Experiment Guide: RC Filters and LabVIEW

Objective

In this lab you will (a) manipulate instruments manually to determine the input-output characteristics of an RC filter, and then (b) use an instrument control system called LabVIEW (made by National Instruments, Inc.) to measure and plot RC filter characteristics automatically.

Background: RC Filter Characteristics

Figure 1 below shows an RC filter connected to a sinusoidal voltage source. This circuit is termed a two-port circuit (see Fig. 2) where the voltage source produces the input voltage V_{in} and the output voltage V_{out} appears across resistor R.

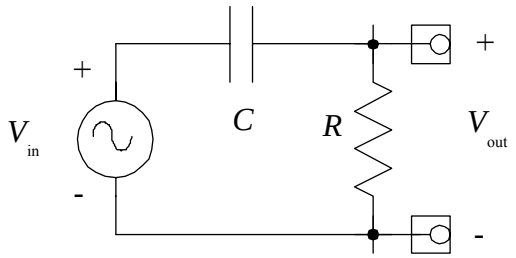

Figure 1. RC filter with series capacitor and output resistor R.

Figure 2. Two-port circuit.

Recall that we customarily represent an AC voltage as a periodic function of time such as $V(t) = V_0\cos(\omega t)$ where V_0 is the amplitude of the voltage, t is time, and ω is the so-called angular frequency, whose units are radians per second. The angular frequency is related to the "ordinary" frequency, f, measured in Hertz, by $\omega = 2\pi f$. For example, if the frequency, f, of the ordinary power line voltage in the U. S. is 60 Hz, then the associated angular frequency, ω, is 377 radians/s ($2\pi\times60$).

Background: Transfer Function

A two-port circuit is characterized by its so-called transfer function, whose magnitude is defined as $|\mathbf{V_{out}}/\mathbf{V_{in}}|$, where $\mathbf{V_{out}}$ and $\mathbf{V_{in}}$ are phasor voltages (as indicated by the boldface type). The variation of the transfer function with frequency characterizes the circuit, whether the circuit is an amplifier (does it amplify high frequencies more than low frequencies?) or a filter (does the filter pass the low frequencies or the high frequencies better?).

If you analyze the RC circuit of Fig. 1 using Kirchhoff's voltage law, the phasor voltages $\mathbf{V_{out}}$ and $\mathbf{V_{in}}$, the resistance R and the impedance of the capacitor $\mathbf{Z_C} = 1/j\omega C$, you can show that the magnitude of the transfer function is

$$\left|\frac{\boldsymbol{V}_{out}}{\boldsymbol{V}_{in}}\right| = \frac{\omega RC}{\sqrt{1+(\omega RC)^2}}$$

An approximate log-log plot of transfer function magnitude vs. frequency is shown in Figure 3:

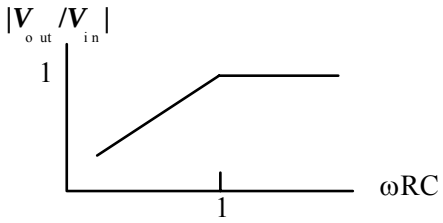

Figure 3. Log-log plot of transfer function magnitude vs. frequency times RC.

The filter characteristic has been simplified to appear as two lines that intersect at the angular frequency for which ωRC = 1, or ωτ = 1, where τ is the time constant RC for this circuit. If plotted precisely, the characteristic would transition smoothly from the upward sloping line to the horizontal line, but for many purposes the two-straight-line approximation is adequate. This circuit is called a *high-pass filter*, since for frequencies above ω = 1/RC the output voltage equals the input voltage.

If we reverse the positions of R and C in the filter circuit (Figure 4), we obtain the transfer function and filter characteristic shown below:

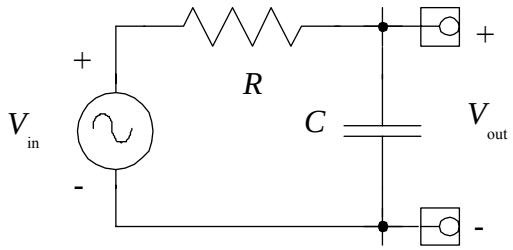

Figure 4. Circuit with a series resistor R and the capacitor C as the output element.

$$\left|\frac{\boldsymbol{V}_{\text{out}}}{\boldsymbol{V}_{\text{in}}}\right| = \frac{1}{\sqrt{(\omega RC)^2+1}}$$

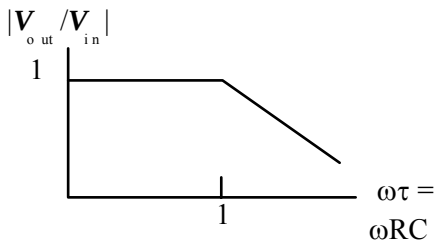

Figure 5. Log-log plot of transfer function magnitude vs. frequency times RC for the circuit of Fig. 4.

Lab Equipment

- Personal computer running Windows XP with LabVIEW 7.1 installed
- Printer
- 10kΩ resistor
- 0.1μF non-polarized capacitor
- HP 54645D oscilloscope
- HP 33120A function generator
- HP 34401A multimeter
- the file "RC Circuit.vi" on the EECS 40 website
- External Interface Command Set Manual for HP 34401A multimeter and HP 33120A function generator (also on EECS 40 website)

Procedures (Manual Plot)

P1. Connect a 10kΩ resistor and a (non-polarized) 0.1 μF capacitor in series with a signal generator, making sure that your oscilloscope ground and the signal generator ground are connected together. Set the signal generator to output a 1-volt peak sine wave. Measure and plot the amplitude of the voltage between the components versus frequency on log-log graph paper. You can download log-log graph paper from the EECS 40 website.

P3. Reverse the order of the two components and repeat.

P2. Observe the effects of filtering on square and triangular waves.

References (on Reserve for EE 40 in Engineering Library)

- P. Horowitz and W. Hill, *The Art of Electronics*, 2nd ed. (Cambridge U. Press, 1989), pp. 35-8.
- R. White and R. Doering, *Electrical Engineering Uncovered*, 2nd ed. (Prentice Hall, 2001). See p. 27 ff. for explanation of decibels, and pp. 285-7 on transfer functions and Bode plots.

Background: LabVIEW

Graphical circuit stimulation software, such as LabVIEW, is popular among engineers working in industry and researchers in universities because it reduces the tedium and cost of circuit and system testing. So far in this lab, you've used an analog function generator and oscilloscope to get the graph that shows the ratio of the voltages versus the frequency. Plotting the graph by hand is time-consuming and it may give inaccurate results. With LabVIEW, however, you can obtain accurate tabular and graphical results automatically after you program the system.

Note that your EECS 40 text (A. R. Hambley, *"Electrical Engineering: Principles and Applications"*, 3rd Ed.) discusses LabVIEW on pages 425-437, and contains a LabVIEW CD-ROM in the envelope inside the back cover of the book.

LabVIEW is a graphical programming language that shares some aspects with traditional non-graphical programming languages (C, BASIC, Pascal, etc.) and some aspects of hardware definition languages (VHDL, Verilog). It combines the generality and power of traditional programming data structures such as loops, if-then branches, and arithmetic operators with the ability of hardware definition languages to perform multiple tasks simultaneously.

Programming in a graphical environment consists of placing functional blocks that perform specific tasks on a worksheet and wiring them together to send data from one block to another. These blocks can do anything from simple tasks (add the data on the two input wires together and place the answer on the output wire) to complex tasks (take two arrays of data as input and display the contents on a log-log graph as x,y pairs). These functional blocks can also translate data in the graphical program into a form that external equipment can use. With the appropriate software drivers, any button or knob that can be pressed manually can be controlled automatically by one of these function blocks. Finally, certain special blocks can control the flow of a program by specifying that a few tasks should be performed in a certain order, or that a task should be repeated a certain number of times. All of these types of blocks are used in this lab.

In addition to placing blocks on the worksheet, blocks must be wired together. This is complicated by the fact that not all wires in LabVIEW carry the same kinds of data! Some wires will carry a single number. Other wires will carry a whole list of numbers. Other wires carry multiple kinds of data, where the amount and type of data are determined by the blocks to which they're connected. Unfortunately, most blocks require that the data coming in be formatted correctly, otherwise they will not perform their job. **Two of the biggest challenges people face when first starting to learn LabVIEW are deciding which type of wire to use where, and converting from one type to another. In this lab, you are provided with a pre-made LabVIEW graphical program, so you will not have to learn these aspects of LabVIEW programming today.** For an example of blocks wired together on a worksheet and a front panel of an instrument simulated in LabVIEW, see Figs. 9.22 and 9.23 in the Hambley textbook.

Procedures (LabVIEW)

P4. First, communication between the computer and the function generator and multimeter must be confirmed. To do this, first we must ensure everything is powered up and functioning. Log onto the computer. Turn on the function generator and multimeter, being sure to note the address of each one as it starts up. If these are different from one another, everything is good and so continue with the instructions here. If they are the same, you must change one of them (Menu on/off->I/O Menu->HPIB ADDR->^## ADDR).

On the computer, open "Measurement & Automation" (Start->Programs->National Instruments->Measurement & Automation). Expand the "GPIB0 (PCI-GPIB)" tab (My System->Devices and Interfaces->GPIB0 (PCI-GPIB)). Scan for Instruments by right-clicking on the tab and selecting the appropriate option. This should display all the instruments that are powered on, as well their GPIB addresses (GPIB stands for General Purpose Interface Bus, and it was developed originally at Hewlett-Packard). You can also use this program to manually send commands to the instruments over the GPIB bus; simply right-click on the instrument and select the "Communicate with Instrument" option in order to issue commands through the computer. A list of commands for each instrument can be found in the instrument manuals on the EECS 40 website, under the heading "Remote Interface Reference." The following are a few examples of the types of commands you can send to the function generator:

- **Applying Waveforms** – Enter the following command in the "Send String" field and click on the "Write" button:

  `appl:<shape (sin, squ, tri, dc)> <freq> <Vpp> <Voffset>`

  For example, to apply a 100 Hz, 3 Vpp square wave with a -2.5V DC offset, enter

  ```
  appl:squ 100 3.0 -2.5
  ```

- **Querying Parameter Settings** – Enter the following commands in the "Send String" field and click on the "Query" button. The result will display in the "String Received" box.

  - `func:shap?` → returns the current waveform shape
  - `freq?` → returns the current frequency
  - `volt?` → returns the current V_{pp}
  - `volt:offs?` → returns the current DC offset
  - `syst:err?` → returns the last error encountered

- **Other Fun Commands** – Enter the following commands in the "Send String" field and click on the "Write" button.

  - `*cls` → clears the error queue
  - `*rst` → resets the function generator to the default settings
  - `disp:text '<string>'` → display a string of up to 11 characters
  - `disp:text:cle` → clear any string previously displayed
  - `syst:beep` → make the function generator make that annoying beep

Develop and test a command string that sets the function generator to a 5 kHz, 0.5V_{pp} sine wave. **Record this command string.**

P5. Next, you need to understand how programs in LabVIEW work. To do this, you are going to "reverse engineer" (a.k.a. figure out) how an already-written program works. It probably won't be obvious how the program works when you first look at it, but if you take an analytical approach and spend ten minutes or so going over the block diagram, you should be able to figure it out. Here are some things that might help you figure out what's going on (it's not necessary to answer all these questions in your write-up):

- Right-click on individual blocks.
- Double-click on blocks. Do the names of the blocks help you figure out their purpose?
- Are there any blocks inside other blocks? Why do you suppose this is?
- Is there any sort of organization that you can find? For instance, does the diagram make more sense if you follow it from left to right than if you read it from right to left?

Open the file, "RC Circuit.vi" from the EECS 40 web site. Double-click on one of the user interface widgets on the Front Panel view to open the Block Diagram view. As a group, try to reverse engineer how the program works, keeping in mind that all blocks execute simultaneously unless placed within a program flow control block (such as a for-loop or film strip, both of which are used in this program). **Describe how the LabVIEW program works in your own words in just a paragraph or two.** If you know how to program, you can write this description in

pseudocode or the programming language of your choice. **Be sure your description includes how the for-loop works and makes explicit which events occur simultaneously and which events occur sequentially.**

P6. There are two Formula blocks in this program. The one outside the for-loop ("Number of loops") defines how many data points to collect, while the one inside the for-loop ("Frequency") determines the frequency to which the function generator will be set for each iteration of the loop. Verify that the formulae given will generate the correct number of data collection points and the correct frequencies. You can do this by *either* **re-deriving the formulae used** *or* **making a table of output values for a frequency sweep from, say, 1Hz to 10kHz with 3 steps per decade**. Note that if you were performing a frequency sweep manually, a data point would need to be collected for each entry in this table.

P7. Hook up a 10kΩ resistor and a (non-polarized) 0.1μF capacitor in series with the signal generator (the same procedure that you used earlier in this lab when you made measurements manually). Attach the probes from the multimeter across the capacitor. Optionally, you may use the oscilloscope to monitor the voltage waveforms during the test: connect the function generator to channel #1 and use channel #2 to monitor the voltage across the capacitor. Return to the Front Panel window and ensure that the addresses listed for the Multimeter and Function Generator match the addresses found in Procedure **P4**. Perform a frequency sweep from 1Hz to 10kHz with 3 steps per decade by filling in the appropriate text boxes and pressing the play button. You can monitor the progress of the frequency sweep by watching it on the strip chart on the Front Panel and by watching the actual voltages on the oscilloscope (you may need to press the "autoscale" button to see the waveforms). **Print the Front Panel**. Reverse the components (capacitor and resistor in series, probes across the resistor) and perform the same frequency sweep. **Print the results.**