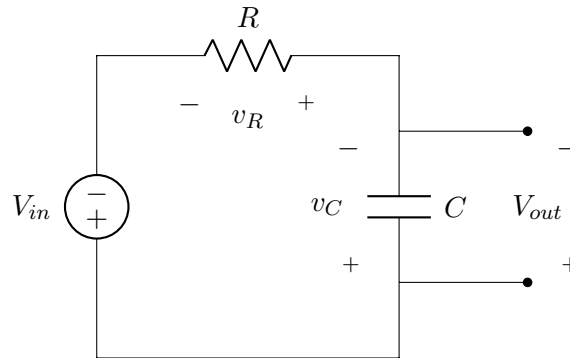


The coursework is due for submission on Monday 16 March 2020 Before embarking on this coursework it is useful to work through the exercises in Example Sheets 1 and 2, covering Euler's method, Heun's method and second-order equations.

## 1 RC circuit

A low-pass filter takes an input signal  $V_{in}$  (for instance, an audio signal) and renders it smooth by removing all high-frequency components (for instance, thermal noise). We will now observe low-pass filter usually used in NAB filters to cut-off frequencies above 1.592 kHz.



Consider the RC circuit as depicted above. For the circuit, we can write the following equations:

$$v_C(t) + v_R(t) = V_{in}(t), \quad \frac{1}{C} \int_0^t i_C(t) + R i_C(t) = V_{in}(t), \quad \frac{1}{C} q_C + R \frac{dq_C}{dt} = V_{in}(t).$$

Note:

1. the state is  $q_C(t)$ , with initial condition: capacitor pre-charged at time  $t = 0$  with  $q_C(0) = 500 \text{ nC}$ .
2. the input is  $V_{in}(t)$ ; the output is  $V_{out} = \frac{1}{C} q_C(t)$ ,

To impose a cut-off frequency of above 1.592 kHz, in a standard NAB filter, we set:

$$R = 1 \text{ k}\Omega, \quad C = 100 \text{ nF}$$

**Exercise 1.** Model the behaviour of the RC-circuit using three second-order Runge-Kutta methods: Heun's method, the midpoint method and a third method of your choice. The first two were covered in lectures, you will have to find the third one yourself.

Write **two files only**, one called **RK2.m** which implements all three methods. Given that they are second-order RK2 methods, you should be able to change from Heun to midpoint to your own method with changes in a single line of code only: recall the derivation of second-order RK and the equations linking parameters  $a, b, p, q$ . Submit it with the lines for one method active and the others commented out with %.

The method should be flexible enough to solve **any** first-order ODE of the form

$$\frac{dy}{dx} = f(x, y)$$

given an initial condition  $y(x_0)$ , a final value  $x_f$ , and a step-size  $h$ . You should write it using variable names  $k_1$  and  $k_2$ , corresponding to generic second-order RK methods.

The details for each case need to be passed as arguments to the function **RK2.m** in the file of the same name. See Problem sheet 1 for the Euler function example. Your second file **RK2-script.m** should contain calls to the function **RK2.m**, for a number of different examples, as outlined below.

Simulate the system and obtain plots for the following different input signals and observe how the amplitude of the output signal changes:

1. step signal with amplitude  $\bar{V}_{in} = 2.5 \text{ V}$ ;
2. impulsive signal and decay:

$$V_{in} = \bar{V}_{in} \exp\left\{-\frac{t^2}{\tau}\right\} \quad V_{in} = \bar{V}_{in} \exp\left\{-\frac{t}{\tau}\right\}$$

with  $\bar{V}_{in} = 2.5 \text{ V}$  and  $\tau = 100 (\mu\text{s})^2$ , resp.  $\tau = 100 \mu\text{s}$ .

3. sine, square, and sawtooth waves with amplitude  $\bar{V}_{in} = 5 \text{ V}$  and different periods  $T = 100 \mu\text{s}$ ,  $T = 10 \mu\text{s}$ ,  $T = 500 \mu\text{s}$ ,  $T = 1000 \mu\text{s}$ . For square and sawtooth wave, try Matlab help: "square-", resp. "sawtooth wave".

Note that in each case you will need to make a decision on how large  $t_f$  should be: this too, will depend on what you can/want to show, such as, for example, transient vs. steady-state. Make sure your pictures show sensible choices.

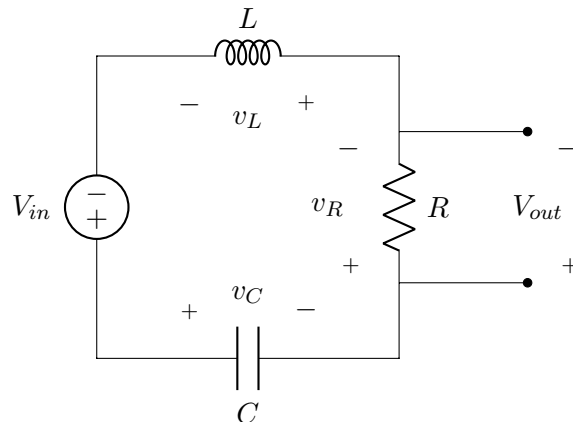
When you have gone through the given set of parameters, try some new ones: vary  $R, C, q_C(0), \tau$ , and  $T$ ; try some different input functions  $V_{in}$ ; see if you can get different kinds of behaviour; this part of the exercise is open-ended and you need to be inventive.

The behaviour of the output can be explained in terms of the input, parameters, the nature of the circuit. In each case, explain briefly what you observe. How do the plots reflect what is happening in the circuit?

Code, pictures of the output and comments go in the report.

**Exercise 2.** Write a script called **error\_script.m** in which you carry out error analysis for the basic case with  $R, C, q_C(0)$  as above, and as input a cosine wave of period  $T = 100 \mu\text{s}$  and amplitude  $\bar{V}_{in} = 5 \text{ V}$ . Obtain the exact solution of the ODE, and for each method of the three from Exercise 1, obtain a numerical solution: again, switching methods by only commenting out lines of code appropriately. Compare the numerical solution with the exact solution, obtaining the error as a function of  $t$ . Plot the error function. Vary the time step  $h$  for a suitable number and range of values, and obtain a log-log plot to show the order of the error. Be sure to explain and justify your empirical observation.

## 2 RLC circuit



An RLC circuit represents a standard example of a harmonic oscillator, namely a device able to resonate to a sinusoidal input signal. Among the many applications for this circuit, there is the tuning of analogue radio receivers. Usually, they are used as band-pass filters.

Consider the RLC circuit as depicted. For the circuit, we sum voltages:  $v_L(t) + v_R(t) + v_C(t) = V_{in}(t)$  and derive the ODEs:

$$L \frac{d}{dt} i_L(t) + R i_L(t) + \frac{1}{C} \int_0^t i_L(t) = V_{in}(t) \quad \equiv \quad L \frac{d^2 q_C}{dt^2} + R \frac{dq_C}{dt} + \frac{1}{C} q_C = V_{in}(t).$$

The state is  $q_C(t)$ . The input is  $V_{in}(t)$ . The output is  $V_{out} = v_R = R \frac{d}{dt} q_C(t)$ . Assume that the capacitor is pre-charged at time  $t = 0$  with  $q_C(0) = 500 \text{ nC}$ . Moreover, no initial current flows through the inductor at time  $t = 0$ :  $i_L(0) = \frac{d}{dt} q_C(0) = 0 \text{ A}$ . The values for resistance, capacitance, and inductance are:

$$R = 250 \, \Omega, \quad C = 3.5 \, \mu\text{F}, \quad L = 600 \, \text{mH}.$$

**Exercise 3** Write a matlab script called **RLC\_script.m** and a matlab function called **RK4.m**. In the script you should set up the two coupled first order ODEs in  $q$  and  $q'$  to solve the RLC second order ODE (1) for  $q$ . The script will include a call to **RK4.m**, the matlab function you will write to implement the *classic fourth-order Runge-Kutta* algorithm for **any** system of two coupled first order equations:

$$z' = f_1(x, y, z) \quad \text{and} \quad y' = f_2(x, y, z)$$

with initial conditions  $z = z_0$  and  $y = y_0$  when  $x = 0$ ; note that both  $y$  and  $z$  are functions of  $x$ .

You will have to think about which functions and parameters you need to pass as arguments in the call to **RK4.m** (Please note: second-order refers to the ODE, fourth-order to the order of the RK algorithm.)

Once you have the matlab code working, use it to simulate the system, and obtain plots, for different input signals and observe how the amplitude of the output signal changes:

1. step signal with amplitude  $\bar{V}_{in} = 5 \text{ V}$ ;
2. impulsive signal with decay  $V_{in} = \bar{V}_{in} \exp\left\{-\frac{t^2}{\tau}\right\}$ , with  $\bar{V}_{in} = 5 \text{ V}$  and  $\tau = 3 \text{ (ms)}^2$ .
3. square wave with amplitude  $\bar{V}_{in} = 5 \text{ V}$  and different frequencies  $f = 109 \text{ Hz}$ ,  $f = 5 \text{ Hz}$ ,  $f = 500 \text{ Hz}$ ;
4. sine wave with amplitude  $\bar{V}_{in} = 5 \text{ V}$  and different frequencies  $f = 109 \text{ Hz}$ ,  $f = 5 \text{ Hz}$ ,  $f = 500 \text{ Hz}$ .

As in Exercise 1, when you have gone through the given set of parameters, try some new ones: vary  $R$ ,  $C$ ,  $Lq_C(0)$ ,  $q'_C(0)$ ,  $\tau$ , and  $f$ ; try some different input functions  $V_{in}$ ; see if you can get different kinds of behaviour. Note that in each case you will need to make a decision on how large  $t_f$  should be: this too, will depend on what you can/want to show, such as, for example, transient vs. steady-state, and how frequencies interact. The behaviour of the output can be explained in terms of the input, parameters, the nature of the circuit. In each case, explain briefly what you observe. Code, pictures of the output and comments go in the report.

### 3 Relaxation

The Laplace equation in the plane

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0,$$

can be solved using finite differences in the form of the *relaxation* method, as outlined in lectures.

**Exercise 4.** Write a matlab script called **relaxation.m** to implement the relaxation method for Laplace's equation on the unit square, with simple boundary conditions

$$\begin{aligned} u(0, y) = u(1, y) &= y, & \text{for } 0 < y < 1; \\ u(x, 0) = u(x, 1) &= x, & \text{for } 0 < x < 1; \end{aligned}$$

Things to consider:

- your choice of gridsize and residual size;
- efficient testing of residual for convergence. **what exactly?**
- Once you have the relaxation method working, try it out for some different, more challenging boundary conditions: this is open ended, be inventive.
- For example, what happens when the BCs don't match at the corners? Does your method reproduce this?
- All code, your choices, all plots into report, well commented.

**Exercise 5.** Implement the solutions in Exercise 4, using Successive Over-Relaxation, SOR. Track the computation time for different values of the discretization  $h$ , and compare convergence time between relaxation and SOR.