Physik auf dem Computer **SS 2016**

Worksheet 12: Solving differential equations

June 29, 2016

General Remarks

- The deadline for handing in the worksheets is **Tuesday, July 5th, 2016, 10:00**.
- On this worksheet, you can achieve a maximum of 15 points.
- To hand in your solutions, send an email to `mkuron@icp.uni-stuttgart.de`.

Task 12.1: Two-dimensional Poisson Equation (5 points)

In this task the two-dimensional Poisson equation

$$\frac{\partial^2 \phi(x,y)}{\partial x^2} + \frac{\partial^2 \phi(x,y)}{\partial y^2} = \rho(x,y)$$

should be solved numerically for a two-dimensional charge distribution ρ.

The notebook `ws11.ipynb` and the Python script `ws11.py` generate and plot a two-dimensional sample charge distribution `rho`.

- **12.1.1** (3 points) Discretize the two-dimensional Poisson equation as explained in chapter 8 of the lecture script and write down the defining equations. Implement a Python function `solve_poisson2d(rho,h,sor_steps,omega)` that uses the SOR method to approximate the solution to the equation. The function should return the potential ϕ, where `rho` is a charge distribution, `h` is the discretization step and `sor_steps` is the number of steps of the successive overrelaxation scheme and `omega` is the SOR parameter ω.

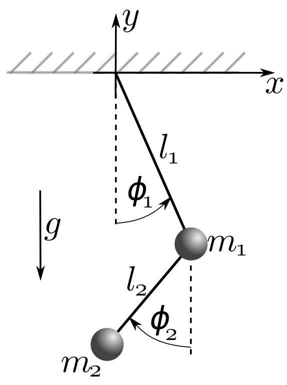
  Hint The Python command `rho_new=rho.reshape(N*N)` transforms any $N \times N$-matrix into a one-dimensional array with a length of N^2.

- **12.1.2** (2 points) Use the function to solve the equation for the sample charge distribution `rho`. Plot the potential after 1, 10 and 50 SOR steps at $\omega = 1.8$.

Task 12.2: Double Pendulum (10 points)

In this task we consider a double pendulum with masses $m_1 = 1$ and $m_2 = 1$ attached by rigid massless wires of lengths $l_1 = 1$ and $l_2 = 0.5$ as it is shown in Fig. 1. Here ϕ_1 and ϕ_2 are the angles between the wires and y-axis. The forces that act on the masses are $F_1 = -m_1 g$ and $F_2 = -m_2 g$, respectively. For such a system the equation of motion can be written as a system of second order differential equations (1) that is derived from the Lagrangian equations. The trajectory of the mass m_1 just falls on a circle, whereas the trajectory of the second mass m_2 is chaotic. One can obtain the trajectories of both masses by solving the following system numerically:

$$\begin{aligned} M l_1 \ddot{\phi}_1 + m_2 l_2 \ddot{\phi}_2 \cos(\Delta\phi) + m_2 l_2 \dot{\phi}_2^{\,2} \sin(\Delta\phi) + g M \sin\phi_1 &= 0 \\ l_2 \ddot{\phi}_2 + l_1 \ddot{\phi}_1 \cos(\Delta\phi) - l_1 \dot{\phi}_1^{\,2} \sin(\Delta\phi) + g \sin\phi_2 &= 0, \end{aligned}$$



r

Figure 1: Double Pendulum

where $M = (m_1 + m_2)$ and $\Delta\phi = \phi_1 - \phi_2$. .

- **12.2.1** (2 points) Convert the equation system to a system of four differential equations of first order which will be suitable for using a Runge-Kutta method, i.e. $\dot{y} = F[t, y(t)]$, where y is the 4-vector $(\phi_1, \dot{\phi_1}, \phi_2, \dot{\phi_2})$. Implement a function `F(t,y)` that takes a 4-vector `y` with angles and velocities and returns a 4-vector with `F(t,y)`.
- **12.2.2** (3 points) Implement a Python function `solve_runge_kutta(F,tmax,y0,h)` that solves the double pendulum for a total time of `tmax` using the 4-th order Runge-Kutta method and returns the angles ϕ_1 and ϕ_2. Here `y0` consists of initial angles $\phi_1(0)$, $\phi_2(0)$ and corresponding initial angular velocities $\dot{\phi_1}(0)$, $\dot{\phi_2}(0)$.
- **12.2.3** (3 points) Implement `solve_velocity_verlet(F,tmax,y0,h)`, a Python function that solves the double pendulum for a total time of `tmax` using the Velocity-Verlet method. The arguments of the function are the same as in the previous subtask.
- **12.2.4** (2 points) Using both implemented functions, solve the equation system with $h = 0.01$, $t_{max} = 100$ at the following initial conditions: $\phi_1(0) = \pi/2$, $\phi_2(0) = \pi/2$, $\dot{\phi_1}(0) = 0$, $\dot{\phi_2}(0) = 0$, in order to get the trajectory of the second mass m_2 and plot the trajectory in x, y axes. Compare the results of both methods.