Prof. Dr. Carmen Gräßle & Jannis Marquardt, M.Sc.
Institute for Partial Differential Equations

Assignment 7

Upload your solution until
Mon, 03. June 2024, 3:00 pm.

Assignment 7.1 *ODE solvers in Python* **(4 x 5 = 20 points)**

In this exercise, we consider a bouncing ball which starts falling down from a height of 1 meter. If the ball hits the ground, it bounces back in the air and starts falling down again. Let us denote the height of the bouncing ball by $h(t)$ for times $t > 0$. Furthermore, denote the radius of the ball by $r > 0$. The bouncing process can be divided into two phases:

(1) The bouncing ball is free falling:
Since we neglect the friction of the air in our experiment, the acceleration of the bouncing ball can be described by the gravitational constant $g \approx 9.81$ and it holds that $\ddot{h}(t) = -g$.

(2) The bouncing ball hits the ground and bounces back
We make the assumption, that the bouncing ball does not significantly deform. The process of hitting the ground can be interpreted as a mass-spring-damping process. As soon as (and as long as) $h(t) < r$, i.e. when the surface of the ball starts touching the ground, the height of the bouncing ball can be described by the differential equation

$$mg - ch(t) - d\dot{h}(t) = m\ddot{h}(t),$$

where m is the mass of the bouncing ball, c the spring constant and d the damping constant.

In order to model the behaviour of the bouncing ball, create a file `bouncing_ball.py` and continue with the implementation as follows:

(a) Find a first order system $\dot{y}(t) = f(t, y(t))$ which describes the behaviour of the bouncing ball for times $t > 0$. Don't forget to distinct between the cases (1) and (2).

(b) Let the constants be given as $g = 9.81$, $r = 0.01$, $m = 0.01$, $c = 5000$ and $d = 0.1$. Furthermore, let the initial height of the bouncing ball be $h(0) = 1m$ and the initial velocity $\dot{h}(0) = 0\frac{m}{s}$. Compute a numerical solution of the system from (a) for all $t \in [0, 3]$ by using the function `scipy.integrate.solve_ivp`. Therefore, subdivide the interval $[0, 3]$ in 300 discrete equidistant timesteps t_i and compute $h(t_i)$ for each timestep. Plot the result in a time $\times$ height diagram (see Figure 1).
Hint: How does the parameter `t_eval` influence `solve_ivp`?

(c) Improve the simple time $\times$ height plot by replacing it with an animation which actually shows the ball falling down and bouncing back (you may find a reference video in StudIp).

Continue on next page ⇝

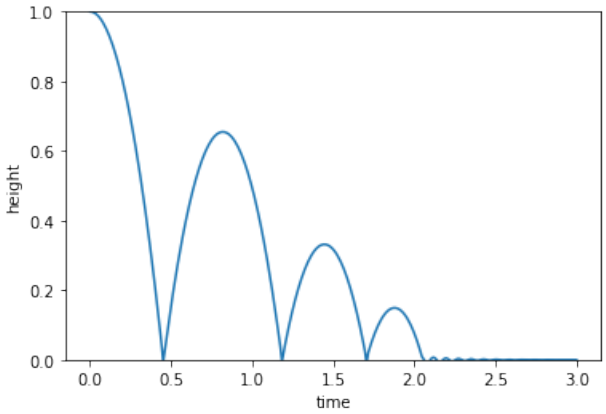

Figure 1: Height of the bouncing ball over time with the method described in (b).

If you have done everything correctly, you will notice a strange and unrealistic behaviour of the bouncing ball starting after $t = 2$ seconds. Under the assumption that we haven't made any errors during the derivation of the system and the implementation of the `solve_ivp` method, we have to assume that the numerical method fails. Let us test, whether this behaviour also occurs with other solvers.

(d) Call the `solve_ivp` again and use other values for the `method`-parameter. Compare at least three different methods and depict the results in a common time $\times$ height diagram.