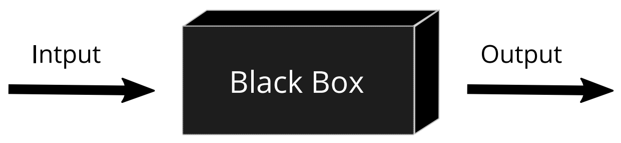
System Identification
Year 2 Scientific Computing 2018-19

Andreas Freise
Aaron Jones, Dominic Danks, Piper Fowler-Wright

Issue: 1
Date: October 17, 2018



School of Physics and Astronomy
University of Birmingham
Birmingham, B15 2TT

1 Introduction

In assignment 2 you will be presented with a number of unknown systems or ‘black boxes’. The systems are represented by python code that accepts input data and produces output data. The aim of the assignment is to create suitable input data for the boxes and record the corresponding output data, in order to learn more about what is going on inside each box. This process is sometimes referred to as ‘system identification’. A common practical example is the need to test subsystems, such as electronic modules, of an experimental setup. Even though the modules comes with their data sheets, the actual behaviour might deviate somewhat from the specification. In precision experiments even very small deviations can cause an experiment to fail in various interesting ways. Measuring the response of each part and creating an accurate model of (parts of) the system is an important aspect of debugging an experiment.

In this assignment we will use electronic circuits as our example systems to test, and each black box emulates the behaviour of one of the circuits described below.

For many system comparing input and output signals will be much simpler in the frequency domain, i.e. when the input and output signals are described as spectra (signal components as a function of frequency) rather than as a time series (signal over time). The Fourier transform can be used to transform time domain signal into the frequency domain and vice versa. You will make use of the Fast Fourier Transform module of the `numpy` package, to investigate the boxes in the frequency domain, and use this to identify the specific electronic system contained in each box.

2 Discrete Fourier Transform

Using the Fourier Transform to analyse signals in the frequency domain is a extensive topic on its own and we cannot cover any of the details here. Instead we will provide a simple recipe to use the Fourier Transform as a tool for this particular task.

A electronic signal $x(t)$ has been recorded such that we have a finite set of uniformly spaced time-samples

$$x_n = x(t_n) \tag{1}$$

with t_n the equally spaced times and $n = 0 \ldots N-1$. The total measurement time is $T = t_{N-1} - t_0$ and the sampling rate or frequency is given as $F_s = N/T$ in samples per second.

Using the (Discrete) Fourier Transfrom (DFT) we can compute a new array of numbers that represent how much of the signal is oscillating at a certain frequency. In the following we assume an even number of samples N.

The result of the DFT are given as:

$$X_n = X(\omega_n) = \sum_{k=0}^{N-1} x_k e^{-2\pi i n k/N} \tag{2}$$

with ω_n representing a list of uniformly spaced frequencies given in radians ‘per cycle’ with $\omega_n = 2\pi f_n$. The frequencies up to $n = N/2$ are given as:

$$f_n = nF_s/N \tag{3}$$

Note that the maximum frequency is given as $F_s/2$. The sampling frequency defines what frequency content can be stored in the discrete time series.

The remaining frequencies (for $n = N/2+1$ to $n = N-1$) correspond to negative frequencies. In physics we often deal with real (measurable) signals, in which case the Fourier components are identical under $\omega \to -\omega$, i.e. the positive and negative frequency Fourier components are equal. In other words all the information of the signal is contained in the first half of the frequency-samples.

Note that the output values X are complex numbers.

We can also perform a inverse transform to go back from the frequency domain:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{2\pi i n k/N} \tag{4}$$

The `numpy` commands for an DFT and inverse DFT are:

```
X = numpy.fft.fft(x)
x = numpy.fft.ifft(X)
```

In order to plot the frequency spectrum X, you also need to determine the corresponding sample points in the frequency domain. The `numpy` function `.fftfreq` should be used for this purpose: If the original numpy array passed to `.fft` is of size `N` and has a constant spacing between elements of `timestep`, `.fftfreq(n, d=timestep)` returns the sample frequencies against which the output of `.fft` should be plotted. In general, the values returned by `.fft` will be complex and so it is important to plot the absolute values against these frequencies using `numpy.abs()`.

3 Transfer functions of electric circuits

Simple RCL circuits (made from resistors R, capacitors C and inductors L) are often used to selectively transmit signals according to their frequency and may be categorised as either low pass, band pass or high pass depending on the range of frequencies transmitted. Their behaviour is characterised by their transfer function:

$$T(\omega) = \frac{V_{out}(\omega)}{V_{in}(\omega)}. \tag{5}$$

Each of the following filter circuits is contained within one of the three boxes you are to investigate in the first task of assignment 2:

1. An RL high pass filter (output taken across the inductor)
2. An RC low pass filter (output taken across the capacitor)
3. An RLC band pass filter (output taken across the resistor)

The components used to construct these circuits are assumed to be ideal, and are shared between the boxes of task 1 in the sense that the resistor, capacitor and inductor have the same resistance, capacitance and inductance wherever they appear. These values are unique to your assignment.

Recall the simple relationship between the input and output voltage for a potential divider circuit containing a number of electrical impedances in series:

$$V_{out} = \frac{Z_{out}}{Z_{tot}} V_{in} \tag{6}$$

Here Z_{out} is the impedance of the component across which the output is taken and Z_{tot} the total impedance of the circuit; a linear sum for components in series. Given the complex impedance of a resistor (R), an inductor

$(j\omega L)$ and a capacitor $(-j/\omega C)$, expressions for the total complex impedance and hence transfer function of each filter circuit may be determined.

A Bode magnitude plot is a graph of the magnitude of the transfer function - a ratio - against angular frequency ω. These often use a decibel scale for the vertical axis and a logarithmic scale for the horizontal axis. The following provides some details on decibels and shows Bode plots of this type because they are commonly shown in this form in the literature. However, during your assignments you will not be using decibels. The difference in decibels (dB) between two voltages, V_{out} and V_{in}, is defined by the logarithmic ratio

$$20 \log_{10} \left| \frac{V_{out}}{V_{in}} \right| = 20 \log_{10} |T(\omega)| \,. \tag{7}$$

The absolute difference is known as a *gain* if $V_{out} > V_{in}$ and a *loss* (or attenuation) if $V_{out} < V_{in}$. For example, $|T(\omega)| = \frac{1}{\sqrt{2}}$ corresponds to $20 \log_{10} \left| \frac{1}{\sqrt{2}} \right| \approx -3\, dB$ i.e. a *loss* of approximately $3\, dB$.

A quantity of particular interest is the cut-off or *corner* frequency, f_0, and its corresponding angular frequency, $\omega_0 = 2\pi f_0$. This is defined as the frequency for which $|T(\omega)| = \frac{1}{\sqrt{2}}$, corresponding to an attenuation of *approximately* $3\, dB$. You will be required to evaluate the cut-off frequency of an RL high pass circuit in the second task of assignment 2.

A schematic and example bode magnitude plot for each circuit is provided in Figures 1, 2 and 3. Note: The parameter values used are for illustrative purposes only.

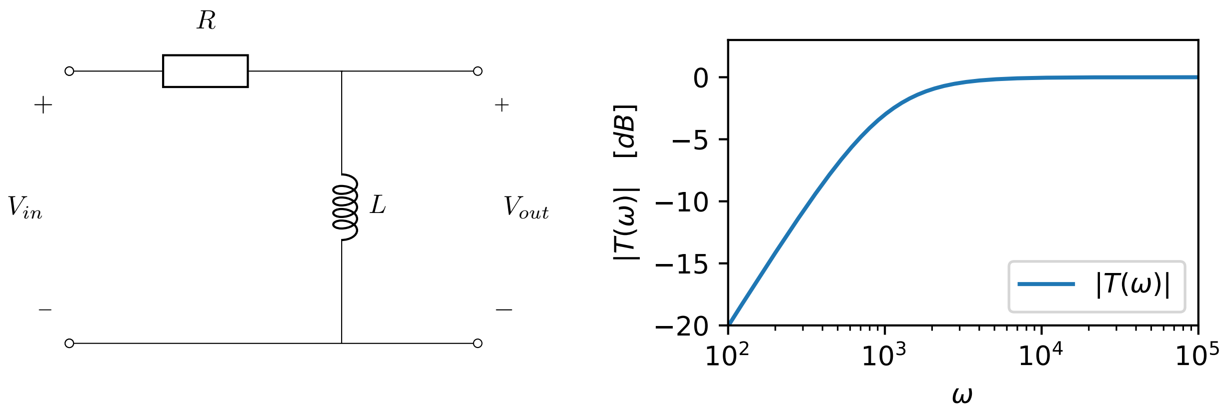
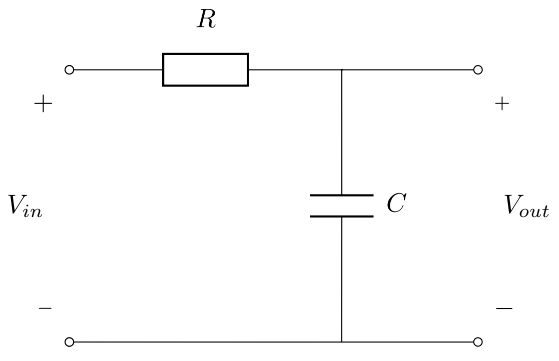


Figure 1: Circuit schematic (left) and bode magnitude plot (right) for a **RL high pass** filter circuit with $R = 10\,\Omega$ and $L = 10\,mH$. The magnitude of the transfer function is given by $|T(\omega)| = \frac{\omega/\omega_0}{\sqrt{1+(\omega/\omega_0)^2}}$, where $\omega_0 = R/L\ (= 10^3)$ is the angular cut-off frequency.



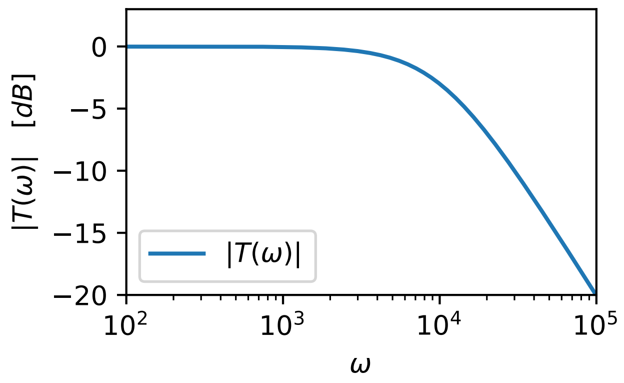
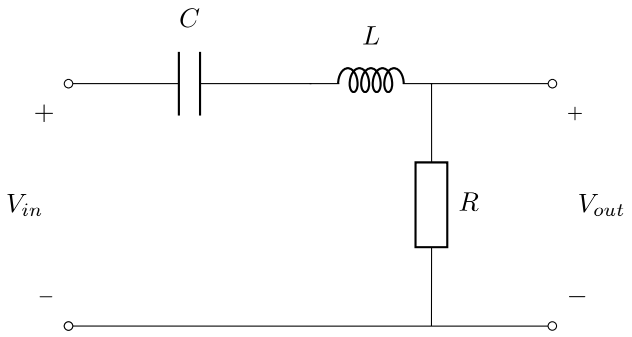


Figure 2: Circuit schematic (left) and bode magnitude plot (right) for a **RC low pass** filter circuit with $R = 10\,\Omega$ and $C = 10\,\mu F$. The magnitude of the transfer function is $|T(\omega)| = \frac{1}{\sqrt{1+(\omega/\omega_0)^2}}$ where $\omega_0 = 1/RC$.



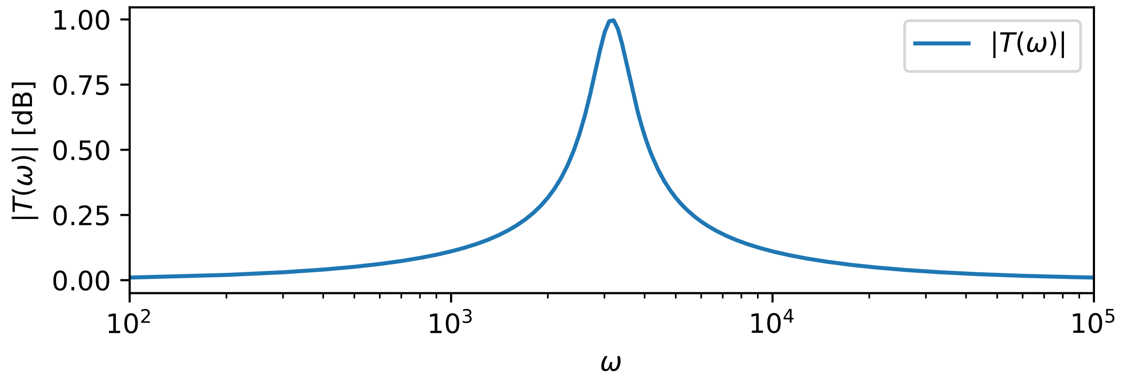


Figure 3: Circuit diagram (top) and transfer function (bottom) for a **RLC band pass** filter circuit with $R = 10\,\Omega$, $L = 10\,mH$ and $C = 10\,\mu F$. For the sake of clarity, logarithmic scales have not been used. The magnitude of the transfer function is $|T(\omega)| = \frac{R}{\sqrt{R^2+(\omega L-1/\omega C)^2}}$.

4 Interacting with the black box

Throughout assignment 2, you will be required to pass time-domain input voltage signals to the various 'Boxes' objects defined in the first code cell of `assignment2.ipynb`, and analyse the resulting output. This is done by invoking the `.process` method on each box. For example, the call `example_box.process(t, s_in)` passes a signal, whose amplitude at times in `t` is given by `s_in`, to `example_box`, and returns an array containing the amplitude of the resulting output signal. The use of `.process` is demonstrated in `black_box_example.ipynb`. You should run through that example notebook *before* attempting the assignment.