *Solving Ordinary Differential Equations in Python*. Implement the right-hand side function $f(t,u)$ as a class, just as you did in Problem 2.

Filename: **simple_ODE_class_ODESolver.py**

Problem 4: Compare ODE solving methods

There are several methods for solving ODEs in the **ODESolver** hierarchy described in Section 2.2 of the document *Solving Ordinary Differential Equations in Python*. One of them is called the Explicit Midpoint method and is implemented by the class **ExplicitMidpoint**. Suppose we have the ODE $u'(t) = f(t, u(t)) = \cos t - t \sin t$ with initial condition $u(0) = 0$. Solve this equation with the Explicit Midpoint method and the Forward Euler method and plot the solutions together with the analytical solution

$u(t) = t \cos t$. Use 20 time steps on the interval $t \in [0, 4\pi]$. Are the results obtained with the two methods similar?

Filename: **Midpoint.py**

Problem 5: Solve an ODE describing cooling of coffee

We all know that a hot object placed in a colder environment will cool down over time (unless of course the hot object generates heat, such as an oven!). However, not everyone knows how fast the hot object cools down. This is described by Newton's law of cooling. Suppose $T(t)$ is the temperature of the object at time t and that the environment has constant temperature T_s. Then the law says that the temperature changes according to the ODE

$$T'(t) = -h(T(t) - T_s) \tag{4}$$

The parameter h (with unit s^{-1}, i.e. "per second") is an experimentally determined constant (heat transfer coefficient) describing the efficiency (speed) of the heat exchange with the environment. In this exercise, we will model the cooling of freshly brewed coffee. First, we must find a measure of h. Suppose we have measured $T(0)$ and $T(t_1)$ for some time point $t_1 > 0$. We can then use a Forward Euler approximation of $T'(t)$ with one time step of length t,

$$\frac{T(t_1) - T(0)}{t_1} = -h(T(0) - T_s) \tag{5}$$

to make the estimate

$$h = \frac{T(t_1) - T(0)}{t_1(T_s - T(0))}. \tag{6}$$

a) Write a **class Cooling** containing the parameters h and T_s as data attributes (instance variables). Define a constructor in the class that sets these parameters. Implement the right hand side of the ODE as a **__call__(self, t, T)** method (if you prefer, you can call the last parameter to this method **u** rather than **T** so the method head becomes **__call__(self, t, u)** which is more in line with the notation used in the lectures).

b) Create a stand-alone function (i.e. a function that is not part of the above class) called **estimate_h(T0, T1, t1, Ts)** that returns an estimate of the parameter h, based on the temperature **T0** at time 0, the temperature **T1** at time point **t1**, and the temperature of the environment **Ts**. Use the formula in (6) to calculate the estimate.

c) Implement a test function **test_Cooling()** for testing that the class Cooling works. The test function should verify that the **__call__** method returns the correct results for given values of the arguments **T** and **t**.

d) Suppose the temperature of freshly brewed coffee is 95° C at time 0 (when it is poured into your cup) and 92° C after 15 seconds, in a room with temperature T_s. Solve the ODE numerically by a method of your choice from the ODESolver hierarchy described in Section 2.2 of *Solving Ordinary Differential Equations in Python*, for $T_s = 20$ and $T_s = 25$. Plot the two solutions in the same plot. The time interval where you solve the equations should be chosen to be long enough for the solutions to be "almost flat" and close to the room temperature T_s.

Filename: **coffee.py**