**Abdullah Aljandali**

**EE 380 - Project 2**

**3/21/2019**

**Problem Description:**

Create a GUI program that displays bode plots and circuit schematic of user-designed filters. Filters can be RC, RL or RLC. For RC filters, the user must input two of R, C or fc along with Low/high. For RL filters, the user must input two of R, L or fc along with Low/high. For RC and RL filters, the program should be able to calculate the third parameter using the 2 inputs. For RLC filters, the user must input three of R, L, C or fc along with Pass/Stop.

**Project Design:**

A. Frontend Design:

We will start by designing the frontend to be as user friendly as possible, and alter the backend depending on it, instead of the other way around.

First of all, we will have a Combo Box for the user to select what kind of filter they want. It can be one of:

1. RC Low Pass
2. RC High Pass
3. RL Low Pass
4. RL High Pass
5. RLC Bandpass
6. RLC Bandstop

Then, depending on the user’s selection of filter, we will ask the user for parameters. We will have text boxes for:

1. Resistance
2. Inductance
3. Capacitance
4. Cutoff Frequency 1
5. Cutoff Frequency 2

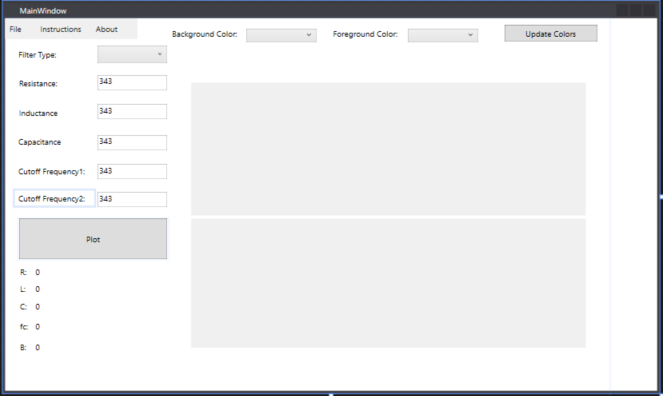
In the case of RC filters, we will only have R, C and fc1. The RL filters will use the same text boxes, except that it will use L instead of C. As for the RLC filters, it will use all of the text boxes. For RC and RL filters, cutoff frequency1 is the cutoff frequency for the filter, and cutoff frequency2 is not used. Though, RLC filters require 2 cutoff frequency that they either pass or stop what’s in between these frequencies. Therefore, only RLC filters will use both cutoff frequencies. To avoid the user entering the wrong parameters for the filters, we will disable some of the text boxes depending on which filter is being used.

The frontend will also contain 4 labels, R, L, C, fc, B that are different from the text boxes that the user enters. The reasoning behind these labels is that the user need not enter all parameters, as some can be calculated by the program. Thus, these labels display the values after being calculated by the program. The B in this case displays the width of the passed or stopped frequencies.

There will be two graphs, one for the Magnitude plot, and one for the Bode plot. Moreover, there will be an image that is dynamically inserted depending on the filter selected. The image shows the circuit diagram of the selected filter.

We will use 2 combo boxes for the user to select background and foreground colors for the plots.

There will also be two buttons. “Plot” button will be used to plot the graphs once the user has entered in their desired values. Secondly, an “update colors” button will be used to update the colors of the plots once the user has changed the colors selected in the colors-combo-boxes.



B. Backend Design:

First, I designed my filters on paper before putting it in the code. Using the ideas that an inductor acts as a short in low frequencies and as an open in high frequencies, and that the capacitor is the opposite way, I designed circuit diagrams for all six filters. Using these circuit diagrams, I found the transfer function for each filter. I used the transfer function to plot the magnitude and phase bode plots. Please see next page for details.

Moving on to the coding part, because the user can select between six filters, the filtering has to change depending on this selection. The program fires an event whenever a selection change occurs. I use this event to view the new selection, and irrelevant text boxes based on it (For example, it disables Inductance for an RC filter).

Whenever the “plot” button is clicked, the program reads the user inputs, calculates missing parameters, inserts circuit diagram, and plots the bode plots. The missing parameters are calculated in a function using formulas depending on the type of filter (as seen in the next page). The program saves a sample circuit diagram for each filter and inserts one of them corresponding to the user’s selection. As for the plots, I used a canvas and added a Windows Forms chart to it. The x-axis was made logarithmic to make the plot a bode plot, and the axes were dynamically calculated depending on the cutoff frequency of the filter. The program then loops on frequencies and uses it to calculate magnitude and phase and plot the graph.

The last event that could be fired is the change color event. It is fired whenever the user clicks the “update color” button. The way this works is that I create a Color using the string name of the selected color in the combo box.

**Experimental Design:**

To test the program, I used a trust online filter design tool to compare its results to mine. Here is the link for that tool: http://sim.okawa-denshi.jp/en/Fkeisan.htm

I used that tool to test the filters one by one. I entered the same parameters for both the online tool and my filter, and compared the circuit diagram, the calculations of the missing parameters, and the bode plots. For every filter I did 2-3 tests. Each of these tests had one missing parameter. Then I checked to make sure that all of these 3 tests give the same results and also are similar to the online tool.

**Theoretical Results:**

Based on the design I made, I expect the project to behave like what the problem statement states. In other words, I expect every filter to do the math correctly as I would do it by hand (except not as fast obviously 😉). I also expect the calculations of the missing parameters to be correct.

**Measurement Results:**

The program behaved like it is expected to. I compared the results with the results of the online trusted tool and they matched. I also did the math on paper and it is like the results of the program. I specifically compared the plots, circuit diagram, and the calculations of the missing parameters.

The only difference between my program and the online program is that my graphs were less smooth/precise. The reason for this was that I was not using as many sample points as they were. The issue was that the ranges of the graphs can go all the way from 1 to 1 \* 10^8. To help solve this, I first reduced this range by making it dynamically allocated depending on the cutoff frequency. This means I only allowed three decades before fc, and three decades after. To further solve the problem, I made my for-loop logarithmic. This the sampling frequency doesn’t increase by a constant, but rather depends on itself. To do this, I allowed for the same number of samples between every decade. Meaning if 10 samples are taken between 1-100hz, only 10 samples will be taken between 100-1000 samples. This reduced the run time by a lot, and at the same time provided reasonably precise plots.

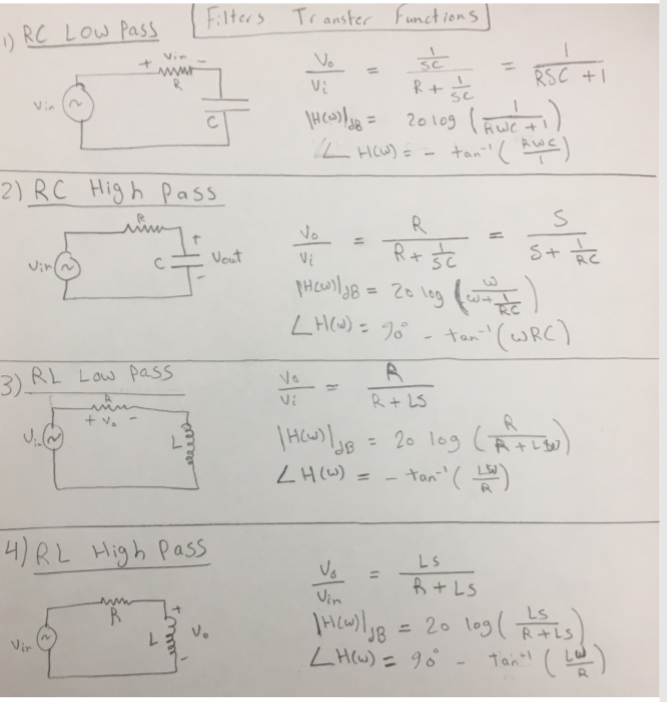
**Conclusion:**

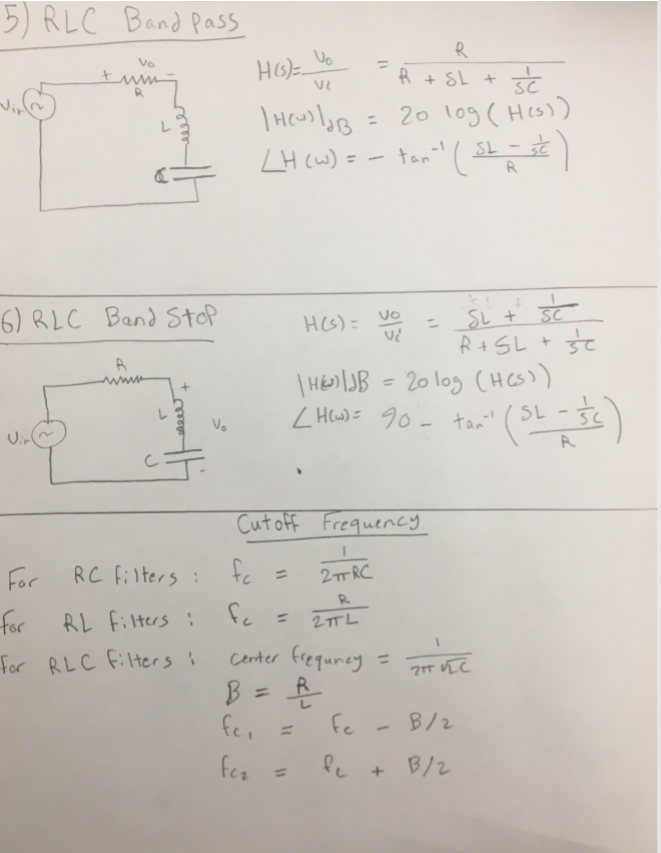
The project satisfies the minimal requirements with minimal error. There are also novel features that have been added (as discussed later). The project as whole has been a good learning experience as the student must have deep understanding of multiple areas in order to complete it. First of all, the student most have good GUI design skills, strong C# programming and problem skills, and be able to do independent research. Also, a big part of the project was understanding circuits and signals. For the student to make a program that filters waves, they need to understand how different filters work, and be able to design filters.

**Novel Features**

The project included three additional features that are not captured in the problem statement:

1. Instruction: The GUI provides a button for the user to view instructions on how to use this tool.
2. About: The GUI provides an about button explaining what this tool is about
3. Color Change: There is two buttons combo boxes to change the colors of the background and foreground.
4. Reset choice: The user can select “reset” in the filters list to reset the graphs.
5. Value calculation display: The GUI displays the calculations of the missing parameters for the user to use as verification
6. Disables irrelevant text boxes: The GUI disables irrelevant text boxes in order to make the tool more user friendly and avoid issues.
7. Dynamically calculated axes: The axes are dynamically allocated in the code depending on the user-inputted values.

****

****