Passive Filtering in MATLAB

In this lab our goal is to learn the function of passive filters in AC circuits through theoretical analysis in MATLAB.

1 Introduction to MATLAB

MATLAB stands for Matrix Laboratory. It is an important tool for creating/viewing/playing with signals which are essentially just 1-D matrices. It does lots of really cool stuff, a small portion of which will be demonstrated below. For getting started with MATLAB you could either download and install it from MathWorks website, or just use the online version. For both options you have to sign up on MathWorks using your NJIT email address.

a) Download and install: https://www.mathworks.com/downloads/

b) Use the online version: https://matlab.mathworks.com/

Open MATLAB and familiarize yourself with different parts of the environment. Try finding the Editor (probably containing one tab named "untitled.m" or press Ctrl/Cmd+N), the Command Window and also the Workspace (which would be empty in the beginning).

1.1. In MATLAB's the command window (usually at the bottom of the screen) type `help sin`. It tells you how to use the sin function. Neat, huh? Now type `doc plot`. Even better! `help` and `doc` are two of the most important MATLAB commands to know.

Based on the documentation, briefly explain what do `sin` and `plot` functions do in 2 sentences.

1.2. Since this is a digital computer, everything we do is discrete, as opposed to continuous, which is a difference that will be discussed at great length, but basically it means that our signals will be defined over a finite set of indices, rather than being a continuous function of some independent variable.

To create that set of indices, type: `t = [0:0.001:10];`

This creates a vector of time that goes from 0 to 10 in increments of 0.001. You should also see `t` appear in the "Workspace" portion of the MATLAB window. Also, note the semicolon at the end of the command. Unlike C, this is not necessary, but it suppresses the resulting output of that command, which is nice. Try it without the semicolon and you'll see what I mean.

1.3. Now for the first signal. Type `x1 = cos(t);`

`x1` should show up in the Workspace (usually on the left or right side of the screen).

1.4. Let's take a look at the signal using the plot command: type `plot(t, x1)`.

Include this plot in your report.

1.5. Let's mess with our signal some. `x1` is a vector (a list of numbers), as opposed to a scalar (one number). MATLAB let's us be kind of sloppy, sometimes. Let's halve the amplitude of x1 and shift it up by 1. The command, as you may have guessed, is `x2 = x1/2 + 1;`

Seems obvious at first, but can you do that in C? Nope.

1.6. Plot `x2` in the same way you plotted `x1`.

1.7. Let's make yet another signal. `x3 = exp(-t/2);` makes a decaying exponential with a time constant of 2.

Plot `x3` against `t`.

1.8. When multiplying a vector by a scalar, MATLAB cleans up the slop for you. However, try creating `x4` as the product of multiplying `x1` and `x3` together using the `*` operator. Doesn't work. In MATLAB, the normal operations are for matrices. If you want to operate element-by-element, you need to use the dot operator.

Now try `x4 = x1 .* x3;` and plot it. You should be staring at a decaying sinusoid.

Plot `x4` against `t`.

So there's some of the most basic things you can do with signals in MATLAB. Notice: no `for` loops. Using `for` loops in MATLAB is way slower than using matrix operations and should be avoided where possible.

Here are some other useful things to try:

a) At the prompt, press the up arrow. Press it again. It goes back through your command history. This is a great way to save time.

b) Type `plo` and then press up. Press it again. You can constrain your trip backward through your command history to commands that started with a certain string.

c) Type `size(x1)` (no semicolon). This tells you the size of any variable. The size is defined as the number of rows (i.e., the vertical size), then number of of columns (i.e., the width of the matrix). If there are third, fourth, and higher dimensions, their sizes will follow these two numbers. Now type `size(x1.')`. The numbers should be switched. What you've done is take the transpose of the matrix and then found its size.

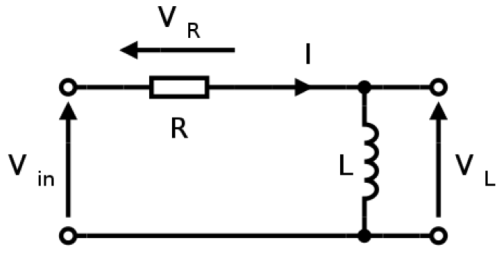
N.B. The dot in front of the transpose operator is different from the other dots we've used so far. Without it, the conjugate transpose will be computed, which means that the matrix will be transposed and every element will become its complex conjugate. `x1` is all real numbers, so here it doesn't really matter, but it is good to get into the habit of using the dot transpose operator unless you explicitly want to take the conjugate transpose.

d) Type `length(x1)` and `length(x1.')`. This is equivalent to `max(size(x1))` and should be used carefully. I've gotten into trouble in the past because of the non-specificity of the length command.

e) Wanna get rid of a variable? Type `clear x2`. Gone. Don't type it just yet, but `clear all` gets rid of every variable in the workspace. `close all` will close any figures that are open.

2 Passive RL Filter

In the RL circuit below assume V_{in} is an AC signal, the input to the circuit, and V_L is the output.



2.1. First, the math. Write down the equation of V_L as a function of V_{in}.

2.2. NOW, let's use MATLAB to plot the transfer function of our RL circuit. First, a few cosmetic steps: let's clear our workspace, close all figures and clear our command view. Write the following inside the Editor window in a new file.

```
clear
close all
clc
```

Let's define our resistor $R = 100\Omega$ and our inductance $L = 10mH$ in units of Ohm and Henry, respectively.

```
% These parameters give tau = 100 us
R = 100;       % In Ohm
L = 10e-3;     % In Henry
```

The cutoff frequency of this circuit, as you'll hopefully remember by now is this:

```
fc = R/(2*pi*L);
```

Let's define the magnitude of our voltage source.

```
vS = 1;     % Voltage source, in units of Volt
```

As well as the frequency range we'd like to look at:

```
f = 10 : 1 : 100e3;     % Frequency in Hz
```

Also, define our impedances ...

```
ZR = R;
ZL = j*2*pi*f*L;
```

And finally, our voltage dividers:

```
vR = vS * (ZR ./ (ZR+ZL));
vL = vS * (ZL ./ (ZR+ZL));
```

Let's plot it out:

```
figure(1);
plot(f, abs(vL))
```

And while we're at it, let's also plot the Bode plot. Here we will be using `semilogx` instead, which is very similar to `plot` with only one difference that the x-axis is plotted in a logarithmic scale.

```
figure(2);
semilogx(f, 20*log10(abs(vL)))
```

Press the F5 button to run your code and see the plots. Use the the commands `xlabel` and `ylabel` to label the axes in both figures and write their units.

From the File menu save the plots in PNG or JPEG format. Include both plots in your report.

2.3. Based on the plots, what type of filter is this? Explain why.

2.4. Theoretically, calculate what should the magnitude of V_L be at the cutoff frequency f_C? Mark this value on both plots using the "Data Cursor" or "Data Tip" and include the marked plots in your report.

2.5. Earlier we used the `abs` function to plot the magnitude of V_L at different frequencies. To complete the Bode plots we also need to draw the phase of V_L. The `angle` function would give us the phase in Radians.

Use the `angle` function along with `semilogx` to plot the phase of V_L in unit of degrees. Note that you would have to convert Radians to degrees.

BONUS: Read the documentation on `subplot` and plot the phase diagram below the magnitude diagram in the same figure.

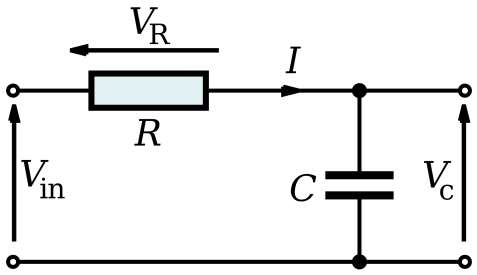
2.6. Theoretically, calculate what should the phase of V_L be at the cutoff frequency f_C? Mark this phase on the plot from previous step and include the marked plot in your report.

2.7. Imagine you need to design a passive RL filter with a cutoff frequency of $f_C = 3.2kHz$. With an inductor $L = 1mH$, what resistor R should we use to obtain the desired cutoff frequency?

2.8. Verify your calculations by adjusting your MATLAB code for the new R and L values and plotting V_L in linear units and also Bode plot (both magnitude and phase). Mark the cutoff frequency.

3 Passive RC Filter

In the RC circuit below assume V_{in} is an AC signal, the input to the circuit, and V_C is the output.



3.1. Write down the equation of V_C as a function of V_{in}.

3.2. What type of filter is this circuit? Using $R = 1k\Omega$ and $C = 30nF$, calculate what would the cutoff frequency be?

3.3. Write the resistor and capacitor values in your MATLAB code. Replace the line where you calculate Z_L with the proper code to calculate Z_C. Also, edit the line where V_L is calculated appropriately to calculate V_C instead. Make sure to rename `vL` to `vC` in all lines where you plot the transfer function.

Plot the transfer function of the RC circuit in linear units and also Bode plots. Mark the cutoff frequency and verify your theoretical calculation.