Exercises for the lecture

Fundamentals of Simulation Methods

WS 2016/17

Lecturers: Frauke Gräter and Rüdiger Pakmor

Exercise sheet 2 ***(due date: Nov 3, 2016, 11:59pm)***

Integration of ordinary differential equations

1) Order of an ODE integration scheme (10 points)

Consider the differential equation

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

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

$$k_1 = f(y_n), \tag{2}$$

$$k_2 = f(y_n + k_1 \Delta t), \tag{3}$$

$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2)\,\Delta t. \tag{4}$$

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) Integration of a stiff equation (20 points)

Consider an ionized plasma of hydrogen gas that radiatively cools. Its temperature evolution is governed by the equation

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -\frac{2}{3\,k_\mathrm{B}} n_\mathrm{H}\,\Lambda(T) \tag{5}$$

where $\Lambda(T)$ describes the cooling rate as a function of temperature, $k_\mathrm{B} = 1.38 \times 10^{-23}\,\mathrm{J/K}$ is Boltzmann's constant, and n_H is the number density of hydrogen atoms. The cooling rate is a strong function of temperature T, which we here approximate by

$$\Lambda(T) = \begin{cases} \Lambda_0 \left(\frac{T}{T_0}\right)^{\alpha} & \text{for } T \le T_0 \\ \\ \Lambda_0 \left(\frac{T}{T_0}\right)^{\beta} & \text{for } T > T_0 \end{cases} \tag{6}$$

with $\Lambda_0 = 10^{-35}\,\mathrm{J\,m^3\,s^{-1}}$, $T_0 = 20000\,\mathrm{K}$, $\alpha = 10.0$, and $\beta = -0.5$. We consider isochoric cooling of gas at density $n_\mathrm{H} = 10^6\,\mathrm{m^{-3}}$, with an initial temperature of $T_\mathrm{init} = 10^7\,\mathrm{K}$.

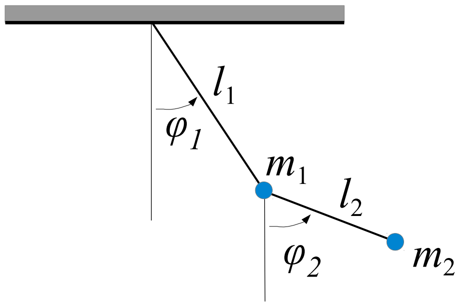
(a) Determine the temperature evolution $T(t)$ by integrating equation (1) with a second-order explicit RK predictor-corrector scheme and a fixed timestep, until the temperature has dropped below 6000 K. Use a timestep size of $\Delta t = 10^{10}$ sec. Make a plot of the time evolution of the temperature, with a logarithmic scale for temperature and a linear scale for the time.

(b) How many steps do you roughly need in (a) to reach the final temperature? Try to play with the timestep size and see whether you can significantly enlarge the timestep without becoming unstable.

(c) Now implement the second-order integration from (a) with an adaptive step size control, based on estimating the local truncation error by carrying out two half-steps for every step. Use an absolute local error limit $\Delta T_{\rm err}^{\rm max} = 50\,{\rm K}$ for every step. Overplot your result for the temperature evolution, on the plot for (a), using symbols or a different color. How many steps do you now need? Confirm that your scheme is robust to large changes of the timestep size given as input for the first step.

3) Double pendulum (20 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

$$L = \frac{m_1}{2}(l_1\dot{\phi}_1)^2 + \frac{m_2}{2}\left[(l_1\dot{\phi}_1)^2 + (l_2\dot{\phi}_2)^2 + 2l_1 l_2 \dot{\phi}_1\dot{\phi}_2 \cos(\phi_1 - \phi_2)\right]$$
$$-m_1 g\, l_1(1-\cos\phi_1) - m_2 g\,[l_1(1-\cos\phi_1) + l_2(1-\cos\phi_2)] \tag{7}$$

(a) Derive the Lagrangian equations of motion,

$$\frac{\rm d}{{\rm d}t}\frac{\partial L}{\partial \dot{\phi}} - \frac{\partial L}{\partial \phi} = 0, \tag{8}$$

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 $\frac{{\rm d}q_1}{{\rm d}t}$ and $\frac{{\rm d}q_2}{{\rm d}t}$.

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

$$\frac{{\rm d}\mathbf{y}}{{\rm d}t} = \mathbf{f}(\mathbf{y}), \tag{9}$$

where $\mathbf{y}$ is a four-component vector. Hint: Use the conjugate momenta to eliminate the second derivatives, i.e. adopt $\mathbf{y} = (\phi_1, \phi_2, q_1, q_2)$ 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 $\phi_1 = 50^o$, $\phi_2 = -120^o$, $\dot{\phi}_1 = \dot{\phi}_2 = 0$, 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, $(E_{\rm tot}(t) - E_{\rm tot}(t_0))/E_{\rm tot}(t_0)$, 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) 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 such 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 for example use the Python template `plot.py` (that makes images based on "fake data": $\phi_1(t) = \sin(0.1 \cdot t)$, $\phi_2(t) = \cos(0.1 \cdot t)$) that is provided on Moodle and 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.