Laboratory 4 – Analysis of a Low-Pass Filter

In this laboratory we analyse the frequency response of a LC low-pass filter. A passive low-pass filter contains only reactive components (capacitors and inductors).

All the power delivered to the load resistor (R_L) comes from the source V_S (i.e., the circuit can only attenuate the power from the source). Since capacitors and inductors do not dissipate power, all the power from the source is dissipated on the load (P_S=P_L).

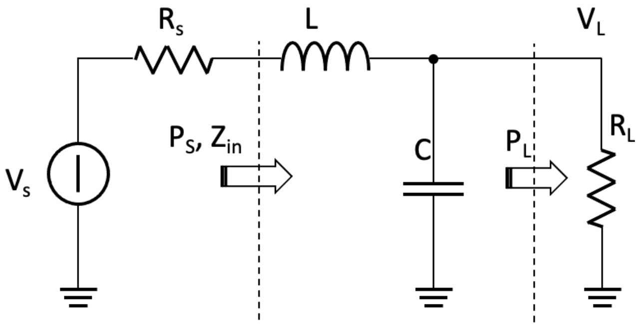

Figure 1 - Low Pass filter

The circuit can be analysed in the frequency domain $\omega = 2\pi f$ [rad/s] using complex numbers. In particular, we use the complex impedance Z_L for the inductor and the complex impedance Z_C for the capacitor using the following formulas (j is the complex unit):

$$Z_L = j\omega L \qquad\qquad Z_c = \frac{1}{j\omega C}$$

Description

Part 1 – Complex numbers

First, we implement a set of functions operating on complex numbers. A complex number is defined by the following declaration (note that this a declaration for a type, not a variable):

```
typedef struct {
    double real;
    double imag;
} complex;
```

Having this declaration is possible to instantiate complex variables using the typedef definition. For example, the following statement declares a complex ZL and initialses the real and imaginary part:

```
complex ZL;
ZL.real = 50; ZL.imag = 0;
```

To use the complex numbers, it is convenient to write a set of functions that implements the basic arithmetic operations. The table below lists all the operation required for the program in this laboratory.

Function Prototype	Description
`complex add(complex a, complex b);`	receives two complex numbers a, b and returns the sum a+b
`complex mul(complex a, complex b);`	receives two complex numbers a, b and returns the product a*b
`complex cnj(complex a);`	receives a complex numbers a, b and returns its complex conjugate
`complex div(complex a, complex b);`	receives two complex numbers a, b and returns the ratio a/b

Write the program that tests part 1 through the following steps:

1. Declare and initialise the complex numbers a=100+j50 and b=-50+j100
2. Implement all the functions listed in the table above. The function implementation should be consistent with the prototype provided by the table.
3. Test the functions using "a" and "b" as input and printing the results.

Expected output:

```
a =      100 + j*50
b =      -50 + j*100
a + b =  50 + j*150
a * b =  -10000 + j*7500
cnj(a) = 100 + j*-50
a/b    = 0 + j*-1
```

Part 2 – Analysis of frequency response

The parallel Z_P and series Z_S between two (complex) impedances Z_1 and Z_2 are given by the following formulas:

$$Z_P = \frac{Z_1 \cdot Z_2}{Z_1 + Z_2} \qquad\qquad Z_S = Z_1 + Z_2$$

These formulas can be used to calculate input impedance Z_{in} , that is the impedance looking toward the from the source. This is simply a sequence of parallel and series:

$$Z_{in} = series\left(Z_L,\ parallel(Z_C,\ R_L)\right)$$

Clearly, the reactive components (Z_C and Z_L) change their impedance according to the frequency. A for-loop where the frequency is varied should be implemented, and the angular velocity ω should be calculated at each frequency to update the complex value of Z_C and Z_L.

We use a range of frequencies between 100Hz and 1kHz with a step of 100 Hz. Once the Zin is known, we can show that the power gain (P_{out}/P_{max}) can be evaluated as:

$$G = \frac{P_{in}}{P_{max}} = \frac{4R_S Re\{Z_{in}\}}{|R_s + Z_{in}|^2}$$

Here P_{max} represents the maximum power attainable on the load which is reached achieved when Zin = RS. The power gain should always be less than 1 (it is an attenuation!) and the frequency behaviour should exhibit a low-pass characteristic, i.e.. the attenuation increases as the frequency increases.

Write the program through the following steps:

1. Implement the function *parallel()* and *series()* receive two impedances (complex number) as input and return respectively the parallel and series impedance (complex number). Test the functions before using them.
2. Write the program that receives from keyboard the following inputs:

- The load resistance R_L in Ω
- The source resistance R_S in Ω
- The capacitance C in F
- The inductance F in H

3. Calculate the impedances of the reactive components at each frequency
4. Calculate the gain and print correctly formatted. In order to print in scientific notation use the printf() with the formatter %E.

Expected input/output:

```
RS> 5.00E+0
RL> 5.00E+0
C> 1.00E-06
L> 2.00E-0

Zin=5.00E+01+j*1.10E-01

Freq[Hz]    Gain
1.00E+02    9.88E-01
2.00E+02    9.54E-01
3.00E+02    9.01E-01
4.00E+02    8.35E-01
5.00E+02    7.62E-01
6.00E+02    6.87E-01
7.00E+02    6.14E-01
8.00E+02    5.44E-01
9.00E+02    4.80E-01
1.00E+03    4.23E-01
```