Computational Neurodynamics

Exercise Sheet 1 (Unassessed)
Numerical integration and neuron models

All the files for these exercises can be found online at

https://github.com/pmediano/ComputationalNeurodynamics

Question 1

The systems of ordinary differential equations (ODEs) that describe spiking neurons are usually impossible to solve analytically – at least in the complex, interesting settings we usually deal with. In those cases, we must resort to numerical integration methods. In general, an ODE is given by

$$y'(t) = f(y(t), t) \ .$$

The simplest algorithm for numerical integration is known as the Euler method. First, to integrate a continuous function $y(t)$, we discretise time in steps of width h (sometimes referred to as δt). Starting from time t_0, the Euler approximation to $y(t)$ is then

$$\begin{aligned} y_{n+1} &= y_n + h\, f(y_n, t_n) \\ t_{n+1} &= t_0 + nh \end{aligned}$$

Note that y_n could be a vector-valued variable (and f its gradient), so that the Euler method trivially generalises to systems of multiple ODEs.

a) Start up Python and run the program `EulerDemo.py`. Inspect the code and make sure you understand it. What is the effect of larger or smaller step sizes?

b) The Euler method can also be applied to second-order ordinary differential equations. For example, a mass-spring-damper system can be described by the following second-order ODE

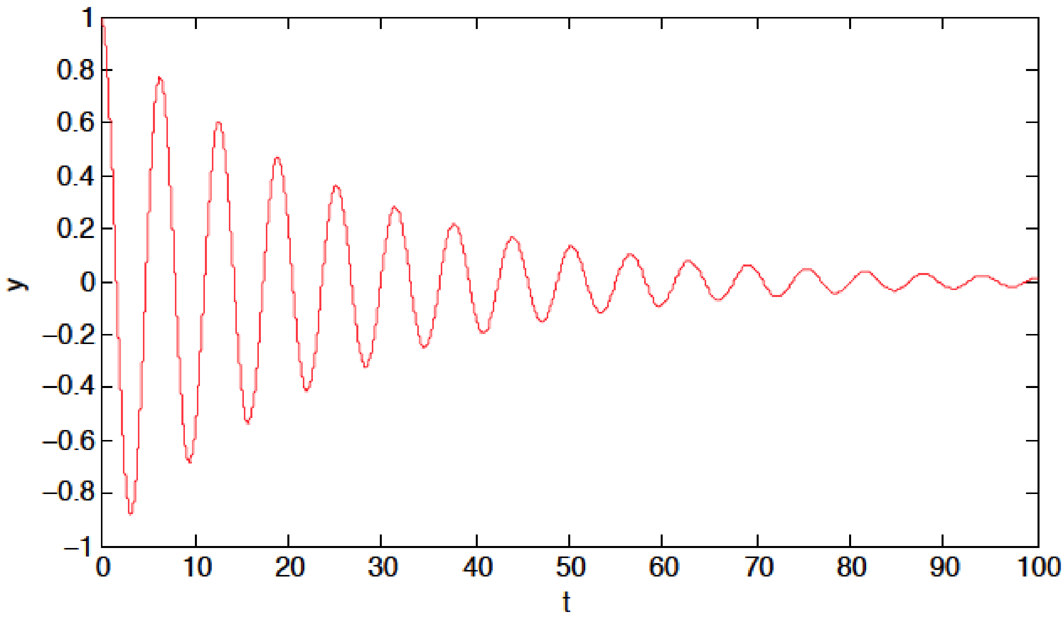
$$y''(t) = -\frac{1}{m}(c\, y'(t) + k\, y(t))$$

where m is the mass, c is the damping coefficient, and k is the spring constant. To do this you could create two auxiliary variables z_1, z_2 such that $z_1(t) = y(t)$ and $z_2(t) = y'(t)$, and use the Euler method to solve the ODE system z_1, z_2.

Using the code in `EulerDemo.py` as a model, write a Python script to simulate a mass-spring-damper system using the Euler method. Initially let

$$\begin{aligned} y(0) &= 1 \\ y'(0) &= 0 \end{aligned}$$

Letting $m = 1$, $c = 0.1$, and $k = 1$, you should be able to reproduce the following plot:



Question 2

a) Write a Python script that simulates the activity of a single Izhikevich neuron receiving a constant input current $I = 10$. Use the parameters for an excitatory (regular spiking) neuron from the notes, a step size $\delta t = 0.01$ and initial conditions $v = -65, u = -1$.

b) Using the lecture notes, adjust the parameters of the Izhikevich neuron so that they emulate an inhibitory and a bursting neuron. Run the code again and verify that it reproduces the relevant plots from the notes.

Question 3 (for enthusiasts only)

As simple as it is, the Euler method is often not good enough for simulation of complex systems. The go-to method in research applications is usually the 4th-order Runge-Kutta method, which involves more sophisticated approximations. In a nutshell, one RK4 integration step is

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$t_{n+1} = t_n + h$$

With:

$$k_1 = f(y_n, t_n)$$
$$k_2 = f\left(y_n + \frac{h}{2}k_1, t_n + \frac{h}{2}\right)$$
$$k_3 = f\left(y_n + \frac{h}{2}k_2, t_n + \frac{h}{2}\right)$$
$$k_4 = f(y_n + hk_3, t_n + h)$$

Write RK4 solvers for the ODEs of the previous questions. Visually, which method is better given the same step size? And given the same number of function evaluations for f? How can you quantify the performance of these methods?

If you are very keen, implement an RK4 solver for the Hodgkin-Huxley equation to get one of the most accurate neuron models known to date.