Designing and Implementing RLC Circuits in MATLAB

Robert Jacobs
Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, MO, United States
robert.jacobs@wustl.edu

Max Cusick
Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, MO, United States
c.marguerite@wustl.edu

Ben Ko
Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, MO, United States
ben.k@wustl.edu

***Abstract*—This case study focused on designing and implementing a RLC circuit in MATLAB. We developed a RLC circuit from scratch and then applied it to a variety of tasks. By slightly modifying our base circuit, we made three new circuits, each for a specific task. We then modified our existing circuit to act as a tuning fork, a circuit which can clean up audio of a helicopter on mars, and a circuit which can clean noise out of a music file. We were able to successfully make circuits to accomplish each of these tasks through our trial and error method of finding RLC values.**

I. Introduction

This case study focuses on designing a linear dynamical model for implementing circuits. The goal of the case study is to explore how the circuits respond to different inputs and to design an RLC circuit to operate as a resonator, sensor, and audio filter. We first explored circuits through their components such as capacitors and inductors. Then we took what we learned to design a basic RLC circuit. Finally we modified our basic RLC circuit for a variety of tasks.

II. Methods

A. Task 1, Modeling an RC Circuit

For task 1.1, we modeled a resistor-capacitor circuit using Ohm's Law and Kirchhoff's Voltage Law. To model the circuit we defined an index k to represent discrete time steps in terms of h seconds as the following:

$$k = \frac{t_k}{h}$$

Using Kirchoff's voltage law (conservation of energy) and Ohm's law we found two additional equations below where vR and vc are the voltages across the resistor and capacitor and i is the current through the resistor and capacitor.

$$v_{in} - v_R - v_C = 0$$

$$v_R = i_R$$

We put these two equations together and derived two equations below:

$$v_{R,k} = v_{in,k} - v_{C,k}$$

$$v_{C,k+1} = \left(1 - \frac{h}{RC}\right) v_{C,k} + \frac{h}{RC} v_{in,k}$$

The first equation shows voltage across resistor at time index k in terms of voltage input and voltage across capacitance. The second equation is a linear dynamical model of the voltage at time index k+1 written in terms of voltage across capacitance and voltage input at time index k.

We used following values for variables for simulation of circuit A in MATLAB:

$$R = 1\,k\Omega$$

$$C = 1\mu F$$

$$v_C = 0\,V \; at \; t = 0$$

$$v_{in} = 1V \; for \; t > 0$$

Under constant 1 V input, and h value of 110-5, we were able to get the following output for the voltage measured across a capacitor in response to a constant 1 V input.

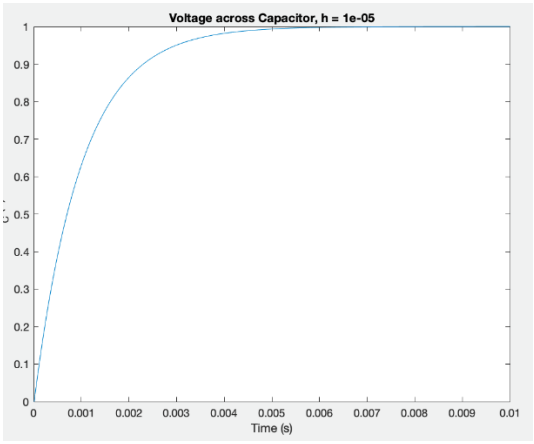

Figure 1: Voltage measured across a capacitor in response to a constant 1 V input

For the theoretical changing curve, we used the following equation:

$$v_C(t) = 1 - e^{\frac{-t}{RC}}$$

While testing, we found that our most accurate graphs were outputted when h was less than or equal to 1.0*10^-4. When we made an "inaccurate" choice for h, i.e, high h value, we observed a "spike" in the plot where voltage is shown to be steady until it goes either straight up or straight down. The charging voltage of a real capacitor would not change with different values of h since it would retain physical properties, but since we based our implementation off of values of h, our simulated output changes with it.

For a timestep that was small enough to observe the actual change of the voltage across capacitor, we could see how voltage across capacitor stayed constant once it hit the input voltage of 1 V.

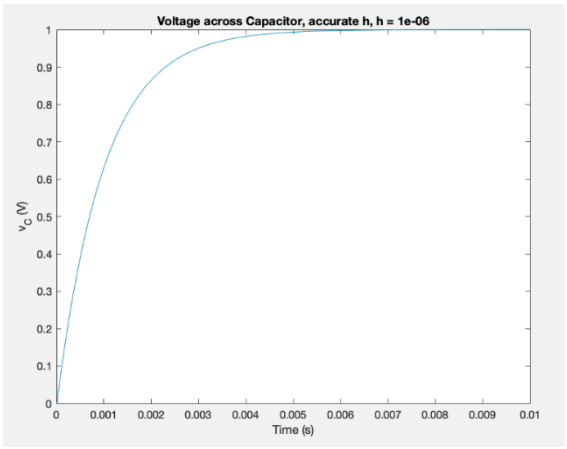

Figure 2: "accurate" choice of h

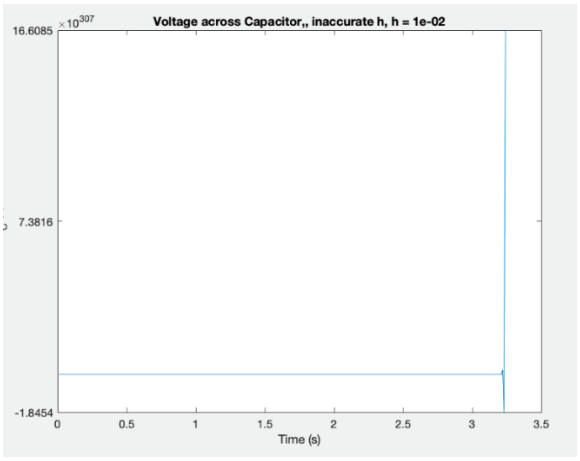

Figure 3: "inaccurate" choice of h

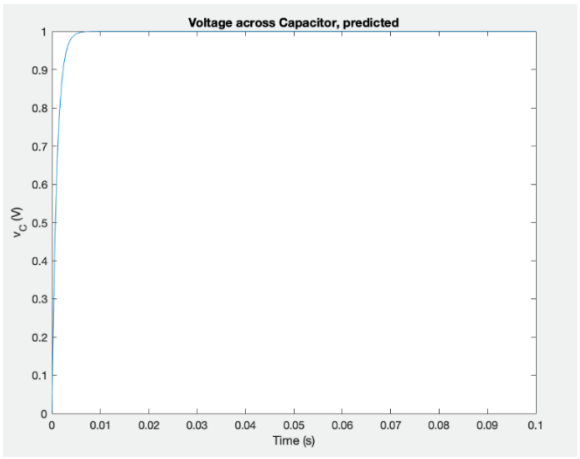

Figure 4: the theoretical changing curve

Relative to the charging curve of the capacitor, we would interpret the meaning of RC time constant of circuit,

$$= RC$$

as the amount of time required to change the charge of the capacitor.

B. Task 2, Modeling and RL Circuit

For part 2 we expanded on our circuit from part 1, this time adding an inductor, an electrical component that stores potential energy in a magnetic field. The inductance quantifies how much magnetic flux (the surface integral of the magnetic field through a cross-section of the inductor) is generated by a current flowing through the inductor. The inductance of the

inductor could be expressed by the following equation where L stands for inductance, B, and i stands for a current.

$$L = \frac{\Phi_B}{i}$$

We derived a linear dynamical equation for current across the inductor i_k below.

$$i_{k+1} = (1 - \frac{hR}{L})i_k + \frac{h}{L}v_{in,k}$$

Using the following equation, we can calculate the current within the inductor for any input voltage vector and initial condition.

For the simulation of the circuit on MATLAB, we used the values of R = 100 and L = 100 mH and for our initial condition, set i = 0 A, at t = 0 and constant input voltage as 1V. Figure 4 is the plot from the simulation.

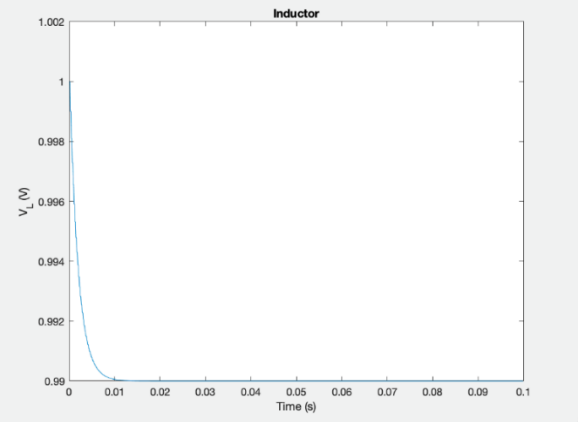

Figure 5: Voltage measured across an inductor in response to a constant 1 V input.

We compared our capacitor circuit to our inductor circuit and observed them having complementary characteristics. As observed in Figure 5, once the capacitor reaches steady-state voltage as the voltage across it goes to equal 1 V. The inductor displays a complementary characteristic reaching steady-state voltage as the voltage across it goes to 0 V. It can also be seen that the summation of voltage across inductor and capacitor always equals the input voltage.

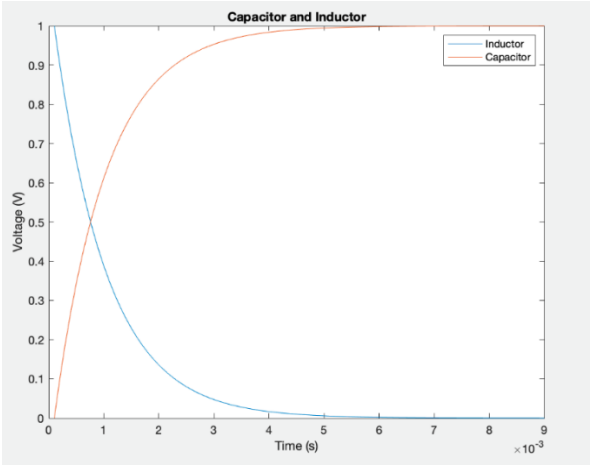

Figure 6: Voltage change over time for inductor and capacitor for an input voltage of 1 V

It can also be seen that once steady-state voltage is reached, the steady-state current goes to 0 C/s in both the inductor and capacitor. Since the capacitor and inductor reach a steady-state, there is no more change in the voltage across them and therefore no current in the circuit after the steady-state is reached.

C. *Task 3.1, Deriving an RLC Circuit*

We used three equation to derive a linear dynamical equation for voltage and current across circuit C. The first equation we used was a linear dynamical equation on voltage over capacitor:

$$v_{C,k+1} = (1 - \frac{h}{RC}v_{C,k} + \frac{h}{RC}v_{in,k}$$

The second equation we used was Ohm's Law equation across the resistor.

$$v_R = iR$$

The third equation is a linear dynamical equation on current across the inductor.

$$i_{k+1} = (i_k + \frac{h}{L}v_{L,k}$$

Since x_k was a transpose of the concatenation of i_k and $v_{c,k}$ we made the first row of the A matrix to be $v_{c,k}$ and the second row i_k .
Using these three equations, we derived the following linear dynamical equation for voltage and current across circuit C:

$$x_{k+1} = \left[(1)\left(\frac{h}{c}\right); \left(\frac{-h}{L}\right)\left(1 - \frac{hR}{L}\right)\right]x_k + [(0); \left(\frac{h}{L}\right)]v_{in,k}$$

D. *Task 3.2, Modeling an RLC Circuit*

For task 3.2, we took the equation we derived in 3.1 and iterated it using MATLAB. This was done by plugging in the A and B matrices to create a linear dynamical system that could adjust the values of x, a vector made to store the

adjustments to voltages at a specific time step. The different oscillations generated by the equation were then plotted against each other for comparison.

As we implemented our original equation, we noticed it was flawed since it would not generate the expected output. We fixed this through trial and error and AI meetings which led to us changing both how we implemented our equation in MATLAB, and the equation itself.

After correctly implementing the equation, we made three different plots by tweaking the R, L, and C values we plugged into the simulation. These three different outputs were then played as sound through the use of soundsc(). The first function produced a steadily oscillating wave by leaving our current and inductance the same and halving our original resistance. This resulted in a quick and steady sound. Then we made an increasing function by decreasing the current over a factor of ten and slightly increasing the resistance. This resulted in a slightly louder and sharper sound. Finally, we made a decreasing oscillation by using our original current and slightly increasing both the resistance and inductance. This resulted in a lower, less sharp sound. All three functions started from 0 and then oscillated around 1V.

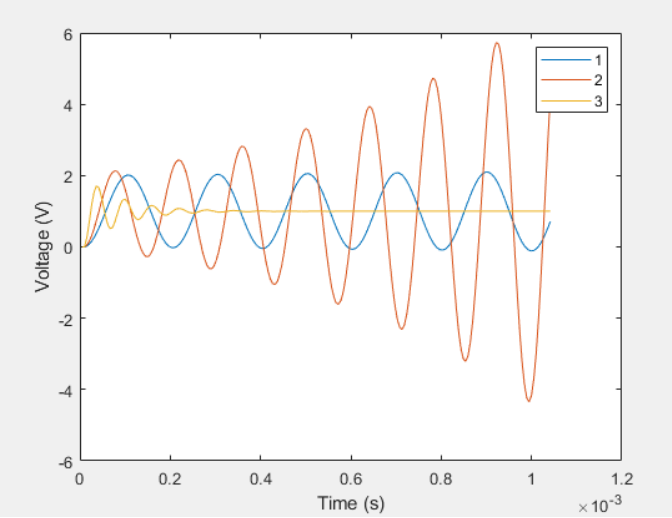

Figure 7: Oscillations from the RLC circuit.

E. *Task 3.3, Implementing A Sinusodial Response*

For task 3.3, we took the circuit model we made in part 3.2 and implemented the sinusoidal function $v_{in}(t) = sin(2\pi ft)$. The new input values created a different graph than previously. Next, we began testing how different frequencies affected the graph. As expected, we saw that the higher the frequency, the more waves (oscillations) occurred during the specified time. Additionally, as frequency rose, oscillations became faster with smaller and smaller amplitudes. As the frequency rose, the pitch of the sound generated became higher. For example, a frequency of 100 produced half of a sine wave (looked like a concave down quadratic) while a frequency of 1000 produced a steady oscillating input. We observed that a frequency of 200 had a full sine wave and 250Hz had a wave and a half. This suggests that there is some proportionality in this function. When testing different frequencies, it became apparent that similar frequencies had very similar amplitudes. We would consider our circuit a high-pass filter because we had great success hearing the high-frequency sounds whereas the low-frequency sounds were hard to hear or sounded like a short beep.

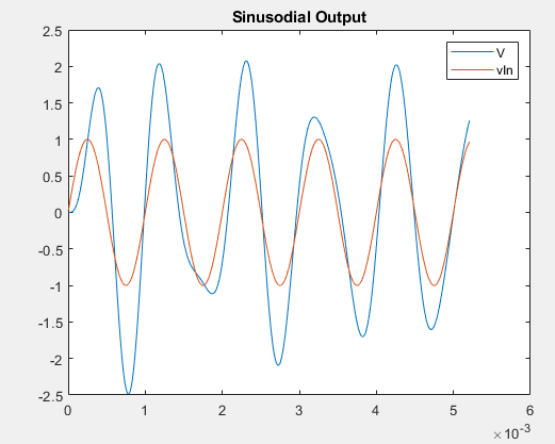

Figure 8: Sinusoidal output of the RLC circuit.

F. *Task 4.1, Making a Resonator Circuit*

The first task of part four was to recreate a tuning fork using RLC values. We used our equation from part 3 and experimented with different RLC values. Our strategy mainly consisted of trial and error. Our testing was primarily done by modifying a value at a time and seeing how it affected the result by playing the generated tone for a tone detecting mobile app. This gave us a good idea of what tones we were generating so long as they were high enough frequency for the app to pick up and report. As we iterated on the values, we would compare the current output to the expected output and use that to influence the RLC values. We decided to make our tuning fork output a frequency of 523.2511 Hz or a C_5 note. In our testing to produce this note, we found that we got the best results by leaving the capacitance at 1.0*10^-7 and adjusting the resistance and inductance. After some trial and error, we found the best resistance value to be 0.5*10^2 and the best inductance value to be 0.9255. Our base RLC circuit was the same as we used in task 3.2 with the time modified to be much longer so the tone would play for 5 seconds.

G. *Task 4.2, Making a Sensor Circuit*

For the second task of part four, we used a similar approach as we did for task one. We started with a base case of around one for the starting coefficients of RLC (with proper units: Ex. 1e-6). We then iterated on the starting RLC values with trial and error till the 84Hz sound was isolated. Since we knew that the helicopter sound was at 84Hz we devised an RLC combination such that it reduced the "power" of the surrounding noise, hopefully isolating the 84Hz. Hearing a difference in the audio proved very difficult as nearly all the sounds were popping noises. We used a testing script to hopefully predict how the RLC values would affect the sound file. We ended up with the following values: R=700, L = 18e-1, C = 19e-7.

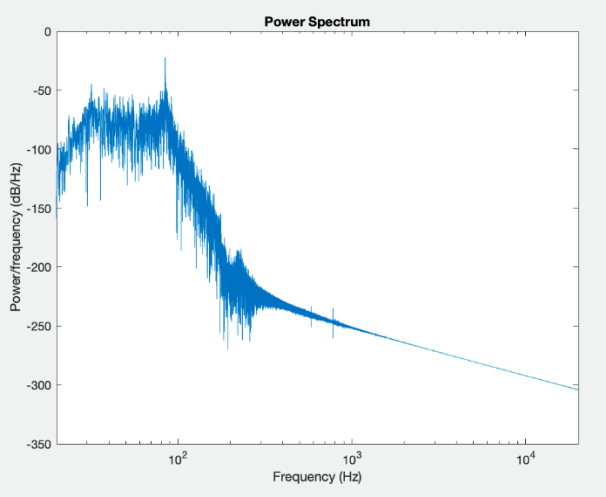

Figure 9: Filter power spectrum from circuit

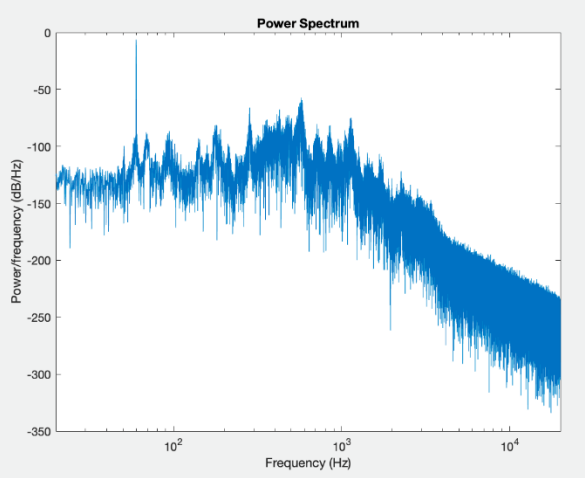

Figure 10: Filtered power spectrum from music

Task 4.3, Making a Filter Circuit

For the third and final task of part four, we were tasked with making a music filter to remove static noise (and plain unwanted noise) from the audio file. Our process was similar to tasks 1 and 2 in that our main strategy was using educated trial and error. To make our process more refined, we made a testing script, Testing.m, to provide a graphical representation as we tested different RLC values. The file mostly served as a way to test RLC values with our own plots outside of the ones provided by competitionTest.m. This was useful in better visualizing what we were filtering.

Our trial-and-error workflow began with figuring out how each RLC value affected the circuit and the resulting audio quality. Over time, we kept increasing each value and used the testing script to help model the changes. We found that the tolerances of the values are very dependent on each other. If we used large values for all numbers, it sometimes produced a very similar result to ones with generally small values. Eventually, we found the following values: R=1800, L = 12e-1, C = 0.05e-6. These RLC values gave us the best sound quality from our testing. There is room for improvement, but it's hindered by the required time investment. If we had gone by even smaller steps, our filter would most likely be better.

III. Results and Discussion

Our main results are the three circuits designed for tasks 4.1, 4.2, and 4.3. Of these circuits, we achieved modest results for all three. Our tuning fork successfully played a C_5 note for approximately five seconds. Our sensor appears to filter out most of the undesired noise from the helicopter audio. It's very difficult to tell how well when listening to the audio but from our best judgment, it's a big improvement over the native audio. Our music filter successfully filtered out most of the noise over the music while also leaving the music itself mostly clear. The main challenge for all three of these tasks was finding the correct R, L, and C values for the circuit. Our strategy of trial and error led to us finding reasonably working values, but was time intensive. Our strategy was improved slightly through using the Testing.m file to better understand how our modifications affected our results, especially with our music filter. Overall, our circuits were successful at completing their tasks, but still have room for improvement. In theory, we could develop strategies beyond trial and error for finding appropriate RLC values, and further improve steps like the derivation of our RLC circuit.

IV. Conclusion

Our work produced modestly successful results though there are still areas to be improved. Our resonator circuit, filter circuit, and sensor circuit work about as expected with some minor inaccuracies. For future projects of a similar nature, our testing methods could be improved to be slightly better than trial and error and educated guessing to find RLC values. We could also improve our circuits to be able to do multiple tasks such as filtering noise regardless of the audio file given or generating more than one specific tone. For now, though our circuits appropriately accomplish their required tasks while still leaving room for improvement. In conclusion, we have developed a successful base circuit and moderately successful testing method for finding RLC values.