# Exercise 1 (15 pts in total)

1. **ODE general**

Get familiar with the file solveODE.py/m. For the ODE y’=-20y with y(0)=1 the analytical (y=e-20\*t), the forward Euler, and the backward Euler solutions are implemented. Step size for the Euler implementations is defined by *tDt*. The analytical solution is taken as reference for the Euler implementations.

* 1. How does the step size influence the results of the Euler methods? At which step size the forward Euler method becomes instable? In your own words, why will the forward Euler method become instable? (**1 pt**)
  2. Implement a built-in solver (e.g., *ode45* in MATLAB or *scipy.integrate.solve\_ivp* with solver *‘RK45’* in Python) for the ODE y’=-20y and plot the result. (**1 pt**)
  3. Take the solveODE.py/m file as reference for your implementations.  
     Solve the ODE y’=-3y+9t with y(0)=9, time from 0 to 10 by a built-in solver (see above) (**1 pt**) and the backward Euler method (**1 pt**). Find the analytical solution (*dsolve* in *syms* mode in MATLAB or *dsolve* in the *sympy* package in Python). (**1 pt**)

1. **RC circuit / passive neuron**

RC.py/m implements a model of an RC circuit which computes the change in the membrane voltage of a passive cell membrane to a current step. The properties of the simplified cell membrane are described by a resistance and a capacitance in parallel, no active ionic currents are present.

* 1. Change the values for resistance (*R*) and capacitance (*C*) and find out how this influences the results. (**1 pt**) Describe the time constant of an RC circuit and how it can be computed. (**1 pt**) How can the maximum voltage in response to a current step input be computed (Hint: dv/dt = 0 in this case). (**1 pt**) In an RC circuit with a resistor and a capacitor: how long does it take until the capacitor is fully charged (>99%)? (**1 pt**)
  2. Implement a built-in solver (e.g., *ode45* in MATLAB or *scipy.integrate.solve\_ivp* with solver *‘RK45’* in Python) for the RC model and plot the result together with the forward and backward Euler solutions. Indicate the time steps made by the built-in solver. (**2 pts**)
  3. A list of ODE solvers for MATLAB is available at <https://de.mathworks.com/help/matlab/math/choose-an-ode-solver.html>; for Python at <https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html>. Solve the model with different available solvers and compare the results to the *ode45* (MATLAB) and *RK45* (Python) solutions of task 2.2. Give an explanation why you think the solver of your choice better fits the problem. (**2 pts**)
  4. There are several options available for fine-tuning of ode solvers. Find an option for the ode45 solver which improves the result. Provide a plot on how the result has improved compared to the original solution of task 2.2. (**2 pts**)

Short meaningful answers underlined with screenshots of the results are appreciated. For programming tasks, also provide the source code.

After completion of all four exercises, send your reports including your name and student ID to [paul.werginz@tuwien.ac.at](mailto:paul.werginz@tuwien.ac.at).