Problem Set 2

Exercises for the Lecture Fundamentals of Simulation Methods

Prof. Dr. Ralf Klessen (Lecture Tuesday 9h - 11h and Thursday 9h - 11h)
Loke Lönnblad Ohlin (Tutor Group Thursday 11h - 13h, e-mail: loke.ohlin@uni-heidelberg.de)
Toni Peter (Tutor Group Thursday 14h - 16h, e-mail: toni.peter@uni-heidelberg.de)
Marcelo Barraza (Tutor Group Friday 11h - 13h, e-mail: barraza@mpia-hd.mpg.de)
Submit the solution to your tutor in electronic form by **Wednesday October 30, 2019**.

1. Order of an ODE integration scheme *(4 points)*

Consider the differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,t) \tag{1}$$

for the function $y(t)$ and a general right hand side $f(y,t)$. This may be integrated discretely with a Runge-Kutta scheme of the form:

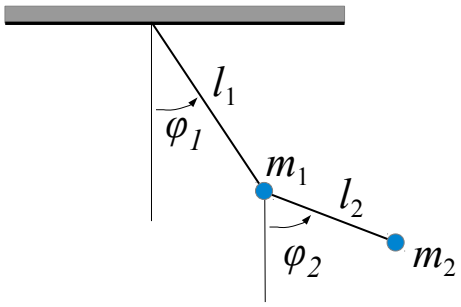
$$\begin{aligned} k_1 &= f(y_n, t_n), && (2)\\ k_2 &= f(y_n + k_1\Delta t, t_n + \Delta t), && (3)\\ y_{n+1} &= y_n + \frac{1}{2}(k_1 + k_2)\,\Delta t. && (4) \end{aligned}$$

Show that the truncation error per step is of third-order in the step-size Δt, or in other words, that the scheme is second-order accurate for the global integration error.

Hint: Consider a Taylor expansion of the difference $y_{n+1} - y(t_n + \Delta t)$ and assume that at the beginning of the step one starts out with the exact solution $y_n = y(t_n)$.

2. Double pendulum *(16 points)*

We consider a friction-less double pendulum that is constrained to move in one plane. The two masses m_1 and m_2 are connected via massless rods of length l_1 and l_2, respectively, as depicted in the sketch.



The Lagrangian of this system is given by the expression

$$\begin{aligned} L = \frac{m_1}{2}(l_1\dot{\phi}_1)^2 + \frac{m_2}{2}\left[(l_1\dot{\phi}_1)^2 + (l_2\dot{\phi}_2)^2 + 2l_1l_2\dot{\phi}_1\dot{\phi}_2\,\cos(\phi_1 - \phi_2)\right] \\ -m_1 g\, l_1(1-\cos\phi_1) - m_2 g\,[l_1(1-\cos\phi_1) + l_2(1-\cos\phi_2)] \end{aligned} \tag{5}$$

(a) Derive the Lagrangian equations of motion,

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\phi}} - \frac{\partial L}{\partial \phi} = 0, \tag{6}$$

for the angles ϕ_1 and ϕ_2. Hint: Declare conjugate momenta $q \equiv \frac{\partial L}{\partial \dot{\phi}}$ and *do not* explicitly carry out the absolute time derivative; it is sufficient if you give $\frac{\mathrm{d}q_1}{\mathrm{d}t}$ and $\frac{\mathrm{d}q_2}{\mathrm{d}t}$.

(b) Cast the system of equations into 1st-order form, such that the dynamics is described by the ODE

$$\frac{\mathrm{d}\vec{y}}{\mathrm{d}t} = \vec{f}(\vec{y}), \tag{7}$$

where $\vec{y}$ is a four-component vector. Hint: Use the conjugate momenta to eliminate the second derivatives, i.e. adopt $\vec{y} = (\phi_1, \phi_2, q_1, q_2)$ as state vector. Hint 2: When you define f_3, f_4, you can save time/effort if you "re-use" the values of f_1, f_2, no need to plug in their expressions again. You should do so when you are writing the program as well.

(c) Write a computer program that integrates the system with a second-order predictor-corrector Runge-Kutta scheme. Consider the initial conditions $\phi_1 = 50^\circ$, $\phi_2 = -120^\circ$, $\dot{\phi}_1 = \dot{\phi}_2 = 0$, and adopt $m_1 = 0.5$, $m_2 = 1.0$, $l_1 = 2.0$, and $l_2 = 1.0$. For simplicity, we shall use units where $g = 1$. Use a fixed timestep of size $\Delta t = 0.05$, and integrate for the period $T = 100.0$ time units (equivalent to 2000 steps). Plot the relative energy error, $(E_{\mathrm{tot}}(t) - E_{\mathrm{tot}}(t_0))/E_{\mathrm{tot}}(t_0)$, as a function of time.

(d) Produce a second version of your code that uses a fourth-order Runge-Kutta scheme instead. Repeat the simulation from (c) with the same timestep size, and again plot the energy error. How does the size of the error at the end compare, and is this consistent with your expectations?

(e) **(optional)** Let's make a visualization of our double pendulum in order to get a feel for its interesting and quite complex behavior. In fact, this pendulum is one of the simplest systems that shows non-linear chaotic behaviour. We would like to end up with a movie file if possible, so this part of the exercise is also meant to guide you through the steps that are necessary for this. But you may also hand in a sequence of still images if you prefer.

A standard method to make a digital movie is to produce a stack of images equally spaced in time, and then to encode them into a heavily compressed digital video stream. Suppose you have produced images, named `pic_000.jpg`, `pic_001.jpg`, `pic_002.jpg`, ..., etc., perhaps the simplest method to make a movie file from them is to encode them with the `ffmpeg` program. A possible command for this is

```
ffmpeg -r 24 -i pic_%03d.jpg movie.mp4
```

which will make use of a high-quality MPEG-4 compression scheme and a frame rate of 24 images per second. Numerous alternative programs for this exist, including `mencoder` and others.

To produce the images you can use the Python template `plot.py`, which makes images based on "fake data": $\phi_1(t) = \sin(0.1 \cdot t)$, $\phi_2(t) = \cos(0.1 \cdot t)$ and which is provided via the moodle website. You can also write your own routine. Combine it (i.e., the function `frame(...)`) with your pendulum simulation code. For a nice result, you may want to plot besides the current position of the pendulum the track of all past positions of the masses, as shown in the Python example.