MA5233 Computational Mathematics

Homework Sheet 6

Simon Etter, 2019/2020

Final deadline: 25 October 2019, 7pm

Trajectory of a cannonball

Consider a cannonball which leaves the barrel at time $t = 0$ and location $\vec{x}(0) = \vec{0}$ with an initial velocity $\dot{\vec{x}}(0) = \vec{v}_0$. The trajectory $\vec{x} : [0, T] \to \mathbb{R}^2$ of this cannonball can be obtained as the solution to the ODE

$$\ddot{\vec{x}} = -\|\dot{\vec{x}}\|_2 \, \dot{\vec{x}} - \vec{e}_2, \qquad \vec{x}(0) = \vec{0}, \qquad \dot{\vec{x}}(0) = \vec{v}_0 \tag{1}$$

where the term $-\|\dot{\vec{x}}\|_2 \, \dot{\vec{x}}$ describes drag and the term $-\vec{e}_2$ with $e_2 = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$ describes gravity.[1] Equivalently, the ODE (1) can be split into the two coupled ODEs

$$\dot{\vec{v}} = -\|\vec{v}\|_2 \, \vec{v} - \vec{e}_2 \quad \text{with} \quad \vec{v}(0) = \vec{v}_0 \qquad \text{and} \qquad \dot{\vec{x}} = \vec{v} \quad \text{with} \quad \vec{x}(0) = \vec{0}.$$

which is sometimes more convenient. Our aim is to write code which solves these equations efficiently.

1. Complete the functions `midpoint_step()` and `heun_step()` such that they perform a single Runge-Kutta step according to the following Butcher tableaus:

   Midpoint:

0	0	0
1/2	1/2	0
	0	1

   Heun:

0	0	0	0
1/3	1/3	0	0
2/3	0	2/3	0
	1/4	0	3/4

   Test your code using the provided functions `trajectory()` and `convergence()`. If your code is correct, the error e_n will decay as follows:

   $$\text{Midpoint: } e_n = \mathcal{O}\big(n^{-2}\big), \qquad \text{Heun: } e_n = \mathcal{O}\big(n^{-3}\big).$$

   Hints.

   - Have a look at `euler_step()` to figure out the meaning of the function arguments and the expected return values.

[1] Note that we assume all physical constants like mass of the cannon ball, drag coefficient and gravitational constant to be one for simplicity

- The code for `midpoint_step()` can be simplified by using `euler_step()` to do some of the computations. Similarly, the code for `heun_step` can be simplified by using `midpoint_step()`.
- `convergence()` as provided uses Euler's method to compute a reference solution, but due to the slow convergence of Euler's method this reference solution will not be accurate enough to measure the errors in the solutions obtained using the midpoint and Heun's method for large numbers of steps `n`. Keep updating `convergence()` to use the most accurate stepping method available as you implement more stepping methods.

2. We observe that the ODE $\dot{\vec{v}} = \vec{f}(\vec{v})$ for $\vec{v}$ has a fixed-point $\vec{f}(\vec{v}_F) = \vec{0}$ for $\vec{v}_F = -\vec{e}_2$, which corresponds to the cannonball falling straight down and the drag balancing the acceleration due to gravity. Compute the Jacobian $\nabla\vec{f}(\vec{v}_F)$ and determine its eigenvalues.

3. You will find that the two eigenvalues λ_1, λ_2 computed in Task 2 both lie on the negative real axis. Let us denote by $\lambda := \min\{\lambda_1, \lambda_2\}$ the "more negative" of the two eigenvalues. We have seen in class that under these circumstances, explicit Runge-Kutta methods are stable only if the time step `dt` is chosen such that $|R(\lambda\,\texttt{dt})| \leq 1$, where $R(z) = 1 + z$ for Euler's method and $R(z) = 1 + z + \frac{z^2}{2}$ for the midpoint method. For both methods, determine $\texttt{dt} > 0$ such that $|R(\lambda\,\texttt{dt})| = 1$. Test your answer by replacing the placeholder `TODO` in `stability()` with the values of `dt` determined before. If your answer is correct, you will find that the distance $d := |\texttt{dy} + 1|$ between `dy` and its fixed-point value $(v_F)_2 = -1$ is approximately constant rather than exponentially decaying.

   Hint. You will find that d decays slightly for the midpoint method. This is due to the nonlinearity of the ODE, which is not captured by our linearised analysis around the fixed point.

4. Derive the stability function for Heun's method. You will find that $|R(z)| = 1$ for $z \approx -2.512745326618328624 0237$. Compute the time step `dt` and update `stability()` as described in Task 3.

5. To avoid the time-step constraints derived in the previous two tasks, we consider a semi-implicit Euler method given by

$$\vec{v}(t) = \vec{v}(0) - \|\vec{v}(0)\|_2\,\vec{v}(t) - \vec{e}_2, \qquad \vec{x}(t) = \vec{x}(0) + \vec{v}(0)\,t.$$

   Implement this scheme in the function `semi_implicit_euler_step()`. Test your code as described in Task 1. If your code is correct, you will find that the error e_n decays as $e_n = \mathcal{O}\big(n^{-1}\big)$.

6. (unmarked) Uncomment the line for the semi-implicit Euler method in `stability()`. Note how even with a time step $\texttt{dt} = 10^3$, the distance $d = |\texttt{dy} + 1|$ still decays (albeit slowly) for the semi-implicit Euler method.

The practical implications of this are as follows: Once the velocity $\vec{v}(t)$ approaches its steady state $\vec{v}_F$, we should be able to take arbitrarily large time steps `dt` since the two ODEs for position and velocity simplify to $\dot{\vec{v}} = 0$, $\dot{\vec{x}} = \vec{v}$ and these equations can be solved *exactly* with a single step of any of the methods considered above. However, for the explicit methods it is not possible to increase the time step `dt` beyond some `max_dt` $< \infty$ due to the stability constraint. The semi-implicit method has no such constraint; hence for large enough time intervals $[0, T]$ the semi-implicit method can be arbitrarily much faster than the explicit methods.