Exercise 3, November 20, 2015, to be handed in November 26.

This exercise requires you to produce some code that approximates the solution of a simple dynamical system. You should use MATLAB for that, which is freely available for students of RUB (follow instructions at
`http://www.rz.rub.de/dienste/software/beschaffung/firmen/matlab.html`).

Your source code, a MATLAB `*.m` file (or files if you do the bonus exercise) should be the final version of your code that enabled you to do all exercises. Submit this file to `oliver.lomp@ini.rub.de`. Document the code by comments, explaining each operation and highlighting by comments the particular change you made in each case.

Also write a regular report (can be hand-written) that adresses each exercise and includes figures as requested. You can submit this in paper or electronically.

For those of you, who have no or little experience with programming or MATLAB, a template source code file `simplesimulator.m` is available on the course web page. You can work on the basis of that template. Use the help function of MATLAB to understand function calls (e.g., `help plot`). You are welcome to redo the simulator, create your own, or extend it for your convenience. You may also use the MATLAB built in simulation tools like `ODE45` (type `help ODE45` in MATLAB and follow the links).

1. Write a simulator (or use the template) to numerically solve the differential equation

   $$\dot{x} = -\alpha x.$$

   The simulator should allow you to vary the initial value, $x(0)$, the length of the time interval over which you integrate, and the model parameter, α. You should also have some control over numerical precision (in the template code, this is the time step of the Euler formula).

   Explain your simulator (or the template) in words. Write down the numerical approximation method as a formula (in the template: the Euler formula) or as a verbal description of an algorithm. Explain how the parameters initial condition, model parameter, and numerical precision are controlled. Make a single plot of a solution generated from your simulator.

2. With your simulator, make a series of plots of the dynamics for a few different initial conditions, $x(0)$, both positive and negative. Document which initial values you used. Keep $\alpha = 0.5$ fixed. Choose the time interval of integration such that something happens within that time interval. A too short interval will lead to the solution hardly changing, a too long interval will lead to all action happening in the first few steps, and the solution being in the fixed point attractor for most of the time interval. The best would be to plot all solutions into a single figure (use MATLAB features `hold on` and `hold off` in `figure`).

3. Next, keep the initial value fixed, but now vary the parameter value of α and plot the different solutions in the same figure. Describe the effect of α on the solution in words.

4. Push the system to the limit of numerical stability. In the Euler procedure of the template, this can be done by making the time step larger (but adjust the number of time steps to cover the same time interval). In other methods you may directly control the admitted absolute error of the simulator. Describe what happens as you push the limits. Plot a series of solutions for increasing time step in the same time window for two different values of α (or for descreasing precision requirement in other methods). Comment on any observations you make.

Bonus Extend the simulator to simulate a two-variable dynamical system

$$\dot{x} = y; \qquad \dot{y} = -\omega x.$$

Plot both variables as time functions and vary the initial conditions. Does this system converge to the same state when initial conditions are varied?

Bonus Compare the template Euler approach to your approach in terms of computational time vs. precision.