Physik auf dem Computer **SS 2013**

Worksheet 1: Python and NumPy

April 22, 2013

General Remarks

- Deadline is **Tuesday, 23rd April 2013, 10:00**
- On this worksheet, you can achieve a maximum of 10 points.
- To hand in your solutions, send an email to
  - Olaf (`olenz@icp.uni-stuttgart.de`; Wednesday, 14:00–15:30)
  - Elena (`minina@icp.uni-stuttgart.de`; Wednesday, 15:45–17:15)
  - Tobias (`richter@icp.uni-stuttgart.de`; Friday, 15:45–17:15)
- Attach all required files to the mailing. If asked to write a program, attach the *source code* of the program. If asked for a text, send it as PDF or in the text format. We will *not* accept MS Word files!
- The worksheets are to be solved in groups of two or three people. We will not accept hand-in-exercises that only have a single name on it.
- Die Übungen finden statt im CIP-Pool des Instituts für Computerphysik (ICP) im Pfaffenwaldring 27.

Task 1.1 (4 points): Spring Pendulum

Copy the Python program `pendulum.py` from the home page to your home directory.

The program simulates a spring pendulum, *i.e.* a mass ($m = 1$) that is coupled to a harmonic spring (spring constant $k = 1$), as shows in figure 1. Initially, the mass is displaced from the equilibrium position ($x_0 = 0.3$) and at rest ($v_0 = 0$). The force that acts on the mass is $F = -kx$, the potential energy is $E_{\text{pot}} = \frac{1}{2}kx^2$ and the kinetic energy $E_{\text{kin}} = \frac{1}{2}mv^2$. To simulate the pendulum, it uses a time step of $dt = 0.1$ time units and simulates for $t_{\max} = 30$ time units. Let the frictional forces be negligible.

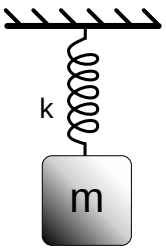

Figure 1: Spring Pendulum

At the end, the program plots the position x, the total energy E and the energy components over the time.

- 1.1.1 (2 points) Encapsulate the simulation of the pendulum into a class `Pendulum` that looks as follows:

```
class Pendulum:
  def __init__(self, k, dt, x_init = 0.0, v_init = 0.0):
      # fill in
  def step(self):
      """Perform␣a␣single␣simulation␣time␣step."""
      # fill in
  def getEnergy(self):
      """Compute␣and␣return␣the␣energy␣components."""
      # fill in
  def simulate(self, tmax):
      """Simulate␣the␣pendulum␣up␣to␣tmax.
␣␣␣␣␣␣Return␣the␣measurements␣as␣a␣NumPy␣array."""
      # fill in
```

- 1.1.2 (2 points) Extend the program such that it simulates two pendulums, one with $k = 1.0$ and $x_0 = 0.3$, the other with $k = 0.7$ and $x_0 = 0.1$. Let the program create a figure that shows four subplots:
  - the positions x of both pendulums vs. time
  - the total energy E of both pendulums vs. time
  - the kinetic energy E_{kin} of both pendulums vs. time
  - the potential energy E_{pot} of both pendulums vs. time

  Hint

  Use `pydoc matplotlib.pyplot.subplot` to get help on how to create the sub plots

Task 1.2 (6 points): Coupled Spring Pendulum

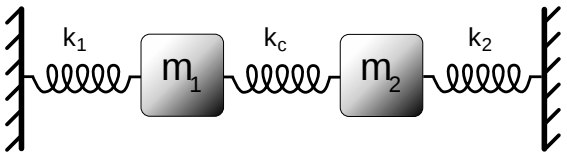

Figure 2: Coupled Spring Pendulum (Sketch[1] by `jim.belk`, CC-BY-SA 3.0)

In this task, the program from the previous task should be extended to simulate a coupled sping pendulum. A coupled spring pendulum consists of two masses $m_1 = m_2 = 1$, that are fixed between walls with three springs (spring constants $k_1 = 1$, $k_2 = 0.7$, $k_c = 0.2$) as in figure 2. Let the gravitation and frictional forces be negligible.

[1] `http://en.wikipedia.org/wiki/File:Coupled_Harmonic_Oscillator.svg`

Initially, the masses are at rest ($v_1(0) = v_2(0) = 0$) and both a are deplaced from their equilibrium positions ($x_1(0) = 0.3$ and $x_2(0) = 0.2$). The forces acting on both masses are $F_1 = -k_1 x_1 - k_c(x_1 - x_2)$ and $F_2 = -k_2 x_2 - k_c(x_2 - x_1)$. The "energy of mass i" is $E_i = \frac{1}{2}k_i x_i^2 + \frac{1}{2}m_i v_i^2$, the energy that is stored in the coupling spring is $E_c = \frac{1}{2}k_c(x_1 - x_2)^2$. The total energy of the system is $E = E_1 + E_2 + E_c$.

- 1.2.1 (4 points) Extend the program to simulate the coupled spring pendulum. Let the program create a figure with three subplots that shows
  - the positions x of both masses vs. time
  - the energies of both masses E_i vs. time
  - the total energy E of the system vs. time
- 1.2.2 (2 points) Modify the algorithms as follows: Change the method `step` such that it first computes the new positions from the old velocities, and only then the new velocities (the forces are still to be computed at the beginning of the method). Let the program create a plot of the total energy of the system both for the original algorithm (the *Symplectic Euler algorithm*) as well as for the modified algorithm (the *Euler forward algorithm*).

  What is the difference between the algorithms? Which of both algorithms would you prefer for an actual simulation?

Hint The length of the springs is not required as long as the equilibrium length is larger than the maximal displacement.