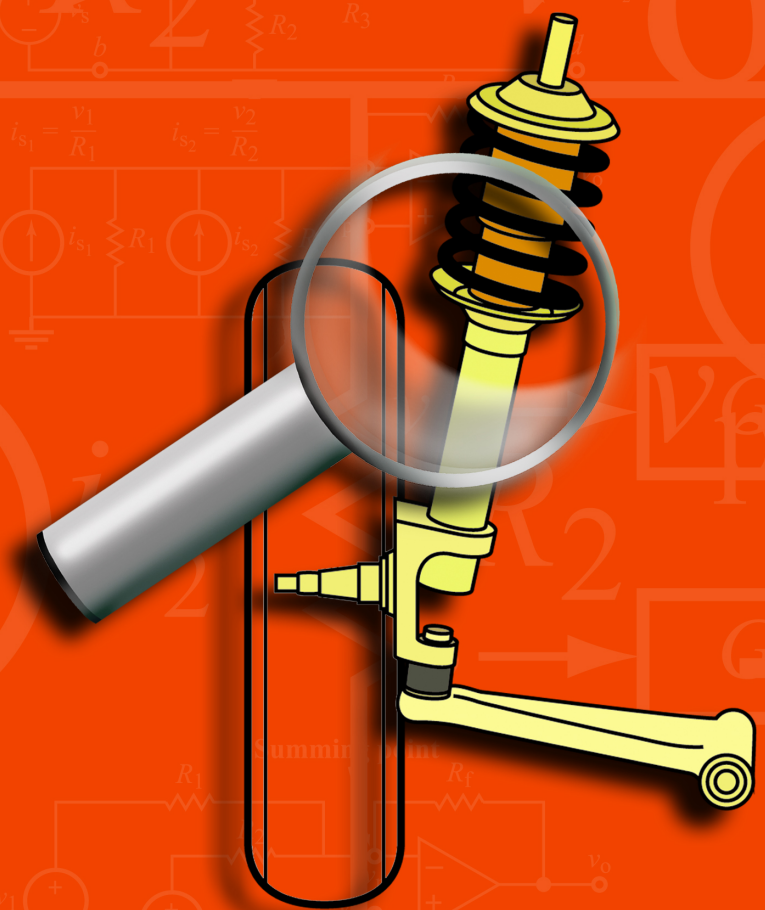
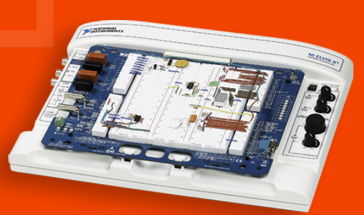
Engineering SIGNALS & SYSTEMS: HANDS-ON LABS with NI ELVIS

Ed Doering





nts PRESS

Contents

Laboratory Project 0

Introduction

This supplement to *Engineering Signals and Systems*[1] by Ulaby and Yagle contains twelve hands-on laboratory projects designed to complement most of the chapters in the textbook. Five of the projects focus on simulation and hardware with NI Multisim and NI ELVIS II, six of the projects revolve around NI LabVIEW, and one project combines all three tools. The lab projects are designed to motivate students with engaging and practical systems including mechanical system modeling, control systems, image filtering and deblurring, AM radio, audio processing, and DTMF decoding. Each lab project guides the students through the activity with detailed instructions and video-based tutorials to demonstrate specific techniques for each of the software tools.

This document is fully hyperlinked for section and figure references, and all video links are live hyperlinks. Opening the PDF version of this document is the most efficient way to access all of the links; clicking a video hyperlink automatically launches the video in a browser. Within the PDF, use `ALT+leftarrow` to navigate back to a starting point.

0.1 Resources

- Appendix A details the parts list required to implement all of the circuits and includes links to component distributors.

- Appendix B describes the Texas Instruments TL072 dual operational amplifier used in several of the circuits.

[1] `http://www.ntspress.com/publications/engineering-signals-and-systems`

- Appendix C lists all of the available video tutorial links.

0.2 Goals for Student Deliverables

Students should document their work in sufficient detail so that it could be replicated by others. Your instructor will provide specific information regarding the lab report format, e.g., lab notebook, memo, or electronic submission. This lab manual uses the symbol "▶" to indicate an instruction and "$\square_N$" to indicate an activity with a reportable result. Number your work with the corresponding activity number N to ensure that your instructor can quickly locate your work for each numbered activity.

Many of the activities require screen shots of the software tools you used to obtain meaningful information. For NI Multisim and NI ELVISmx screen shots, circle parameters that you entered or changed away from default values, and highlight regions where you obtained information. Figure 1 illustrates a screenshot from NI Multisim properly highlighted to indicate control settings that were adjusted away from default values as well as regions on the screen where measurements were obtained.

NOTE: Screen shots in Microsoft Word 2010 can be easily captured and highlighted as follows:

1. Select "Insert" tab and then "Screenshot,"
2. Choose the desired window or select "Screen Clipping" to define an arbitrary region,
3. Select "Shapes," and
4. Place circles or boxes to highlight important values.

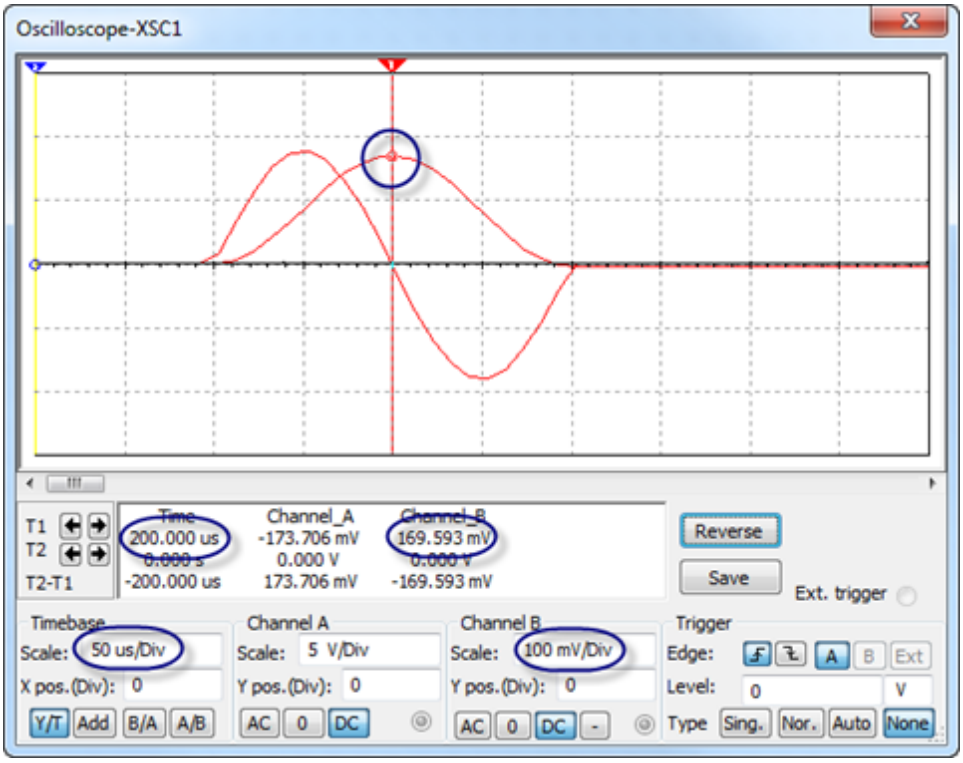


Figure 1: NI Multisim screenshot showing proper markings to indicate control settings adjusted away from default values as well as regions where measurement was obtained.

0.3 Acknowledgements

I gratefully acknowledge contributions from the following individuals:

- Tom Robbins (NTS Press) for his editorial support throughout this project,
- Gretchen Edelmon (National Instruments) for her valuable comments and feedback regarding the design of the projects,
- Mark Walters (National Instruments) for his helpful support of the NI Multisim product,
- Mark Yoder (Rose-Hulman) for his enthusiastic review of selected projects, and
- Rose-Hulman students in Discrete-Time Signals and Systems ECE380 (Winter 2012-13) who offered much helpful feedback on their experience with selected projects.

Ed Doering
Department of Electrical and Computer Engineering
Rose-Hulman Institute of Technology
Terre Haute, IN 47803

doering@rose-hulman.edu

Laboratory Project 1

Graphical Convolution

1.1 Synopsis

Convolution describes the time-domain process by which a linear time-invariant system responds to an input signal. Knowledge of the system's impulse response permits the system output to be computed for *any* input signal shape.

In this lab project you will determine the impulse response of two first-order systems given their step responses, apply graphical convolution to calculate the system output for a non-standard signal shape, and then compare your calculations to simulation and hardware.

Ulaby/Yagle Sections 2-2 to 2-4

1.2 Objectives

1. Determine the impulse response of a system given its step response
2. Calculate the system output to a non-standard pulse shape
3. Simulate the system output
4. Measure the system output in hardware
5. Compare analytical, simulated, and measured results

1.3 Deliverables

1. Hardcopy of all Multisim circuits and annotated screen shots of simulation results used as the basis of a measurement
2. Hardcopy of all ELVISmx instrumentation screen shots annotated to show measurement procedures
3. Lab report or notebook formatted according to instructor's requirements

1.4 Required Resources

1. NI Multisim
2. NI ELVIS II
3. Resistor: $R = 10.0$ kΩ
4. Capacitor: $C = 0.1$ μF
5. Jumper wire kit

1.5 Preparation

Impulse response from step response

Figure 1.1 on the facing page shows two systems implemented as first-order *RC* circuits. The step response of System A is

$$y_{\text{step}_A}(t) = \left(1 - e^{-t/RC}\right) u(t), \tag{1.1}$$

and the step response of System B is

$$y_{\text{step}_B}(t) = e^{-t/RC} u(t). \tag{1.2}$$

As discussed in Section 2-2.4 of your text, the impulse response $h(t)$ of the system is the time derivative of the step response $y_{\text{step}}(t)$:

$$h(t) = \frac{dy_{\text{step}}}{dt} \tag{1.3}$$

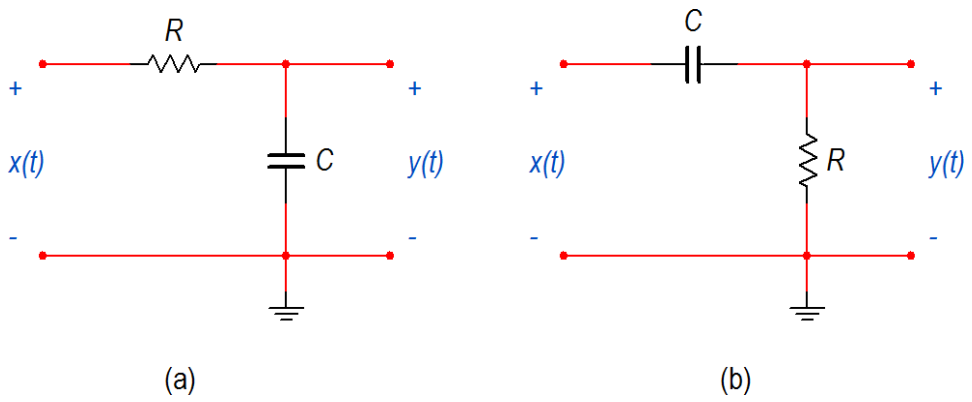


Figure 1.1: Two systems implemented as first-order *RC* circuits: (a) System A with grounded capacitor, and (b) System B with grounded resistor.

$\square_1$ Determine the impulse response waveforms $h_{\mathrm{A}}(t)$ and $h_{\mathrm{B}}(t)$ for the two systems A and B. HINT: $h_{\mathrm{B}}(t)$ is the sum of *two* terms, one of them due to the step discontinuity at time $t = 0$. Sketch the impulse response equations for $R = 1\ \Omega$ and $C = 1$ F.

Graphical convolution

Equation 2.30 in your text and replicated below describes how to convolve an input signal $x(t)$ with the system impulse response $h(t)$ to form the output $y(t)$:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\,d\tau \tag{1.4}$$

Graphical convolution provides a visual method to set up the integral for piecewise-continuous functions.

▶ Review Section 2-3 in your text, and then follow along with this video tutorial `http://youtu.be/zoRJZDiPGds` to learn how to apply the graphical convolution procedure.

$\square_2$ Evaluate the impulse response $h_{\mathrm{A}}(t)$ for $R = 1\ \Omega$ and $C = 1$ F.

$\square_3$ Apply graphical convolution to determine the output $y_{\mathrm{A}}(t)$ of System A caused by the system input signal $x(t)$ plotted in Figure 1.2; set the amplitude A to 5 volts.

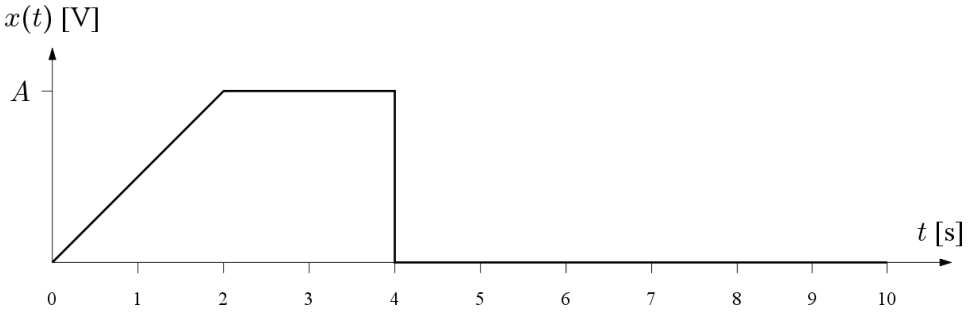


Figure 1.2: Input signal $x(t)$ with amplitude A.

$\square_4$ Use a computer plotting tool to visualize $y_{\mathrm{A}}(t)$ as a graph over the time range 0 to 10 seconds.

$\square_5$ Evaluate $y_{\mathrm{A}}(t)$ at the times 2, 4, and 6 seconds.

$\square_6$ Evaluate the impulse response $h_{\mathrm{B}}(t)$ for $R = 1\ \Omega$ and $C = 1$ F. Determine $y_{\mathrm{B}}(t)$ by recycling as much as possible your results for $y_{\mathrm{A}}(t)$. HINT: Your result for $h_{\mathrm{B}}(t)$ should include an impulse function $\delta(t)$; determine the system output for this portion of $h_{\mathrm{B}}(t)$ and then apply superposition to determine the complete system output $y_{\mathrm{B}}(t)$.

$\square_7$ Use a computer plotting tool to visualize $y_{\mathrm{B}}(t)$ as a graph over the time range 0 to 10 seconds.

$\square_8$ Evaluate $y_{\mathrm{B}}(t)$ at the times 2, 3.99, 4.01, and 6 seconds.

Video tutorials

Study the following video tutorials that relate to this project:

- NI Multisim:

  - Piecewise linear (PWL) voltage source:
    `http://youtu.be/YYU5WuyebD0`
  - Plot time-domain circuit response with Transient Analysis:
    `http://youtu.be/waKnad_EXkc`

- NI ELVISmx:
  - Arbitrary Waveform Generator (ARB):
    `http://decibel.ni.com/content/docs/DOC-12941`
  - Oscilloscope:
    `http://decibel.ni.com/content/docs/DOC-12942`

1.6 Simulation

For this and the subsequent hardware measurements, set $R = 10$ kΩ and $C = 0.1$ μF, and change the time scale unit of the plot in Figure 1.2 on the facing page to milliseconds, i.e., the time span of the signal is 10 ms. These values permit direct comparison to your analytical results provided that you interpret your earlier plots as functions of time in milliseconds.

$\square_9$ Enter the circuit of System A into NI Multisim. Set up a transient analysis to plot the circuit input and output over the time span 0 to 10 ms with at least 200 sample points. Use the `PIECEWISE_LINEAR_VOLTAGE` source to create $x(t)$.

$\square_{10}$ Take cursor measurements of $y_{\mathrm{A}}(t)$ at times 2, 4, and 6 ms.

$\square_{11}$ Calculate the percent difference between your simulation and your earlier analytical results from graphical convolution.

$\square_{12}$ Repeat these activities for the circuit of System B, taking cursor measurements at the times 2, 3.99, 4.01, and 6 seconds.

1.7 Hardware Measurements

▶ Build System A; use the ELVISmx Arbitrary Waveform Generator to create $x(t)$ on the NI ELVIS II analog output A0 with sampling rate set to 100 kS/s. Use the ELVISmx Oscilloscope to monitor the input signal $x(t)$

on the analog input AI6+ and the output signal $y(t)$ on AI7+; remember to ground AI6- and AI7-.

► IMPORTANT: Once you are satisfied that the input and output signals look correct on the oscilloscope, *disable* the input signal channel and display only the output signal to ensure the best possible accuracy.

$\square_{13}$ Adjust the oscilloscope settings (especially the trigger settings) to match as much as possible your simulation and analysis plots.

$\square_{14}$ Take cursor measurements of $y(t)$ at times 2, 4, and 6 ms.

$\square_{15}$ Calculate the percent difference between your measurement results and your earlier analytical results from graphical convolution.

$\square_{16}$ Repeat these activities for System B, taking cursor measurements at the times 2, 3.99, 4.01, and 6 seconds.

1.8 Discussion

$\square_{17}$ Reflect on your analytical, simulation, and hardware measurement results. How well do these three viewpoints on the same system agree with each other?

$\square_{18}$ How are the systems of System A and System B very similar to each other?

$\square_{19}$ Compare the input and output waveforms for System A. Generally speaking what is the effect of System A? Repeat for System B.

Laboratory Project 2

Step Response of a Second-Order LCCDE

2.1 Synopsis

A second-order linear constant-coefficient differential equation (LCCDE, for short) models a wide range of engineering systems, including mechanical, thermal, chemical, and electrical. Understanding the basic properties of a second-order LCCDE provides a good foundation for study of many types of systems.

In this project you will study the properties of an inductor-capacitor-resistor (*LCR*) circuit with circuit simulation and then compare your results to measurements from the physical circuit.

Ulaby/Yagle Section 2-8

2.2 Objectives

1. Write second-order system parameters in terms of *LCR* circuit component values
2. Set up a parameterized transient analysis to plot step response as a function of resistance
3. Measure damped natural frequency and peak overshoot from underdamped response waveforms
4. Measure rise time

5. Compare simulation and circuit measurements

2.3 Deliverables

1. Hardcopy of all Multisim circuits and annotated screen shots of simulation results used as the basis of a measurement
2. Hardcopy of all ELVISmx instrumentation screen shots annotated to show measurement procedures
3. Lab report or notebook formatted according to instructor's requirements

2.4 Equipment and Software

1. NI Multisim
2. NI ELVIS II
3. Resistors: $R = 1.0$ kΩ (4)
4. Capacitor: $C = 0.1$ μF
5. Inductors: 33 mH (3)
6. Jumper wire kit

2.5 Preparation

Second-order LCCDE

The step response of a second-order linear constant-coefficient differential equation (abbreviated as "LCCDE") takes on three distinct forms depending on the *damping coefficient* ξ; these forms are called underdamped, critically damped, and overdamped. The second-order LCCDE describes a wide variety of engineering systems, including the *LCR* circuit pictured in Figure 2.1 on the next page.

► Study Section 2-8 in your text.

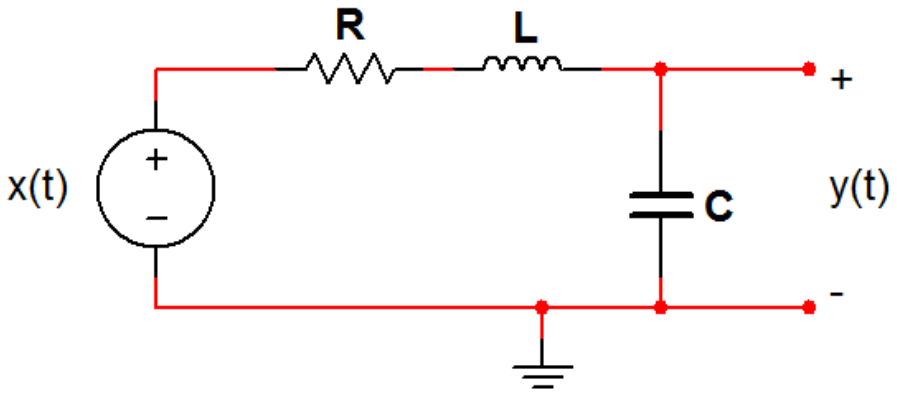


Figure 2.1: Second-order *LCR* circuit.

► Several equations from your text serve as the foundation for this lab project and are reproduced here for convenience:

Equation 2.121, general-purpose second-order linear constant coefficient differential equation (LCCDE):

$$\frac{d^2y}{dt^2} + a_1\frac{dy}{dt} + a_2y(t) = b_1\frac{dx}{dt} + b_2x(t) \tag{2.1}$$

From Table 2-3, equations for attenuation coefficient α, undamped natural frequency ω_0, damped natural frequency ω_d, and damping coefficient ξ:

$$\alpha = \frac{a_1}{2} \tag{2.2}$$

$$\omega_0 = \sqrt{a_2} \tag{2.3}$$

$$\omega_d = \omega_0\sqrt{1-\xi^2} \tag{2.4}$$

$$\xi = \frac{\alpha}{\omega_0} \tag{2.5}$$

► The LCCDE that describes the *LCR* circuit of Figure 2.1 is:

$$\frac{d^2y}{dt^2} + \frac{R}{L}\frac{dy}{dt} + \frac{1}{LC}y(t) = \frac{1}{LC}x(t) \tag{2.6}$$

where $x(t)$ is the circuit's input voltage signal and $y(t)$ is the circuit's output signal.

$\square_1$ Write the generic LCCDE coefficients a_1, a_2, b_1, and b_2 in terms of the circuit component values L, C, and R.

$\square_2$ Write the attenuation coefficient α, the undamped natural frequency ω_0, and the damping coefficient ξ in terms of the circuit component values, as well.

$\square_3$ Calculate α, ω_0, and ξ for $L = 99$ mH, $C = 0.1$ μF, and four values of R: 500 Ω, 1 kΩ, 2 kΩ, and 4 kΩ. Identify the resistor value (or values) associated with underdamped ($\xi < 1$), critically-damped ($\xi = 1$), and overdamped step response ($\xi > 1$).

$\square_4$ A "critically-damped" response forms the theoretical boundary between underdamped and overdamped responses. In practice a system can be close to this theoretical boundary but will always be either underdamped or overdamped. Identify which of the four resistance values above is closest to a critically-damped response.

$\square_5$ The damped natural frequency ω_d serves as a distinct and easily-measured feature of the oscillatory nature of an underdamped response, and offers a point of comparison between theory, simulation, and measurement. Note that the similarly-named *undamped* natural frequency ω_0 cannot be directly measured. Calculate the damped natural frequency ω_d for each of the resistance values associated with an underdamped response.

Video tutorials

Study the following video tutorials that relate to this project:

- NI Multisim:
  - Use a Parameter Sweep analysis to plot resistor power as a function of resistance:
    `http://youtu.be/3k2g9Penuag`
  - Plot time-domain circuit response with Transient Analysis:
    `http://youtu.be/waKnad_EXkc`
  - Find the maximum value of a trace in Grapher View:
    `http://youtu.be/MzYK60mfh2Y`
- NI ELVISmx:

- Function Generator (FGEN):
  `http://decibel.ni.com/content/docs/DOC-12940`
- Oscilloscope:
  `http://decibel.ni.com/content/docs/DOC-12942`

2.6 Circuit Simulation, Part 1

► Enter the circuit of Figure 2.1 on page 17 into NI Multisim. Use a `PULSE_VOLTAGE` source to create a 1-volt input step signal with zero initial value. Set the pulse width to 4 ms and the period to 5 ms.

$\square_6$ Set up a "Parameter Sweep" analysis based on transient analysis to plot the step response $y(t)$ for each of the four values of R on the same graph. Set up the transient analysis to run until 2 ms with at least 200 time points. Refer to the video tutorial `http://youtu.be/waKnad_EXkc` regarding transient analysis setup and `http://youtu.be/3k2g9Penuag` for parameter sweep setup; use "List" mode for the "Sweep variation type" and enter a comma-delimited list of the four resistance values. Capture a screen shot of the plot, and discuss your general observations about the nature of the second-order step as the damping coefficient ξ increases.

$\square_7$ Take cursor measurements to determine the damped natural frequency ω_d. Recall that angular frequency ω is 2π divided by the oscillation period T. Compare this value by percent difference to the theoretical value you calculated earlier.

$\square_8$ Take cursor measurements to determine the *peak overshoot* of the underdamped responses, with peak overshoot defined as the maximum excursion of the signal above its final value.

2.7 Hardware Measurements, Part 1

$\square_9$ Show how to connect the three available 33 mH inductors to form a 99 mH inductor.

$\square_{10}$ Show how to connect one or more of the four available 1 kΩ resistors to create each of the four required resistances 500 Ω, 1 kΩ, 2 kΩ, and 4 kΩ. Hint: Recall what you know about series and parallel equivalent resistance.

► Build the circuit of Figure 2.1 on page 17:

- Begin with $R = 500\ \Omega$.
- Create the circuit input $x(t)$ with NI ELVIS II FGEN and monitor this signal with AI7+. Connect AI7- to AIGND.
- Connect the circuit output $y(t)$ to AI6+. Connect AI6- to AIGND.
- Set up the ELVISmx Function Generator to drive the circuit input with a 1V step at 200 Hz. Adjust the DC offset so that the input begins at -1 V and reaches 0 V.
- Set up the ELVISmx Oscilloscope to observe the step input $x(t)$ on Channel 0 and the step response $y(t)$ on Channel 1.

► Set the oscilloscope "Timebase" control to 200 μs per division. Adjust the "Trigger" controls to left-justify the display on the rising edge of the input $x(t)$. Adjust the "Scale" controls to stretch the traces to fill as much of the display as possible without clipping. Use the "Run Once" mode and repeatedly press "Run" until you get an acceptable set of traces.

$\square_{11}$ Take cursor measurements to determine the damped natural frequency ω_d; refer back to the simulation section for the measurement procedure. Capture a screen shot of the oscilloscope to show your measurement.

$\square_{12}$ Record the "Vp-p" (peak-to-peak voltage) value for Channel 1 displayed below the waveform traces. The oscilloscope reports the difference between the observed minimum and maximum values of the trace, therefore subtract one from this value to obtain the peak overshoot value.

$\square_{13}$ Compare (by percent difference) your measured values for ω_d and peak overshoot with those you obtained by simulation. State whether or not you believe the percent differences to be reasonable.

$\square_{14}$ Physical inductors contain a significant effective resistance. Use the ELVISmx DMM (digital multimeter) set up as an ohmmeter to measure the resistance of your 99 mH inductor; remember to disconnect the resistor from the *LCR* circuit before you measure resistance.

2.8 Circuit Simulation, Part 2

► As you have just learned, the physical inductor is not well-modeled by the ideal inductor in your earlier simulation. To remedy this problem, add a series resistor into the circuit next to the inductor and set its value to the DMM ohmmeter measurement from the previous step.

$\square_{15}$ Re-run the parameter sweep to create a new set of step response signals. Capture a screen shot of this updated graph.

$\square_{16}$ Take cursor measurements as before to determine the new damped natural frequency ω_d and the peak overshoot. Compare these values by percent difference to those you obtained from the physical circuit. Discuss the performance of the improved simulation model of the physical circuit.

2.9 Hardware Measurements, Part 2

$\square_{17}$ The *rise time* of the step response is another important descriptor of the response waveform. Rise time measures the length of time for the system to reach 90% of its final steady-state value after the initial step. Take cursor measurements to determine the rise time of each of the four step response waveforms; set the trigger level to -0.1 V to create a convenient visual marker for the 90% signal crossing. Tabulate your results with columns for R, ξ, and rise time. Capture one screen shot for each measurement.

Laboratory Project 3

Car Suspension Analog Model

3.1 Synopsis

A wide variety of engineering systems – thermal, fluid, mechanical, chemical, electrical – share *identical* second-order linear constant coefficient differential equation forms. The constants have different dimensions as required by the different system types, but the underlying system dynamics such as damping and oscillation are the same.

In this project you will implement and study an electrical analog to an automobile suspension system.

Ulaby/Yagle Section 4-3

3.2 Objectives

1. Choose component values for an electrical analog to a mechanical system
2. Determine velocity profile from physical geometry
3. Simulate the system
4. Compare simulation and measurements
5. Use an integrator and differentiator to obtain displacement and acceleration from velocity

3.3 Deliverables

1. Hardcopy of all Multisim circuits and annotated screen shots of simulation results used as the basis of a measurement
2. Hardcopy of all ELVISmx instrumentation screen shots annotated to show measurement procedures
3. Lab report or notebook formatted according to instructor's requirements

3.4 Equipment and Software

1. NI Multisim
2. NI ELVIS II
3. Resistors: $R = 1.0$ kΩ (2)
4. Capacitors: $C = 0.1$ μF (4)
5. Inductors: 33 mH (3)
6. Jumper wire kit

3.5 Preparation

Electrical system analog

A second-order *LCR* circuit emulates the second-order system dynamics of the car suspension. In principle it is possible to choose very large inductor and capacitor values and very small resistor values to exactly match the behavior of the mechanical system. Unfortunately these extreme values are not practical. A *time scaling* technique permits practical component values to be selected for the electrical circuit, in effect "shrinking" the car suspension system geometry and operating it on a smaller time scale. For example, the time scale factor $S = 1000$ means that one second of mechanical system time is equivalent to one millisecond of electrical system time. In this section you will apply the time-scaling technique to derive equations for the *LCR* circuit values written in terms of the mechanical system coefficients in such a way that the system dynamics are identical but simply operate on a faster time scale.

► Study Section 4-3 in your text.

► Several equations from your text serve as the foundation for this lab project and are reproduced here for convenience:

Equation 2.121, general-purpose second-order linear constant coefficient differential equation (LCCDE):

$$\frac{d^2y}{dt^2} + a_1\frac{dy}{dt} + a_2y(t) = b_1\frac{dx}{dt} + b_2x(t) \tag{3.1}$$

From Table 2-3, equations for attenuation coefficient α, undamped natural frequency ω_0, and damping coefficient ξ:

$$\alpha = \frac{a_1}{2} \tag{3.2}$$

$$\omega_0 = \sqrt{a_2} \tag{3.3}$$

$$\xi = \frac{\alpha}{\omega_0} \tag{3.4}$$

Equation 4.54, LCCDE that describes the mechanical automobile suspension system:

$$m\frac{d^2v_y}{dt^2} + b\frac{dv_y}{dt} + kv_y(t) = b\frac{dv_x}{dt} + kv_x(t) \tag{3.5}$$

Equation 4.58, LCCDE that describes the electrical analog circuit for the suspension system:

$$C\frac{d^2v_y}{dt^2} + \frac{1}{R}\frac{dv_y}{dt} + \frac{1}{L}v_y(t) = \frac{1}{R}\frac{dv_x}{dt} + \frac{1}{L}v_x(t) \tag{3.6}$$

► The time-scaled version of the electrical system LCCDE is

$$C\frac{d^2v_y}{dt^2} + \frac{1}{RS}\frac{dv_y}{dt} + \frac{1}{LS^2}v_y(t) = \frac{1}{RS}\frac{dv_x}{dt} + \frac{1}{LS^2}v_x(t) \tag{3.7}$$

$\square_1$ Write the attenuation coefficient α for the mechanical system and for the time-scaled electrical system and equate these two forms of α.

$\square_2$ Repeat the previous step for ω_0.

$\square_3$ Increasing the time scaling factor S permits practical component values to be selected for the electrical circuit. To convince yourself that the system dynamics remain the same, show that the electrical system damping coefficient ξ written in terms of your expressions for α and ω_0 remains constant for an arbitrary choice of time scaling factor S.

$\square_4$ Write equations to solve for the capacitance C and the resistance R given a specific choice of inductance L, time scale factor S, mechanical system coefficients for mass m, spring constant k, and damper coefficient b.

► The mechanical system coefficients for this project are the same as for Example 2-17 in your text:

- m = 250 kg,
- k = 10^5 N/m, and
- b = 5000, 10000, and 20000 Ns/m.

► The time scaling factor for this project is S = 1000.

$\square_5$ Select the inductance L as 100 mH, and calculate the required capacitance and three resistance values associated with the three choices of damper coefficient b. Also calculate the damping coefficient ξ for each value of b.

$\square_6$ Show how to arrange three 33 mH, four 0.1 μF capacitors, and two 1000 ohm resistors to implement your required values of L, C, and R. Hint: make use of series and parallel combinations.

Speed bump velocity profile

Figure 3.1 on the next page shows the geometry for an idealized speed bump with height h, width w, and total pavement length L. The rise on the left side is $L/10$ in length as is the fall on the right side. The length of the speed bump plateau is $L/5$.

$\square_7$ Assume that the car's wheel travels with constant velocity v_C over the speed bump and remains continually in contact with the pavement. Draw the displacement profile $x(t)$ for time 0 to L/v_C.

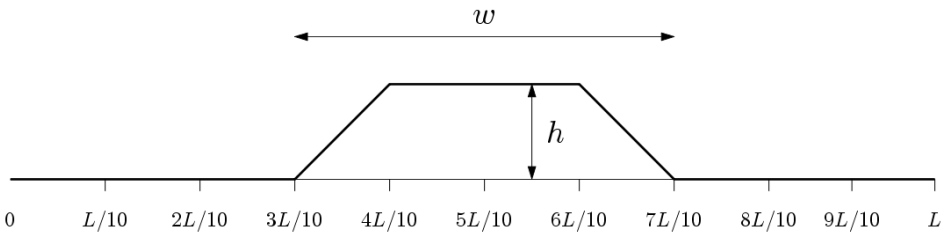


Figure 3.1: Speed bump geometry.

$\square_8$ Draw the velocity profile $v_{\mathrm{x}}(t)$ for the same time range.

$\square_9$ Set the speed bump height to $h = 0.1$ m and the total pavement length to $L = 1$ m, and then calculate the specific velocity values for $v_{\mathrm{C}} = 0.5$ m/s (1.1 mph) and 1.0 m/s (2.2 mph).

► NOTES:

- The electrical analog circuit works in terms of *velocity*, not displacement. Consequently the velocity profiles you just determined serve as the *voltage* inputs to the circuit.
- With the time scaling factor of $S = 1000$, one volt applied to the electrical system analog is equivalent to one meter per second for the original mechanical system.

Video tutorials

Study the following video tutorials that relate to this project:

- NI Multisim:
  - Find components by name:
    `http://youtu.be/5wlFweh4n-c`
  - Plot time-domain circuit response with Transient Analysis:
    `http://youtu.be/waKnad_EXkc`
- NI ELVISmx:

– Arbitrary Waveform Generator (ARB):
`http://decibel.ni.com/content/docs/DOC-12941`

– Oscilloscope:
`http://decibel.ni.com/content/docs/DOC-12942`

3.6 Design evaluation: simulation

► Enter the circuit of Figure 3.2 into NI Multisim. Use a `PIECEWISE_LINEAR_VOLTAGE` source to create the input pavement velocity profile for the 0.5 m/s speed bump.

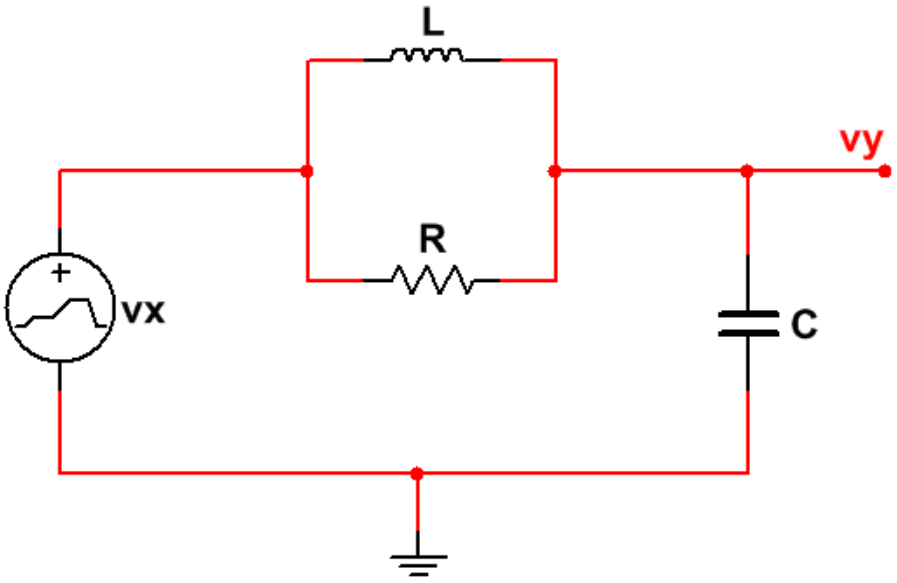


Figure 3.2: Car suspension circuit.

► Add a `VOLTAGE_DIFFERENTIATOR` with a gain of 0.01 millivolts per volt to compute the acceleration $a_y(t)$ from the circuit output velocity $v_y(t)$. Refer to the short video tutorial `http://youtu.be/5wlFweh4n-c` to learn how to look up a component by name. Connect the bottom two terminals to ground.

► Add a VOLTAGE_INTEGRATOR with a gain of 5000 volts per volt to compute the displacement $y(t)$ from the circuit output velocity $v_{\mathrm{y}}(t)$. Connect the bottom two terminals to ground.

► Set up a transient analysis to run until 2 ms with at least 500 time points.

$\square_{10}$ Plot four variables: the input velocity $v_{\mathrm{x}}(t)$ and the three outputs (displacement, velocity, and acceleration). Capture a screen shot of all four variables for each of the three values of R, taking care to use the same vertical-axis scale for each plot to facilitate easy comparison. Label the screen shot with the value of R and ξ.

$\square_{11}$ Discuss your findings, noting the effects of damping coefficient ξ on the behavior of each of the three output quantities.

$\square_{12}$ Create a second version of the piecewise-linear voltage source for the 1.0 m/s speed bump, and then capture another set of screen shots for each of the three values of R. Remember to label the screen shots as in the previous step.

$\square_{13}$ Discuss your findings, noting the effects of damping coefficient ξ on the behavior of each of the three output quantities. Also comment on the effect of increased vehicle speed over the speed bump.

3.7 Design evaluation: hardware

► Build the circuit of Figure 3.2 on the preceding page:

- Set up the ELVISmx Arbitrary Waveform Generator to create the 0.5 m/s speed bump velocity profile $v_{\mathrm{x}}(t)$ on the NI ELVIS II analog output AO with sampling rate set to 200 kS/s.
- Use the ELVISmx Oscilloscope to monitor the input velocity profile $v_{\mathrm{x}}(t)$ on the analog input AI6+ and the output velocity profile $v_{\mathrm{y}}(t)$ on AI7+; remember to ground AI6- and AI7-.

► Set the oscilloscope "Timebase" control to 200 μs per division. Adjust the "Trigger" controls to place the velocity profile in the center of the display.

$\square_{14}$ Capture a screen shot for each of the three values of R, labeling them with the values of R and ξ as before.

► Create the 1.0 m/s speed bump velocity for the arbitrary waveform generator. Use the same total length.

$\square_{15}$ Apply the new speed bump velocity profile to your circuit and capture another set of screen shots for the three values of R.

3.8 Discussion

$\square_{16}$ Evaluate the agreement between your physical circuit measurements and the simulation results. Identify one or two numerical measurements such as peak value or oscillation frequency and report the percentage difference between the two.

$\square_{17}$ Recall that force is proportional to acceleration, i.e., the familiar equation $F = ma$. Study your plots for the acceleration profile $a_{\mathrm{y}}(t)$. Recognizing that a seat-belted passenger experiences the same force as the car mass, discuss how your results for acceleration relate to the forces exerted on the passengers as the car passes over the speed bump for various automobile speeds and damping coefficients.

$\square_{18}$ Explain why experimenting with the electrical analog to the mechanical system provides an advantage compared to experimenting with the mechanical system itself.

Laboratory Project 4

Proportional Control System

4.1 Synopsis

Control systems improve the performance of a physical mechanism by using *negative feedback*. In this lab project you will study a simple proportional control system that speeds up the responsiveness of an emulated DC motor.

Ulaby/Yagle Section 4-8

4.2 Objectives

1. Analyze a proportional control system to determine open-loop and closed-loop performance in terms of time constant and steady-state error
2. Design a proportional control system to achieve specified performance improvements
3. Implement the control system with op-amp modules
4. Evaluate the design with simulation
5. Implement and evaluate the design in hardware
6. Compare design calculations, simulation results, and measurement results
7. Study the qualitative performance of the proportional control system

4.3 Deliverables

1. Hardcopy of all Multisim circuits and annotated screen shots of simulation results used as the basis of a measurement
2. Hardcopy of all ELVISmx instrumentation screen shots annotated to show measurement procedures
3. Lab report or notebook formatted according to instructor's requirements

4.4 Equipment and Software

1. NI Multisim
2. NI ELVIS II
3. TI TL072 dual op-amp (2 devices)
4. Resistors: $R = 1.0$ kΩ, $R = 47$ kΩ (2), $R = 100$ kΩ (2); four additional resistors to be calculated
5. Capacitor: $C = 4.7$ μF, electrolytic
6. Jumper wire kit

4.5 Preparation

Proportional controller analysis

Figure 4.1 on the facing page shows the block diagram of a *proportional control system* that contains the following elements and signals:

- Mechanism, $\mathbf{M(s)}$: The physical mechanism that performs useful work but whose properties such as reaction time to a change in its input may be too slow. In this lab project the mechanism is an RC circuit that simulates the behavior of a DC motor.
- Summing junction: Compares the control input $x(t)$ to the controlled output $y(t)$ to produce the error signal $e(t) = x(t) - y(t)$. The control input expresses the desired motor speed, and the error signal indicates how well the actual motor speed matches the desired speed.

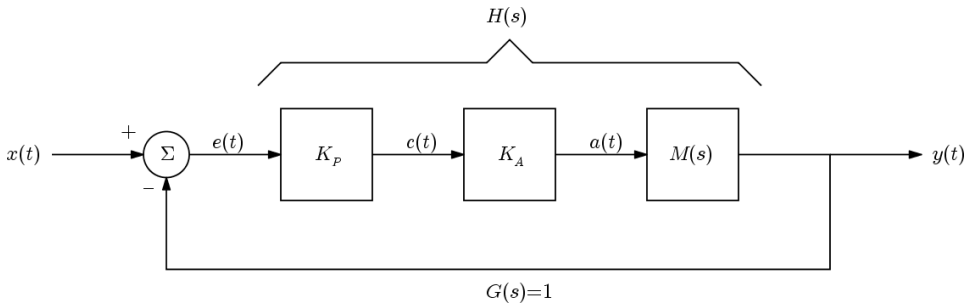


Figure 4.1: Proportional control system.

- Proportional gain, K_{P}: This block amplifies the error signal to produce the *control effort* signal $c(t)$.
- Actuator gain, K_{A}: This block further amplifies the control effort to produce the *actuator effort* applied to the motor input.
- Plant, $\mathbf{H}(\mathbf{s})$: The cascade combination of the two gain stages and the mechanism.
- Feedback, $\mathbf{G}(\mathbf{s})$: The feedback path is taken as unity in this project.

Figure 4.2 shows the circuit emulator of the DC motor mechanism.

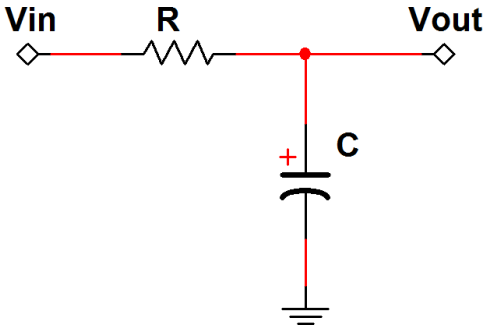


Figure 4.2: DC motor emulated by an RC circuit.

$\square_1$ Derive the open-loop transfer function of the mechanism $\mathbf{M}(\mathbf{s}) = \mathbf{Vout}(\mathbf{s})/\mathbf{Vin}(\mathbf{s})$. NOTE: Write this and all subsequent transfer functions in the standard form of a unity coefficient on the highest power of s in the denominator.

$\square_2$ Determine the open-loop time constant of the mechanism.

$\square_3$ Write the plant $\mathbf{H}(\mathbf{s})$ in the form $\mathbf{H}(\mathbf{s}) = \mathbf{N}(\mathbf{s})/\mathbf{D}(\mathbf{s})$, i.e., distinguish the numerator and denominator polynomials.

$\square_4$ Write the closed-loop transfer function of the entire control system $\mathbf{Q}(\mathbf{s})$; refer to Equation 4.103 in your text.

$\square_5$ Determine the closed-loop time constant of the entire control system.

Proportional controllers rely on a non-zero error output to establish a non-zero control effort. After all, if the motor speed *exactly* matches the desired speed, the control effort goes to zero as does the actuator effort, causing the motor to stop. Consequently, the steady-state controlled output is always slightly lower than the desired control input.

$\square_6$ Derive an equation for the steady-state error due to a unit step function at the control input $x(t)$. Hint: Recall the Laplace transform of the unit step function $u(t)$, and use the final-value theorem.

Proportional controller circuit design

Figure 4.3 on page 39 shows a circuit implementation of the proportional control system using op-amp circuits from Table 4-3 of your text.

Inverting amplifiers (Table 4-3c) serve as the gain blocks K_P and K_A and the inverting summer (Table 4-3d) implements the summing junction. The "sign change" amplifier cancels the sign of one summer input to produce the difference (error signal) $e(t)$. The RC circuit that emulates the DC motor is duplicated as a convenience in the simulator for direct comparison between open-loop and closed-loop response to the step voltage source.

$\square_7$ Set the mechanism elements to $R = 1.0$ kΩ and $C = 4.7$ μF and calculate the open-loop time constant.

$\square_8$ Choose K_P and K_A to make the closed-loop time constant between 10 to 12 times faster than the open-loop time constant. Balance the gains between the two inverting amplifier stages.

$\square_9$ Choose resistor values R_{KPf}, R_{KPi}, R_{KAf}, and R_{KAi} to implement the gains you calculated in the previous step. Keep all resistor values in the 1K to 100K range.

$\square_{10}$ Based on your four resistor values, calculate the closed-loop time constant and the steady-state error expressed as a percentage.

Video tutorials

Study the following video tutorials that relate to this project:

- NI Multisim:
  - Find components by name:
    `http://youtu.be/5wlFweh4n-c`
  - Place dual op amp second device:
    `http://youtu.be/-QDFEf-KdEw`
  - Plot time-domain circuit response with Transient Analysis:
    `http://youtu.be/waKnad_EXkc`
  - Set cursor to a specific value:
    `http://youtu.be/48sQja58I10`
- NI ELVISmx:
  - Function Generator (FGEN):
    `http://decibel.ni.com/content/docs/DOC-12940`
  - Oscilloscope:
    `http://decibel.ni.com/content/docs/DOC-12942`

4.6 Design evaluation: simulation

► Enter the circuit of Figure 4.3 on page 39 into NI Multisim. Use the `TL072CP` dual op-amp model and the `PULSE_VOLTAGE` source to apply a unit step control input. Place "on-page connectors" to name the signal nets of interest.

► Set up a transient analysis to run until 10 ms with at least 200 time points. Select the labeled signal nets `X`, `Yol`, and `Ycl` for display.

$\square_{11}$ Take cursor measurements to determine the open-loop and closed-loop time constants as well as the closed-loop steady-state error. Measure the time constant as the time it takes for the output to reach 63% of its final value after the step is applied.

$\square_{12}$ Report the percentage difference between your simulation results and your design results. Comment on the level of agreement between these two sets of results. Refine your design as necessary to comply with the design specifications of a 10 to 12 times speedup of the time constant.

$\square_{13}$ Modify the transient analysis to add plots of the error signal `E`, negated control effort `C` (add an expression "`-C`" to cancel the effect of the inverting amplifier sign), and the actuator effort `A`.

$\square_{14}$ Study the plots of the various internal controller signals and discuss how the proportional controller is able to achieve its speedup of the DC motor.

4.7 Design evaluation: hardware

► Build the circuit of Figure 4.3 on page 39:

- Use only one RC circuit for the DC motor emulator to permit a proper comparison of open-loop vs. closed-loop performance. Remember to observe the polarity of the electrolytic capacitor.
- Use the TI TL072C dual op-amp or equivalent device; refer to Appendix B on page 111 for details. Power the op-amps with the NI ELVIS II $\pm$15-volt dual supply.
- Create the control input $x(t)$ with NI ELVIS II FGEN and monitor this signal with AI7+. Connect AI7- to AIGND.
- Set up the ELVISmx Function Generator to drive the input with a 1V step at 10Hz. Adjust the DC offset so that the input begins at 0V and reaches 1V.
- Set up the ELVISmx Oscilloscope to observe the control input $x(t)$ on Channel 0.
- Set up the ELVISmx Oscilloscope to observe other signals within the controller by displaying AI6+ on Channel 1. Connect AI6- to AIGND.

► Temporarily disconnect the input of the DC motor emulator from the controller and drive it directly from the function generator. Adjust the oscilloscope timebase and vertical scale to fill the screen as much as possible with signals $x(t)$ and $y_{\mathrm{OL}}(t)$.

$\square_{15}$ Measure the open-loop time constant using the same cursor technique that you used with the simulator.

$\square_{16}$ Reconnect the DC motor emulator to the rest of the controller. Measure the closed-loop time constant and steady-state error.

$\square_{17}$ Report the percentage difference between your measured and simulated results for time constants and steady-state error. Comment on the level of agreement.

► If necessary, trouble-shoot your design by observing intermediate signals within the controller and comparing those to your simulator results.

Qualitative behavior of the controller

► Set up the oscilloscope for a consistent horizontal and vertical scale for all subsequent measurements:

- Timebase = 10ms/div,
- Vertical Scale = 200mV/div (both channels),
- Vertical position = 0 (both channels),
- Trigger type = edge, slope = rising, source = Channel 0, and level = 0.5

► Maintain the same function generator amplitude and frequency used earlier.

$\square_{18}$ As you did earlier, temporarily connect the DC motor emulator in open-loop mode. Display the open-loop output for square, sine, and triangle waveforms. Discuss the ability of the open-loop system to accurately track the desired input waveform, and specifically comment on what works well and what does not.

$\square_{19}$ Reconnect the DC motor emulator for closed-loop operation. Display the closed-loop output for the same three waveform shapes, and then discuss the ability of the closed-loop system to accurately track the desired input waveform. Specifically comment on what works well and what does not; also compare the performance of the closed-loop system to the open-loop system, and state the improvements that you observe.

$\square_{20}$ Display the actuator effort for the same three waveform shapes. Discuss how the actuator effort relates to the controller input waveform, i.e., how must the actuator operate to improve the performance of the DC motor responsiveness?

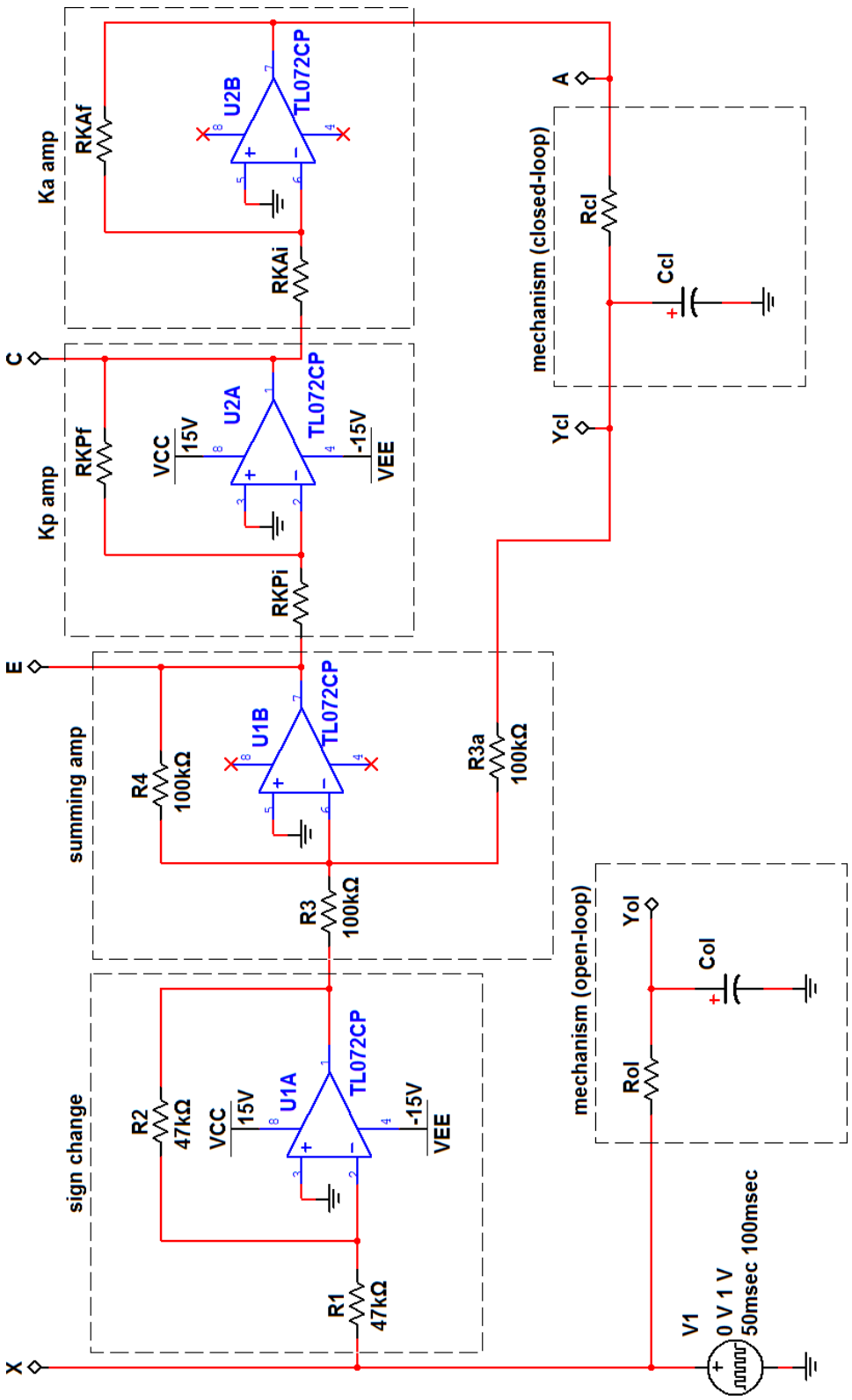


Figure 4.3: Proportional controller at the circuit level.

Laboratory Project 5

Fourier Series Demonstrator

5.1 Synopsis

Any periodic signal $x(t)$ has a Fourier series representation in which suitably scaled and phase shifted sinusoids of increasing frequency sum together to form $x(t)$. Adding together a finite number of sinusoids creates an approximation to $x(t)$.

In this project you will create a Fourier series demonstrator in LabVIEW. The interactive demonstrator will allow you to experience a variety of effects due to a finite number of sinusoids as well as the effects of phase distortion and the consequence of ignoring phase entirely. Your application will also create a waveform data file for the ELVISmx Arbitrary Waveform Generator and get you started learning how to use the ELVISmx Dynamic Signal Analyzer to measure amplitude spectra.

Ulaby/Yagle Section 5-4

5.2 Objectives

1. Implement an interactive Fourier series demonstrator application in LabVIEW
2. Study signals resulting from a truncated Fourier series
3. Study Gibbs phenomenon
4. Study the effects of phase distortion and complete absence of phase information

5. Use the ELVISmx Dynamic Signal Analyzer (DSA) to measure amplitude spectra

5.3 Deliverables

1. Hardcopy of all LabVIEW block diagrams and front panels that you create
2. Screen shots of requested spectrum/time plots
3. Lab report or notebook formatted according to instructor's requirements

5.4 Required Resources

1. NI ELVIS II
2. NI LabVIEW with MathScript RT Module

5.5 Preparation

Fourier series mathematics

► Study Section 5-4 in your text.

$\square_1$ Study the Fourier series descriptions of Waveforms 1, 2, 4, 6, 7, and 8 in Table 5-4 of your text; this list will henceforth be called the "selected waveforms." For each of these waveforms identify the expressions for a_{n}, b_{n}, and a_0, the dc (average) value of the waveform.

$\square_2$ State how to write the amplitude and phase form of the Fourier series, i.e., how to write c_{n} and ϕ_{n} in terms of the sine/cosine coefficients a_{n} and b_{n}.

Fourier series demonstrator construction tips

Figure 5.1 on page 50 shows the front panel of the Fourier series demonstrator LabVIEW application you will create in this project. The application

constructs selected waveforms from Table 5-4 by adding N terms of the infinite series. The application interactively displays the waveform $x_{\mathrm{t}}(t)$ and its associated amplitude/phase spectrum as the N slider control varies.

► Read through these construction tips for the front panel and then refer back to this step as needed while you construct the application; each tip refers to the letter indicator on Figure 5.1 on page 50:

A Menu control with enumerated data type; see video tutorial `http://youtu.be/tImN4C03W1k`

B File Path Control: Right-click on the folder icon, choose "Browse Options," and choose "New or existing."

C Button control: Right-click on the button, choose "Mechanical Action," and select "Switch Until Released."

D Stop button: Right-click on the button, select "Properties," select the "Key Navigation" tab, and set "Toggle" to "Escape." This way you can stop the application by pressing the "ESC" key.

E Plot legend: Right-click on the waveform graph, select "Visible Items," uncheck "Plot Legend," and check "Graph Palette."

F Horizontal Pointer Slide: Right-click, select "Visible Items," and check "Digital Display." Again right-click, select "Representation," and choose "I32" to set N as an integer. Double-click the upper limit of the slider's numerical scale to change the value to 200.

G Plot style: After you place the waveform graph indicator, click the center of the plot legend, select "Common Plots," and choose the discrete lines icon in the center of the lower row. Afterwards hide the plot legend.

H Numerical indicator display precision: After you create the array indicator, select an individual numerical display, right-click and choose "Display Format," and then select "Floating point," 4 digits, and set "Precision Type" to "Digits of precision."

► Save front panel control default values by choosing "Edit" then "Make Current Values Default" to save effort when you close and re-open the VI.

► Turn off auto-scaling on the y-axis of the waveform graph indicators to maintain a fair comparison between one waveform to another.

Figure 5.2 on page 51 shows the block diagram of the Fourier series demonstrator LabVIEW application you will create in this project. A MathScript node contains all of the Fourier series calculations. The event-structure and while-loop combination executes the MathScript node only when a value changes on a front panel control; refer to the tutorial video `http://youtu.be/eGlvOiqYVxg` for more details on event structures. The block diagram also includes a feature to write the text file representation of $x(t)$ that can be loaded directly into the ELVISmx Arbitrary Waveform Generator.

► Read through these construction tips for the block diagram and then refer back to this step as needed while you construct the application; each tip refers to the letter indicator on Figure 5.2 on page 51:

A Execute the event structure one time when the application first runs; see the tutorial video `http://youtu.be/UFEvmmggqLQ` for details.

B Make the event structure respond to all front panel controls; do *not* delete the "Timeout" event.

C MathScript tips:

- Access MathScript help and also experiment with syntax by using the interactive MathScript Window available on the "Tools" menu.
- Right-click on the MathScript node frame and choose "Add Input" and "Add Output" to create connection terminals. Note that a variable must already exist in the MathScript text before "Add Output" will display any variables.
- Set up the time axis as an array: `t=0:dt:P*T0-dt` where "dT" is the waveform period T_0 divided by the number of samples per period N_S and "P" is the number of periods to display.
- Set up "n" as an array `n=1:N`.
- Create a new array y initialized to zero values and of the same length as an existing array x by writing `y=x-x`, i.e., subtract the existing array from itself.
- Use the element-by-element (array style) math operations `.*` (multiplication), `./` (division), and `.^` (exponentiation).

- Use the `switch` construct to choose which of the selected waveforms to create; type `help switch` in the MathScript Window for details; the tutorial video `http://youtu.be/tImN4C03W1k` also shows how to set up a `switch` statement. Create the a_0, a_n, and b_n coefficients by directly translating the Fourier series coefficient equations of Table 5-4 in your text. MathScript provides the constant π as `pi`.
- Waveforms 2 and 4 require zero coefficients for n even. Multiply the coefficient expression by $|\sin(n\pi/2))|$ to zero out the even coefficients. For example, the coefficient calculation for Waveform 2 is `bn=((4*A)./(n*pi)).*abs(sin(n*pi/2))`.
- Form the amplitude coefficients as `cn=real(sqrt(an.^2 + bn.^2))`; the "real()" function ensures that the MathScript node does not automatically create a complex-valued array.
- Form the phase coefficients as `pn = -atan2(bn,an)*(1-pd).*(cn>1e-10)`. The function "atan2()" properly returns an angle in any of the four quadrants; "atan()" can only return an angle in quadrants 1 and 4. The "pd" (phase distortion) value allows a variable amount of phase distortion to be introduced. Multiplying by the inequality test sets all phase values to zero for sufficiently small amplitude coefficients.
- Form the displayed amplitude coefficient array as `cndisp = [a0 cn]` to ensure that the front panel waveform graph indicator x-axis scale appears correctly. Note that MathScript arrays are *1-indexed* meaning that the first array element is index 1. This differs from LabVIEW arrays that are *0-indexed*.
- Form the displayed phase coefficient array in a similar way as `cndisp`, using zero as the first element. Convert the phase from radians to degrees by multiplying by `180/pi`.
- Use a `for` loop to implement the sum for $x(t)$; type `help for` in the MathScript Window for details. Remember to initialize the waveform array `x` to the same length as the time array `t` before you start the loop.

D "Write to Spreadsheet File" built-in subVI: Right-click on the "delimiter" terminal and choose "Create Constant." Right-click on the newly-

created constant, enable `'\' Codes Display` and enter `\n` (newline) as the delimiter character.

E Bundle the waveform array into a waveform data type to ensure that the x-axis scale appears correctly for $x(t)$.

5.6 Fourier Series Demonstrator Implementation

In this section you will create the Fourier series demonstrator LabVIEW application VI.

► Create the front panel and block diagram as pictured in Figures 5.1 on page 50 and 5.2 on page 51.

► Enter the MathScript code to calculate the Fourier series coefficient and waveform arrays for the selected waveforms from Table 5-4 from your text. You may wish to debug your code using the interactive MathScript Window.

► Test each of the front panel controls to ensure that your demonstrator application work correctly. You should find that all of the selected waveforms look like sinusoids at $N = 1$ and then progressively become more distinct as you increase the front panel slide control for N.

5.7 Fourier Series Study

Experiment with your Fourier series demonstrator application to gain experience with the following concepts. Set the front panel control values to match those pictured in Figure 5.1 on page 50.

Gibbs phenomenon

Gibbs phenomenon is the characteristic "ringing" or oscillation that appears at waveform discontinuities.

$\square_3$ Which of the selected waveform numbers shows Gibbs phenomenon? Collect screen shots of these waveforms and state the value of N for each waveform.

$\square_4$ What is distinctive about the waveforms that do not show Gibbs phenomenon?

Minimum number of components

Study the minimum number of frequency components necessary to make a recognizable waveform.

$\square_5$ For each of the selected waveforms determine the minimum number of frequency components required to make the waveform appear visually indistinguishable from the ideal waveform pictured in Table 5-4 of your text.

Phase distortion

The sum of sinusoidal components technique to create an arbitrary periodic waveform involves an amplitude spectrum as well as a phase spectrum. Explore the effect of phase distortion up to complete absence of phase information.

$\square_6$ For each of the selected waveforms investigate phase distortion values from 0 (no phase distortion) up to 1 (complete distortion, all phase components set to zero). Collect representative screen shots to show the impact of varying degrees of phase distortion on the waveform.

5.8 ELVISmx Dynamic Signal Analyzer

The ELVISmx Dynamic Signal Analyzer (DSA) provides a real-time spectrum display of a signal applied to the analog NI ELVIS II input.

► Start the DSA and study its panel controls.

The "FFT Settings" control the *Fast Fourier Transform* computation that serves as the heart of the DSA. These critical settings must be carefully selected to obtain correct amplitude spectrum measurements of periodic signals. First learn how the DSA takes a measurement and then experiment with the settings in a moment.

The DSA repetitively captures a snapshot of the input signal with duration "Resolution (lines)" (R) divided by "Frequency Span" (f_{span}); this time-domain record appears below the frequency display.

When measuring a periodic signal the captured time-domain signal *must* contain an *integer multiple* of periods, consequently R/f_{span} divided by the signal period T_0 must be an integer N and the frequency span may be readily calculated as

$$f_{\text{span}} = \frac{R}{NT_0}, \tag{5.1}$$

where f_{span} is the frequency span in Hz, R is the resolution in "lines" (sample points), T_0 is the period of the periodic input signal in seconds, and N is the number of periods captured. $N = 10$ cycles provides a good starting point for most measurements.

► Set up your Fourier series demonstrator front panel controls to match those pictured in Figure 5.1 on page 50.

► Save this waveform to a text file (click the "Save" button on your front panel).

► Start the ELVISmx Arbitrary Waveform Generator (ARB) and open the waveform editor. You should see a default segment of 10 ms length. Select "File" and open the text file you created; you should see the same waveform you saw in your demonstrator application. Set the sample rate to 200 kHz. Select "File" again, choose "Save As," and select the "Waveform File (.wdt)" option.

► Return to the ARB main panel and open the .wdt file you just created; be sure to enable the channel output. Set the update rate to "200k" and start the generator.

► Use the ELVISmx Oscilloscope to ensure that you are in fact producing a signal on the analog output you selected in the previous step. Stop the oscilloscope when you are finished.

► Return to the ELVISmx Dynamic Signal Analyzer (DSA). Set the resolution to $R = 400$ lines.

$\square_7$ Calculate the required span frequency f_{span} to capture exactly 10 periods of your waveform. Remember to account for the fact that the ARB produces two cycles in 10 ms.

► Use the default DSA settings except for the following:

- Frequency Display:
  1. Units = `Linear`
  2. Mode = `Peak`
- Cursor Settings:
  1. Cursors On = enabled
  2. Cursor Select = `C1`

$\square_8$ Capture a screen shot of the DSA display.

$\square_9$ Measure the amplitude spectrum for odd values of n from 1 to 5 using Cursor 1; take the square root of the displayed cursor value "`dVpk^2`" to obtain the voltage amplitude. **IMPORTANT**: Position Cursor 2 *between* a pair of spectral lines to set its measured value to zero; the value displayed as `dVpk^2` is the *difference* between the two cursors and you want Cursor 2 to serve as the zero reference. Tabulate your measurements and compare using percent difference to the amplitude coefficients calculated by your Fourier series demonstrator application. Additional helpful tips:

- Use the cursor position buttons to make fine adjustments in the vicinity of a spectral line; these are the pair of gray diamonds at the bottom center of the DSA.
- Double-click the upper limit value on the horizontal frequency axis and select a lower value to zoom in on the lower-frequency spectral lines. Do *not* change the "Frequency Span" value for this purpose because this changes the measurement itself.

$\square_{10}$ Discuss the level of agreement you found between the theoretical amplitude spectrum values and those measured by the DSA.

$\square_{11}$ Change the frequency display units to "dB" and collect a screen shot of this result. Discuss the relative advantages and disadvantages of this display mode compared to the linear display mode.

Figure 5.1: Front panel of the Fourier series demonstrator application.

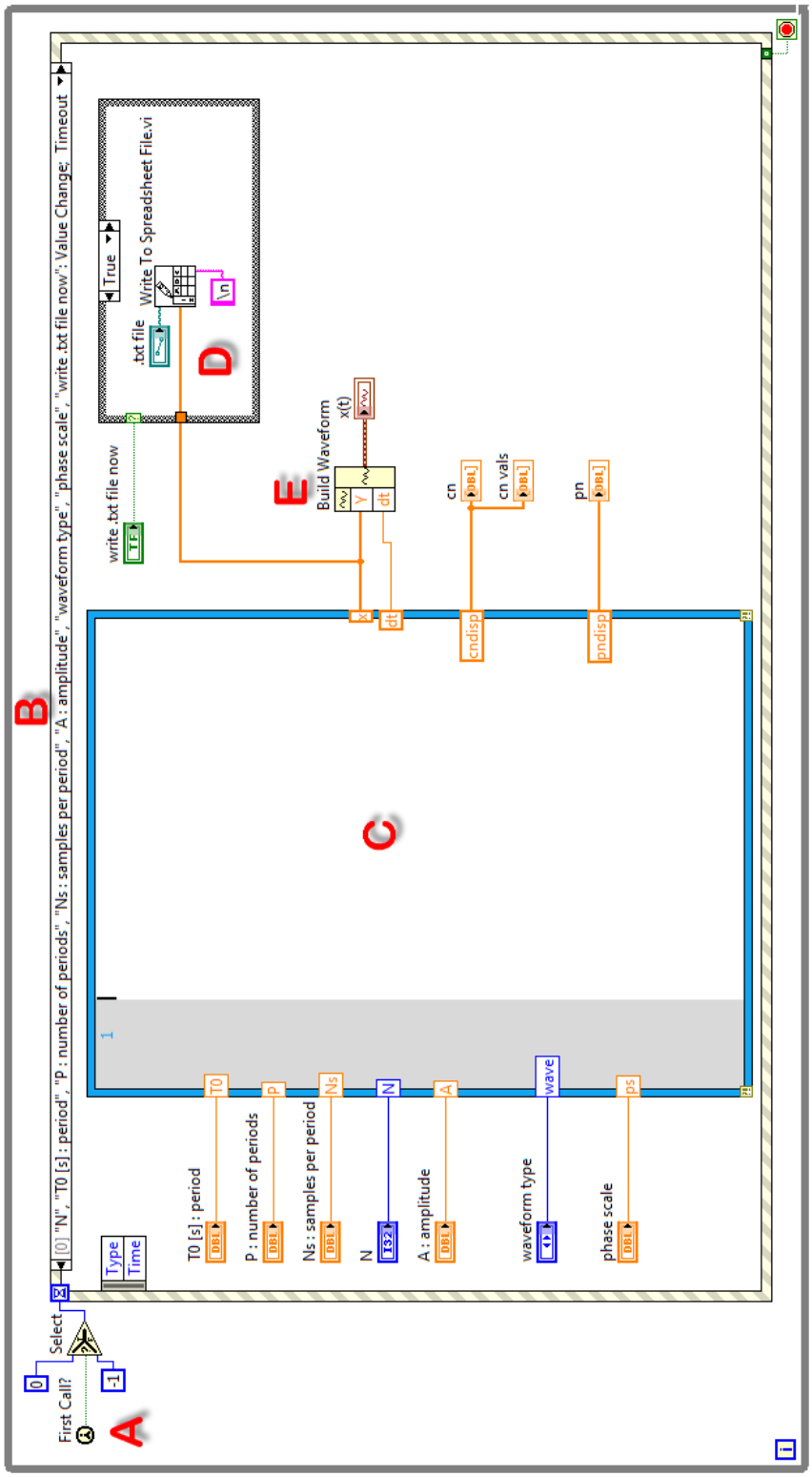


Figure 5.2: Block diagram of the Fourier series demonstrator application.

Laboratory Project 6

Image Filter

6.1 Synopsis

The Fourier transform provides a convenient way to filter signals, and easily extends to multiple dimensions such as two-dimensional image signals.

In this project you will build a 2-D image filter based on the Fast Fourier Transform (FFT), and then study the effect of filter parameters such as cutoff frequency and filter order on the processed image.

Ulaby/Yagle Sections 5-8 and 5-13

6.2 Objectives

1. Use the 2-D Fast Fourier Transform (FFT) to convert from the spatial domain to the frequency domain
2. Devise a 2-D Butterworth image filter from a 1-D magnitude response
3. View image spectra with dynamic range compression
4. Implement a complete image filter in MathScript
5. Study the effects of cutoff frequency, filter order, and center frequency for low-pass, band-pass, high-pass, and band-stop image filters

6.3 Deliverables

1. Hardcopy of all MathScript code that you create

2. Screen shots of requested spectrum/time plots
3. Lab report or notebook formatted according to instructor's requirements

6.4 Required Resources

1. NI LabVIEW with MathScript RT Module
2. Images supplied by your instructor (NOTE: 256×256 pixel 8-bit gray level images work best for this project)

6.5 Preparation

Frequency-domain filtering

► Review Sections 5-8 and 5-13 in your text.

► Study the tutorial video `http://youtu.be/X53SrMnu91A` to learn how the Fourier transform implements two-dimensional image filtering as the product of a frequency-domain image and a two-dimensional filter.

► The one-dimensional Butterworth filter serves as the basis for the four filter styles you will implement in this project (low pass, high pass, band pass, and band stop); study the tutorial video `http://youtu.be/QJdNVdL4iBQ` to learn more.

► A one-dimensional filter easily converts to a two-dimensional filter by changing the frequency variable ω to the radial distance r from zero frequency (DC) in the frequency-domain u-v plane. Study the tutorial video `http://youtu.be/qpwUHt-DEbs` to learn how.

Image processing with MathScript

The LabVIEW MathScript Window and scripting environment will be used for this project.

► Study the tutorial video `http://youtu.be/Tvi1oY0lV2g` to learn basic operations with the MathScript Window.

► Learn everything you need to work with images in MathScript with this tutorial video `http://youtu.be/oM_KSYIDrE4`.

► Type `help` followed by each of these commands and functions that will contribute to your image filter MathScript code; study the help page to ensure that you understand each one:

- File path and directory
  - `pwd`
  - `cd`
  - `dir`
- Image files and image display
  - `figure`
  - `title`
  - `close`; look for the `close all` option
  - `fread_image`
  - `view_image`
  - `size`
- FFT
  - `fft2d`
  - `fftshift`
  - `ifft2d`
- Miscellaneous functions and statements
  - `meshgrid`
  - `real`
  - `sqrt`
  - `log10`
  - `abs`
  - `if`

6.6 Image Filter Implementation

In this section you will gradually create the complete image filter MathScript code.

► Open the MathScript Window and create a new script file.

► Start your script with `clear` and `close all` to clear the variables and close all figure windows.

► Remember to thoroughly document your code with liberal use of '`%`' comment lines.

► Use `cd` to set the current directory to the folder where you have stored the image files.

□$_1$ Use `fread_image` to load an image file into the variable `f`; remember to end the line with a semicolon to suppress echoing of the image data to the MathScript output window. Type `whos` at the command line or select the "Variables" tab to determine the dimensions of the image. Report the name of the image you used and its dimensions; confirm that the image is square, i.e., that it has the same number of pixels in both dimensions.

□$_2$ Use `view_image` to display the image, and then use `title` to label the image. Collect a screen shot of the figure display window.

► Set the variable `N` to the image size using the `size` function. Be sure to call `size` with appropriate options to return a single scalar value rather than an array of values; you may assume that the images you use are always square.

► Convert the image to its frequency-domain form `F` using `fft2d`; remember that MathScript is case sensitive, therefore `f` and `F` are distinct variable names.

$\square_3$ Report the data type of `F` (use `whos` at the command line or look at the "Variables" tab). Also report the maximum absolute value of `F` using `max(max(abs(F)))`; compare this value to the maximum value of the original image `f`.

$\square_4$ Ceate a new figure window using `figure` and then view the image spectrum magnitude `abs(F)` with `view_image`. Collect a screen shot of the image and comment on its appearance.

$\square_5$ As you discovered a moment ago the maximum absolute value of the image spectrum is very large compared to that of the image itself. The image spectrum has a much higher *dynamic range* than the image, and because `view_image` automatically scales the gray level display according to the maximum value most of the interesting features disappear. To remedy this problem, it is common to display the logarithm spectrum (similar to decibels) as $\log_{10}(1 + |F|)$. Capture a screen shot of the log image spectrum and comment on its appearance. **IMPORTANT NOTE:** The log operation is simply for display purposes; be sure to use `F` for subsequent filtering calculations.

$\square_6$ Carefully study your original image spectrum and log image spectrum images. Do you notice anything special about the upper left corner of the image? This is the location of DC (zero frequency). Use `fftshift` to shift the DC location to the center of the image. Note that you can embed functions in other functions, therefore you can write `F=fftshift(fft2d(f))` to take care of the shifting operation as part of the FFT calculation. Capture a screen shot of the shifted log image spectrum and then draw the *u*-*v* plane axes directly on the image. Try to interpret what you see by realizing that the center of the *u*-*v* plane is DC, low frequencies are close to the center, and increasingly-high frequencies occur as you move farther away from DC; compare the image spectrum to the original spatial-domain image.

► Create the *u*-*v* plane index images with `[u,v]=meshgrid2d(-N/2:(N-1)/2)`. You may find it instructive to look at the numerical values of these two arrays for a small value of `N`, say 8. Each value of the `u` array corresponds to the u coordinate in the *u*-*v* plane, and similarly each value of `v` corresponds to the v coordinate. Any given index into the `u` and `v` arrays, say, `u(1)` and `v(1)`, provides the u and v coordinates for that index.

► Create the r (radius) plane as $r = \sqrt{u^2 + v^2}$. Remember to use array-style exponentiation, i.e., `.^2`; without the period, MathScript interprets `u^2` as a matrix product. As in the previous step, look at the result of this step for `N=8`. You should find that zero appears in the center of the array and that values become increasingly large away from the center.

► Create the 2-D Butterworth lowpass filter `Mlp` based on the equation presented in `http://youtu.be/qpwUHt-DEbs` using $r - r_{\text{center}}$ as the frequency variable. The constant offset r_{center} allows the filter passband to be moved away from DC to a new center frequency, thereby converting the lowpass filter into a bandpass filter. Remember to use array-style division and exponentiation.

► Place the Butterworth filter parameters near the beginning of your script for convenience.

► Create the 2-D highpass filter `Mhp` based on your result from the lowpass filter. Use an `if` statement to select between the lowpass and highpass filters to create the filter `M`. View the 2-D Butterworth filter as an image in its own figure window.

► Apply the filter `M` to the image spectrum `F` to create the filtered spectrum `G` (again, remember array-style math operations). Display the filtered log magnitude spectrum its own figure window.

► Transform the filtered image spectrum `G` back to the spatial-domain image `g` with the inverse FFT `ifft2d` – remember to apply `fftshift` *before* you take the inverse FFT to restore the DC location to the upper left corner.

$\square_7$ Check the data type of `g`; you should see that the image is complex-valued. Study the array values for `g` in the "Variable" tab, and then report the typical value of the imaginary component. Explain why either `real` or `abs` could be used to convert the result of the inverse FFT to a real-valued image. Display the filtered image in its own figure window.

► As you are by now aware, the newly-created figure windows overlap existing windows. You can easily cause each figure window to appear in a predefined position:

- Define the display height and width of your monitor with variables `h` and `w`.
- Enter the line `set(gcf,'Position',[0 h/2 w/3 h/2])` immediately after the first `view_image` line in your script. This command places the current figure (`gcf` means "get current figure") with its lower left corner on the far left of the display and halfway up (these are the first two values of the four-element array), sets the figure's horizontal dimension to one third of the display width, and sets its vertical dimension to one half of the display height.
- Place a similar command after each `view_image` to tile your display with all five figure windows.

► Test your finished script with several values each of cutoff frequency r_c, filter order n, and center frequency r_{center} for the lowpass and for the highpass filters. Ensure that your script produces expected results.

► Go back and insert comments as necessary to finalize the documentation on your script. Include your name and other identifying information required by your instructor.

6.7 Image Filtering Study

Experiment with your image filter application to better understand the effect of each of the filter parameters: cutoff frequency r_c, filter order n, and center frequency r_{center}.

Lowpass and bandpass filters

$\square_8$ Set r_{center} = 0 and $n = 3$ for the lowpass filter. Experiment with the cutoff frequency r_c to determine values that make the filtered image appear extremely blurred, moderately blurred, and only slightly blurred. Capture a screen shot of the entire tiled image display for each case and report r_c for each image. Discuss your results.

$\square_9$ Set r_{center} = 0 and Set r_{c} = 20. Experiment with the filter order n. Capture a screen shot of the entire tiled image display for n = 1, 5, and 50. Discuss the overall effect of n on the Butterworth filter image, the filtered spectrum, and the output image.

$\square_{10}$ Set r_{c} = 10 and $n = 3$. Experiment with the center frequency r_{center} to center the passband near low frequencies, mid frequencies, and high frequencies. Capture a screen shot of each and report r_{center} for each image. Discuss the overall effect of r_{center} on the Butterworth filter image, the filtered spectrum, and the output image.

Highpass and bandstop filter

$\square_{11}$ Repeat Activity 8 with the high-pass filter selected.

$\square_{12}$ Repeat Activity 10 with the high-pass filter selected.

Laboratory Project 7

Butterworth Filter

7.1 Synopsis

In this project you will design an active Butterworth lowpass filter to meet specified target values for DC gain and cutoff frequency. Once you have confirmed that your filter design meets specifications you will investigate the filter's ability to suppress high-frequency noise superimposed on a digital-like information signal.

Ulaby/Yagle Section 6-8

7.2 Objectives

1. Design a filter transfer function
2. Design and evaluate an active filter
3. Compare simulated and measured frequency response
4. Evaluate a filter's ability to remove high-frequency noise

7.3 Deliverables

1. Hardcopy of all Multisim circuits and annotated screen shots of simulation results used as the basis of a measurement
2. Hardcopy of all ELVISmx instrumentation screen shots annotated to show measurement procedures

3. Lab report or notebook formatted according to instructor's requirements

7.4 Equipment and Software

1. NI Multisim
2. NI ELVIS II
3. TI TL072 dual op-amp
4. Resistors: two (values to be calculated)
5. Capacitors: two to three (values to be calculated)
6. Jumper wire kit

7.5 Preparation

Butterworth filter design

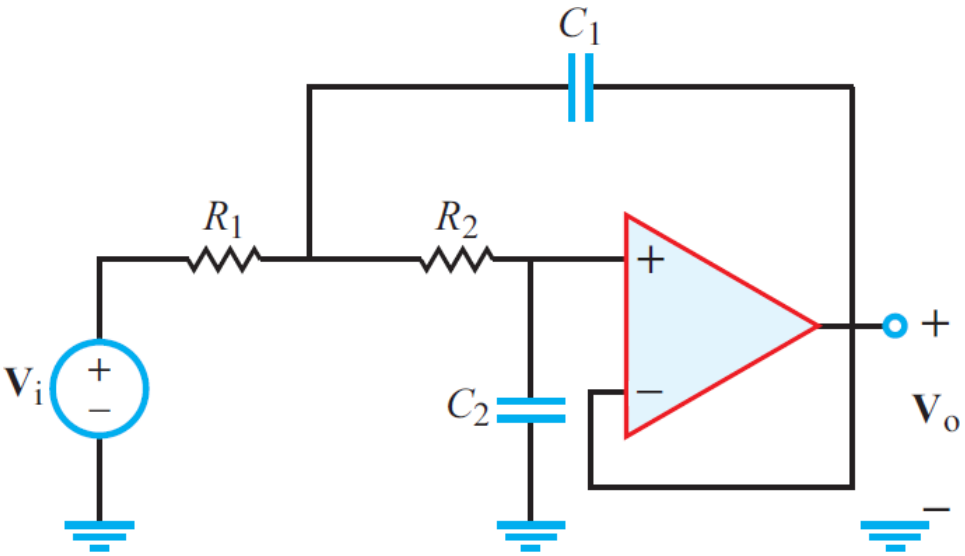
► Study Section 6-8 in your textbook.

□$_1$ Design a second-order Butterworth lowpass filter transfer function to obtain the following target values:

- DC gain = 1 (0 dB) and
- Cutoff frequency f_c = 800 Hz.

Recall that angular frequency is $\omega_c = 2\pi f_c$.

□$_2$ Figure 7.1 on the facing page shows the circuit schematic of the Sallen-Key active filter circuit and the transfer function implemented by the circuit. Match the coefficients of your Butterworth filter transfer function to the Sallen-Key transfer function. Derive equations to calculate the two capacitor values C_1 and C_2 for the special case of $R_1 = R_2 = R$.



$$\mathbf{H(s)} = \frac{\dfrac{1}{R_1 R_2 C_1 C_2}}{\mathbf{s}^2 + \left(\dfrac{1}{R_1 C_1} + \dfrac{1}{R_2 C_1}\right)\mathbf{s} + \dfrac{1}{R_1 R_2 C_1 C_2}}$$

Figure 7.1: Sallen-Key active filter circuit with associated transfer function.

□$_3$ Use your equations to select four readily-available component values from the parts list of Appendix A on page 109 to construct the Sallen-Key circuit. Keep the resistor values between 1 kΩ and 100 kΩ. Report the percent difference between your selected values and the exact values required by your design equations. You may combine two standard-valued components to obtain a needed value, e.g., two parallel capacitors each with value C to yield an effective capacitance $2C$.

7.6 Design evaluation: simulation

► Enter the circuit of Figure 7.1 into NI Multisim. Use the resistor and capacitor values that you will eventually use to build the circuit. Use the

TL072CP dual op-amp model and the AC_VOLTAGE source to apply a sinusoidal input. Place "on-page connectors" to name the signal nets V_{i} and V_{o}.

$\square_4$ Set up an AC analysis to create a Bode-style frequency response plot of V_o/V_i over the range 10 Hz to 10 kHz. Use dB (decibels) scale for the vertical axis.

$\square_5$ Take cursor measurements to determine the DC gain and the cutoff frequency f_c; remember that the gain at the cutoff frequency is 3 dB down from the gain at DC. Report the percentage difference between your simulation results and the target design values; comment on the level of agreement between these two sets of results.

► Iterate on your design (recalculate component values and resimulate) until the DC gain is within 1 dB of the target DC gain and the cutoff frequency is within 10% of the target f_c.

Video tutorials

Study the following video tutorials that relate to this project:

- NI Multisim:
  - Find components by name:
    http://youtu.be/5wlFweh4n-c
  - AC (sinusoidal) voltage source:
    http://youtu.be/CXbuz7MVLSs
  - VCC and VEE power supply voltages:
    http://youtu.be/XkZTwKD-WjE
  - Display and change net names:
    http://youtu.be/0iZ-ph9pJjE
  - Measure frequency response with AC Analysis:
    http://youtu.be/tgCPDBtRcso
  - Plot mathematical expression:
    http://youtu.be/qYVf_rkTF-o
  - Set cursor to a specific value:
    http://youtu.be/48sQja58I10

- NI ELVISmx:
  - Function Generator (FGEN):
    `http://decibel.ni.com/content/docs/DOC-12940`
  - Arbitrary Waveform Generator (ARB):
    `http://decibel.ni.com/content/docs/DOC-12941`
  - Oscilloscope:
    `http://decibel.ni.com/content/docs/DOC-12942`
  - Bode Analyzer:
    `http://decibel.ni.com/content/docs/DOC-12943`

7.7 Design evaluation: hardware

► Build the circuit of Figure 7.1 on page 63:

- Use the TI TL072C dual op-amp or equivalent device; refer to Appendix B on page 111 for details. Power the op-amp with the NI ELVIS II $\pm$15-volt dual supply.
- Create the signal input V_{in} with NI ELVIS II FGEN and monitor this signal with AI6+. Connect AI6- to AIGND.
- Set up the ELVISmx Function Generator to drive the filter input with a 1V amplitude sinusoid.
- Set up the ELVISmx Oscilloscope to observe the filter input V_{i} on Channel 0 and V_{o} on Channel 1 with connections to AI7+. Connect AI7- to AIGND.

► Observe the filter input and output while you vary the frequency of the function generator over a large range. Confirm that the circuit behaves like a lowpass filter with unity gain in the passband and substantial attenuation in the stopband.

$\square_6$ Set up the ELVISmx Bode Analyzer to plot the frequency response of the filter:

- Stimulus Channel: AI6
- Response Channel: AI7

- Start Frequency: 10 Hz
- Stop Frequency: 9.0 kHz
- Steps: 20 per decade
- Peak Amplitude: 1.00
- Mapping: Logarithmic

$\square_7$ Set up the Bode Analyzer to sweep in a small frequency range from 700 to 900 Hz; increase the step size as needed to obtain a good measurement of the -3 dB frequency. Compare this measurement to your simulated value for f_c as well as the design target value.

7.8 Filter high-frequency noise

In this section you will create a digital-like information signal (squarewave) that has been corrupted with high-frequency sinusoidal noise and investigate how the lowpass filter passes the information signal while suppressing the noise.

$\square_8$ Perform a single-frequency AC simulation in Multisim to determine the filter's gain at 200 Hz and also at 5 kHz.

$\square_9$ Predict the filter's output amplitude for a 1-volt amplitude 200 Hz sinusoid applied to the filter input. Repeat for a 4-volt amplitude 5 kHz sinusoid.

► Set up the ELVISmx Arbitrary Waveform Generator to create the sum of a 1-volt amplitude 200 Hz squarewave (the information signal) and a 4-volt amplitude 5 kHz sinusoid (the interfering noise signal) on analog output AO0, and apply this signal to the filter input. Use the maximum available sampling frequency of 200 kS/s and a 5 ms segment length.

$\square_{10}$ Observe the filter input and output for the signal-plus-noise waveform. Take screen shots of the input waveform and the output waveform individually.

$\square_{11}$ Discuss how well the Butterworth filter removes the high-frequency noise.

Laboratory Project 8

AM Radio Simulator

8.1 Synopsis

In this lab project you will create an AM radio simulator that features an audio signal viewer to simultaneously display the frequency spectrum and time-domain representations of any desired signal in the system and play the signal on the sound card. The specific baseband and carrier frequencies specified in this project make it possible for you to listen to the spectrum you see, thereby enhancing your intuitive insights about the AM modulation and demodulation process.

Ulaby/Yagle Section 6-11

8.2 Objectives

1. Develop an intuitive understanding of the modulation and demodulation process
2. Build an AM radio simulator in LabVIEW

8.3 Deliverables

1. Hardcopy of all LabVIEW block diagrams and front panels that you create
2. Screen shots of requested spectrum/time plots

3. Lab report or notebook formatted according to instructor's requirements

8.4 Required Resources

1. NI LabVIEW

8.5 Preparation

AM radio theory

► Study Section 6-11 in your text.

► Figure 8.1 on the next page shows the block diagram of the LabVIEW AM radio simulator that you will construct in this lab project. Compare the section labeled "Three AM transmitters" to Figures 6-58 and 6-64(b) in your text; this section of the LabVIEW VI models three AM transmitters operating at three distinct carrier frequencies. Also compare the section labeled "AM receiver" to Figure 6-63 in your text. The LabVIEW VI directly implements the block diagram of the superheterodyne receiver pictured in your text.

8.6 AM Transmitters

► Create an empty top-level VI to serve as the starting point of your AM radio simulator application.

Global variables

► Global variables reduce block diagram wiring clutter, an especially useful technique for this project because most subVIs require the same value such as system sampling frequency. Study the video `http://youtu.be/PquslTj69f4` to learn how to create a global variable, how to place it in a VI, how to change the variable from write mode to read mode, and how to extend the number of elements contained in the global variable.

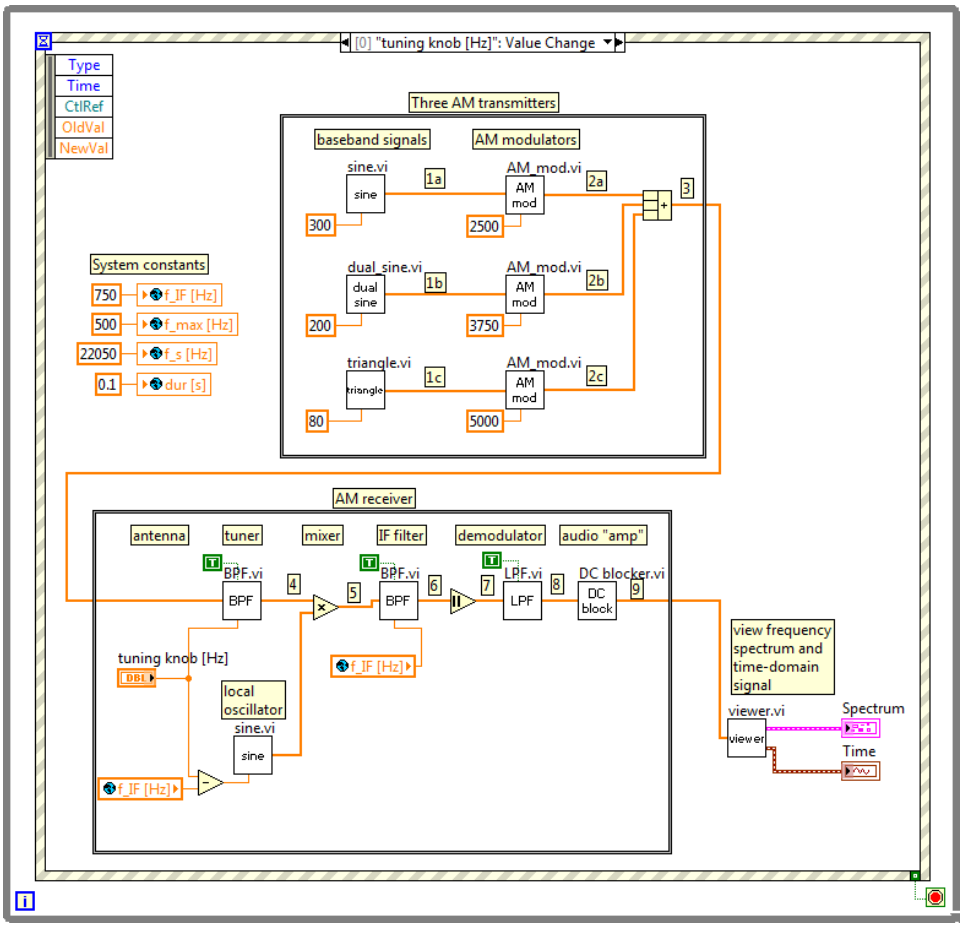


Figure 8.1: Top-level VI block diagram of the AM radio simulator.

► Create the global variable VI `global.vi` that contains four controls of type `DBL`:

- `f_s [Hz]` – system sampling frequency in Hz; default value = 22050
- `dur [s]` – signal duration in seconds; default value = 0.1
- `f_max [Hz]` – maximum frequency in Hz of the baseband signal sources, also called signal bandwidth; default value = 500
- `f_IF [Hz]` – intermediate frequency in Hz used by the AM receiver; default value = 750

► Place four global variables on your radio simulator VI as in the "System Constants" area of Figure 8.1 on the preceding page. Create constants to document the values of the global variables.

Baseband signal sources

► Follow along with the video `http://youtu.be/C2zYeC_RpM8` to create a sinusoidal signal generator subVI `sine.vi` that accepts the desired oscillation frequency `f0 [Hz]` as an input and produces the signal as an array output.

► Follow along with the video `http://youtu.be/R3h4_uR9Dcc` to create an audio signal viewer subVI `viewer.vi` that accepts a signal array as input and produces two graph outputs, one to show the frequency spectrum of the signal and the other to show the time-domain signal. Also include the `Play Waveform` Express subVI to listen to the signal on the soundcard.

> **IMPORTANT:** Protect your hearing! Before running this VI when wearing ear buds or headphones, first turn the volume all the way down, and then gradually increase the volume to a comfortable level. In later steps this VI produces high-frequency sounds that could damage your hearing if you have the volume too high.

► Place and connect `sine.vi` and `viewer.vi` in your radio simulator VI. Create waveform graph indicators on the front panel to display the spectrum and time-domain signals. Try several values of oscillation frequency, and confirm that the displays look correct, and also confirm that you can hear the signal on your soundcard.

► Copy your sinusoidal signal generator subVI file to a new file `triangle.vi`; edit this subVI to create a triangle-wave signal based on the built-in `Triangle Wave` subVI found in the same subpalette as `Sine Wave`.

► Copy your sinusoidal signal generator subVI file to a new file `dual_sine.vi`; edit this subVI to create a dual-sine tone: the first tone frequency is f_0 with unit amplitude and the second tone frequency is 1.7 times f_0 amplitude 0.7.

$\square_1$ Create screen captures of your radio simulator VI front panel for each of your three baseband signal sources, each set to 500 Hz. Adjust the upper limit of the time-domain time axis so that you can clearly see several periods of the waveform.

► Adjust the three baseband signal frequencies to match those shown on the block diagram of Figure 8.1 on page 71.

AM modulator

► Create the subVI `AM_mod.vi` that modulates its baseband input signal array according the AM modulation equation $y_{\mathrm{m}}(t) = [A + x(t)]\cos(2\pi f_{\mathrm{c}} t)$. Create an input `f_c [Hz]` to serve as the carrier frequency. Select the constant A as $|x_{\min}(t)|$, i.e., the absolute value of the most negative value of the input signal array; see the built-in subVI `Array Max & Min`. This choice of A corresponds to a modulation index of 1; refer to Section 6-11.5 in your text for a discussion of modulation index.

$\square_2$ Update your radio simulator VI to place an AM modulator subVI between the sinusoidal signal generator and the signal viewer. Try several different carrier frequencies and signal frequencies. Discuss in some detail your observations about the effect of AM modulation on the baseband signal.

► Add two more AM modulators, one after each of your two remaining signal sources. Use carrier frequencies 2500, 3750, and 5000 Hz. Add the three modulated signal together (`Compound Arithmetic` works nicely here), and then connect to your signal viewer.

$\square_3$ Create a screen capture of your radio simulator VI front panel. NOTE: At this point your VI models the effect of several AM broadcasters operating in a given locality. Comment on the apparent requirements to keep the transmitters from "stepping" on each other's signals.

8.7 AM Receiver

RF filter

► Follow along with this video `http://youtu.be/gBVluWy_UGs` to create a bandpass filter subVI `BPF.vi` that processes an input signal array to produce an output array. The subVI includes an input to set the desired center frequency f_c as well as a Boolean control input to disable the filter, i.e., to pass the input to the output with no processing. Use a 10th-order Butterworth filter from the `Signal Processing` subpalette with the high cutoff frequency set to f_c+f_{max} and the low cutoff frequency set to f_c-f_{max}.

► Update your radio simulator VI to place a bandpass filter as the first stage of the AM receiver.

Local oscillator and mixer

► Continue updating your radio simulator VI by placing `sine.vi` to serve as the local oscillator, and then a multiplier to serve as the mixer.

► Place a front panel control to serve as the "tuning knob" of the AM receiver; the round knob or horizontal pointer slide works nicely for this purpose. The tuning knob selects the center frequency of the first-stage bandpass filter, thereby choosing the carrier frequency of the desired AM broadcaster. The local oscillator frequency f_{LO} tracks the carrier frequency as $f_{LO} = f_c - f_{IF}$. Note that this is a different choice than that suggested by Equation 6.128 in your text because the intermediate frequency is lower than the lowest broadcaster frequency to emulate the commercial AM radio standard (f_{IF}=455 kHz for f_c between 530 and 1610 kHz).

► Now is an appropriate time to enclose your design in an *event structure* to make your VI interactive. Follow along with the video `http://youtu.be/eGlvOiqYVxg` to learn how to place the event structure, adjust it to respond to changes in the tuning knob control, and exit when the "stop" button is pressed.

> **NOTE:** The event structure does not trigger when you first run the VI, giving the impression that nothing happened as a result of modifying the block diagram. Nudge the tuning knob's digital display value up and then back down to see the result

of a modification without changing the tuning knob's value; right-click the tuning knob and select "Visible Items" to enable the digital display. For a more sophisticated approach, study the video tutorial `http://youtu.be/UFEvmmggqLQ` to learn how to execute the event structure one time when you first run the VI.

$\square_4$ Connect your signal viewer to the mixer output. Create a Boolean constant for the bandpass filter enable; set the constant to "false" so that the filter is disabled. Experiment with different values of the tuning knob ranging from 750 Hz and up to 5 kHz; double-click each limit of your front-panel knob and enter these limits. Discuss how the spectrum changes in response to different values of the local oscillator frequency. In particular, comment on how the spectrum of a desired station can be translated to the intermediate frequency.

$\square_5$ Now enable the bandpass filter and continue your experimentation with the tuning knob. Discuss the primary benefit of the bandpass filter as regards translation of the desired station to the intermediate frequency.

► Adjust the limits on your tuning knob to select among only the carrier frequencies pictured on the diagram of Figure 8.1 on page 71.

IF filter

$\square_6$ Place a second bandpass filter after the mixer output; this is the IF filter, consequently set the filter center frequency to f_{IF}. Connect your signal viewer to the IF filter output and experiment with the tuning knob again. Discuss the primary benefit of the IF filter as regards translation of the desired station to the intermediate frequency.

Demodulator

► Review Figure 6-60 in your text; this envelope detector circuit serves as the AM demodulator and can be implemented by an absolute value operator followed by a lowpass filter.

► Copy your bandpass filter subVI file to a new file `LPF.vi`; edit this subVI to create a lowpass filter; change the filter type to "Lowpass" and set the low cutoff frequency set to $f_{\max}$. Disconnect the high cutoff frequency wire, and then delete the front panel control for f_c.

$\square_7$ Update your radio simulator VI with the absolute value node and lowpass filter. Connect your signal viewer to the lowpass filter output. Create a Boolean constant for the lowpass filter enable; set the constant to "false" so that the filter is disabled. Experiment with the tuning knob to select each of the three stations. Discuss how the spectrum changes in response to different values of the local oscillator frequency. Specifically comment on how the absolute value operator modifies the spectrum.

$\square_8$ Now enable the lowpass filter and repeat your experimentation with the tuning knob. Discuss the role of the lowpass filter.

DC blocker

The audio amplifier is the final stage of the superheterodyne receiver of Figure 6-63 in your text. While amplification is not critical for this simulation, the DC blocking feature of the audio amplifier is worth simulating.

► Create the subVI `DC_blocker.vi` to remove the DC component (average value) of the demodulator output. See the built-in subVI `AC & DC Estimator` in the signal processing subpaletter that calculates the average (DC) value of an array.

► Place `DC_blocker` as the final stage of your radio simulator VI. Connect the signal viewer and ensure that the DC value is indeed removed from the signal.

8.8 Discussion

► Set the tuning knob to select the triangle-wave baseband signal. Adjust the vertical axis of the spectrum display to the range 0 to 1 (turn off auto scaling, too), adjust the vertical axis of the time-domain display to the range -1 to 1 (turn off auto scaling), and adjust the horizontal axis of the time-domain display to the range 0 to 0.05; in this way you have a common axis scaling for all plots to be collected.

$\square_9$ Collect a screen shot of your radio simulator front panel at each labeled signal point in Figure 8.1 on page 71, i.e., 1c, 2c, and 3 to 9.

$\square_{10}$ Discuss how will the final AM receiver output matches the original baseband signal.

$\square_{11}$ Discuss your new understanding of AM radio. Specifically comment on any new insights that you gained.

Laboratory Project 9

Convolution Reverb Audio Effect

9.1 Synopsis

Discrete convolution serves as the foundation of a popular audio processing technique called *convolution reverberation* or "convolution reverb" for short. Convolution reverb uses the impulse response (IR) recorded within a physical room or hall to artificially add reverberation to an audio signal such as an instrument or voice.

In this lab project you will gain familiarity with the LabVIEW built-in subVI Convolution as a means to convolve small sequences, and then apply this subVI to implement your own convolution reverb audio effect.

Ulaby/Yagle Section 7-5

9.2 Objectives

1. Evaluate the discrete-time convolution sum by hand
2. Confirm by-hand calculations with the LabVIEW Convolution subVI
3. Implement a convolution reverb effect using speech and recorded impulse response (IR) audio files

9.3 Deliverables

1. Hardcopy of all LabVIEW block diagrams and front panels that you create

2. .zip file containing all audio files that you create
3. Lab report or notebook formatted according to instructor's requirements

9.4 Required Resources

1. NI LabVIEW with MathScript RT Module
2. Speech .wav file from the Open Speech Repository, `http://www.voiptroubleshooter.com/open_speech`
3. Impulse response .wav file from Open Air, `http://www.openairlib.net`
4. Resample Audio .wav VI `http://decibel.ni.com/content/docs/DOC-23105`

9.5 Preparation

By-hand computation of discrete-time convolution

As an initial warm-up on the subject of discrete-time convolution, view this short video tutorial `http://youtu.be/yyTu0SXeW1M` to learn how to convolve two sequences by hand. Apply what you have learned by convolving the following pairs of sequences:

$\square_1 \quad \{\underline{1}, 2, 3\} * \{\underline{3}, 2, 1\}$

$\square_2 \quad \{\underline{0}, 0, 1, 1, 1, 1, 1\} * \{\underline{0.2}, 0.2, 0.2, 0.2, 0.2\}$

$\square_3 \quad \{\underline{0}, 0, 1, 1, 1, 1, 1\} * \{\underline{-1}, 1\}$

$\square_4 \quad \{\underline{1}, 2, 3\} * \{\underline{1}, 0, 0, 0, 0, 0, 0, 0.75, 0, 0, 0, 0, 0, 0.33, 0, 0, 0, 0, 0, 0.1\}$

Confirm by-hand computation

► Follow along with this video tutorial `http://youtu.be/Fsf88Rmimz4` to learn how to create a LabVIEW VI to evaluate the convolution of two sequences, and to visualize the input and output sequences as arrays and as stem plots.

$\square_5$ Run your VI for each of the sequence pairs above. TIP: Delete an element from the array by right-clicking on the element and choosing "Data Operations | Delete Element." Collect a screen shot of the front panel for each sequence pair, and then compare to your by-hand calculations. Reconcile any differences between the two by correcting your by-hand calculation or by revising your LabVIEW code.

Download and prepare audio files

► Visit the Open Speech Repository, `http://www.voiptroubleshooter.com/open_speech`. Follow the link for your language of choice, and then download one or more of the speech audio clips.

► Visit Open Air, `http://www.openairlib.net`. Follow the "IR Data" tab at the top of the page and investigate the available rooms and halls. After you select a specific location, click the "Impulse Responses" tab. Seek an impulse response (IR) audio file that is mono, i.e., single channel. Download one or more of the available IR audio files.

► Download the LabVIEW VI Resample Audio .wav available at `http://decibel.ni.com/content/docs/DOC-23105`. Use this application to produce a set of speech and IR audio files that have the same sampling frequency. Choose a standard sampling frequency such as 44.1 kHz.

9.6 Convolution reverb audio effect

Reverberation is a mass of echoes that cannot be individually perceived as discrete echos. Begin your design by creating a simple IR that contains distinct echoes for testing purposes.

► Study the video tutorial `http://youtu.be/ChSVf9lZ44k` to learn how to read and write .wav audio files, how to extract the audio file as a 1-D array for processing, and how to listen to the audio within the VI.

$\square_6$ Create a LabVIEW VI that generates an IR .wav file containing four unit values (impulses) separated by silence (zero values) at intervals 50, 500, and 900 ms. That is, the signal array looks like a 1 at $n = 0$ followed by 50 ms of silence, and then another 1, and then 500 ms of silence, and so

on. Remember to calculate the number of zeros in the silent regions using the same sampling frequency that you used for your speech and IR files. MathScript or a combination of Initialize Array and Replace Array Subset works well for this part. NOTE: Type `Ctrl+Space` to search for LabVIEW subVIs by name.

$\square_7$ Create a LabVIEW VI to read your speech and IR .wav files, convolve them, and then write the processed audio to a file.

- Remove the Convolution "algorithm" constant to revert back to the default method "frequency domain."
- IMPORTANT: Place the Quickscale subVI in the "Signal Processing | Signal Operations" subpalette just after the convolution subVI to ensure the audio signal does not exceed the ± 1 signal limits which would result in audible clipping noise.
- Add waveform graphs to plot the speech, IR, and processed audio arrays.
- Include three instances of the Play Waveform Express subVI to listen to each audio signal; connect the output error cluster of the first subVI to the input error cluster of the next subVI to guarantee the correct order of audio playback as speech, the IR, and then the processed audio.

$\square_8$ Test your convolution audio effect with the speech audio and the four-echo IR you created earlier. Discuss your test results. Which of the by-hand convolution problems above looks most like the echo IR?

$\square_9$ Now use the IR you downloaded from Open Air to process your speech signal. Discuss the degree of realism achieved by your convolution reverb audio effect.

$\square_{10}$ OPTIONAL: Once you have your reverb audio effect operational, you may enjoy creating additional sound effects:

- Reverse reverb – Reverse the IR signal array with Reverse 1-D Array in the "Programming | Array" subpalette.

- Non-reverb-based IR – *Any* audio signal can serve as an IR, not just those obtained by measuring a room. Try a short audio clip such as a speech utterance or a sound effect.

Laboratory Project 10

Notch and Comb Filters

10.1 Synopsis

Sinusoidal interfering tones in audio signals can be reduced or eliminated altogether with notch and comb filters. A notch filter transfer function $H(z)$ has zero response at the interfering tone frequency yet nearly unit response at all other frequencies, thereby leaving most of the audio intact. A comb filter provides zero response at harmonics as well as the fundamental frequency, making it possible to remove non-sinusoidal yet periodic interfering sounds.

In this project you will analyze audio from the radio communication between Neil Armstrong and NASA Mission Control to determine the frequencies of the "NASA beeps" and then design notch and comb filters to eliminate the beeps. The resulting beep-free audio emulates the radio conversation as heard by the astronauts.

Ulaby/Yagle Sections 8-2 and 8-3

10.2 Objectives

1. Interpret a signal's magnitude spectrum to search for periodic signals
2. Process audio signals and work with audio files
3. Implement and evaluate the performance of a notch filter
4. Implement and evaluate the performance of a comb filter

10.3 Deliverables

1. Hardcopy of all LabVIEW block diagrams and front panels that you create
2. Screen shots of requested spectrum/time plots
3. Two audio .wav files
4. Lab report or notebook formatted according to instructor's requirements

10.4 Required Resources

1. NI LabVIEW with MathScript RT Module
2. Audio file "The Moon Landing" at `http://archive.org/details/Greatest_Speeches_of_the_20th_Century`

10.5 Preparation

Moon landing audio

An estimated 500 million people listened to Neil Armstrong's famous radio conversation with NASA Mission Control on July 20, 1969.

► Visit `http://archive.org/details/Greatest_Speeches_of_the_20th_Century` and listen to "The Moon Landing" audio clip.

$\square_1$ Do an Internet search to determine the following information about the "NASA beeps" that you hear in the audio, more properly called *Quindar tones*:

1. The significance of the name "Quindar,"
2. The purpose of Quindar tones,
3. The waveform type,
4. The two tone frequencies, and
5. The duration of the tones.

Include URLs for the articles from which you obtained this information.

► Download the MP3 audio file collection "64Kbps MP3 ZIP" linked on the left side of the "Greatest Speeches" page. This file contains the entire collection; extract the MP3 file for "The Moon Landing."

► Install the Audacity software available at `http://audacity.sourceforge.net`. Type `Ctrl+Shift+I` to import the MP3 audio file, and then select `File | Export` to save the audio as a WAV 16-bit audio file. Feel free to use an existing package of your choice to perform the file format conversion, but ensure that you create a 16-bit WAV-format audio file.

Notch and comb filter theory

In this project you will create and evaluate two filter methods to eliminate the Quindar tones from the audio recording.

► Study Sections 8-2 and 8-3 in your text.

10.6 Audio Analysis

In this section you will analyze the audio file to precisely measure the actual Quindar tone frequencies.

► Create an empty top-level VI to serve as the starting point of your audio filter application.

► Follow along with the video `http://youtu.be/ChSVf9lZ44k` to learn how to read a .wav audio file, extract the 1-D signal array, and play the audio to your soundcard. Make note of the demonstration showing how to save a processed audio signal to a .wav file, because you will need this at the end of the project.

► Add block diagram code to calculate the audio file length in seconds and the sampling rate in Hz, and display these values on the front panel.

► Study the help page for "Array Subset." Add controls to specify the start time (in seconds) and the duration (also in seconds) of a subrange of audio. The "Horizontal Pointer Slide" control style works well. Enable the digital display for fine control of the desired time values; right-click on the

control, choose "Visible Items," and then check "Digital Display." Create a waveform indicator to display the audio subrange; connect the waveform data type to the waveform indicator to ensure that the horizontal time axis is properly calibrated in seconds. Add the "Play Waveform" Express subVI to listen to the audio subrange, too. Confirm that you can select specific subranges of audio.

► Study the tutorial video `http://youtu.be/R3h4_uR9Dcc?t=53s` to learn how to use the "Spectral Measurements" Express VI to display the frequency spectrum of the audio signal. Place this VI and create a spectrum display next to your waveform indicator. Set the plot type to "stem" style by clicking the plot legend on the upper right corner of the indicator.

$\square_2$ Adjust the start time and duration to isolate a single Quindar tone. Feel free to use Audacity to analyze the audio waveform, too. Record the start time and duration of your isolated tone. Estimate the frequency from your spectrum display. To obtain more accuracy, place and connect the `Extract Multiple Tone Information` subVI from the `Signal Processing | Wfm Measure` subpalette. Create an indicator and extend the indicator size to show five tone measurements. Take note of the detected frequency with the highest amplitude. Record your final estimate of this first Quindar tone frequency.

$\square_3$ Move to the next Quindar tone frequency and repeat the previous measurements. You should find that the frequency is measurably different.

$\square_4$ Report the difference in Hz and the percent difference between your measured Quindar tones and the published specifications that you found from your research. Propose an explanation for the difference.

10.7 Notch Filter Design

According to Equation 8.12 in your text, the filter transfer function of a notch filter written in terms of $\mathbf{z}^{-1}$ is

$$\mathbf{H}(\mathbf{z}) = \frac{1 - 2\cos\Omega_{\mathrm{i}}\mathbf{z}^{-1} + \mathbf{z}^{-2}}{1 - 2R\cos\Omega_{\mathrm{i}}\mathbf{z}^{-1} + R^2\mathbf{z}^{-2}}$$

where $\Omega_\mathrm{i} = 2\pi(f_\mathrm{i}/f_\mathrm{s})$, f_i is the interfering tone frequency in Hz, and f_s is the sampling frequency, also in Hz. The numerator places a zero on the unit circuit at frequency Ω_i while the denominator places a pole just inside the unit circle at the same frequency; the constant R is the radial distance from the origin to the pole, and filter stability requires $|R| < 1$.

$\square_5$ Design one notch filter for each Quindar tone; call these $\mathbf{H}_1(\mathbf{z})$ and $\mathbf{H}_2(\mathbf{z})$. Use the same filter constant R for both notch filters.

$\square_6$ Cascade the two notch filters to create a single system function $\mathbf{H}(\mathbf{z}) = \mathbf{H}_1(\mathbf{z})\mathbf{H}_2(\mathbf{z})$; expand the products of the numerator and denominator polynomials to obtain a ratio of two polynomials.

► Create a MathScript structure that accepts the audio sampling frequency, filter constant R, and two Quindar tone frequencies as inputs and produces the difference equation a and b coefficient arrays. The tutorial video `http://youtu.be/IsbItFzzOdA` shows how to translate a transfer function $\mathbf{H}(\mathbf{z})$ into a MathScript structure that in turn feeds the coefficient inputs of the digital filter subVI `IIR Filter`. Note that the MathScript convolution function `conv` conveniently calculates the numerator and denominator coefficient arrays of $\mathbf{H}(\mathbf{z})$ given the individual $\mathbf{H}_1(\mathbf{z})$ and $\mathbf{H}_2(\mathbf{z})$ arrays.

$\square_7$ Place the `Plot Pole-Zero` and `Plot Frequency Response` VIs (available at `https://decibel.ni.com/content/docs/DOC-24032` and `https://decibel.ni.com/content/docs/DOC-24021`) and connect these to the outputs of your MathScript structure. Study the video tutorials `http://youtu.be/S8WmBxIlnEc` and `http://youtu.be/IsjXautfUXs` to learn how to use these subVIs. Create default indicators for the frequency response magnitude and pole-zero plots. Capture a screen shot of these two indicators for your filter system function $\mathbf{H}(\mathbf{z})$. Confirm that the poles and zeros appear in the correct locations (zeros on the unit circle, poles just inside the unit circle), and also confirm that the frequency response goes to zero at the correct frequency.

► Place the `IIR Filter` subVI from the `Signal Processing | Filters | Advanced IIR` subpalette. Connect your a coefficients to the `Reverse Coefficients` input and your b coefficients to the `Forward Coefficients` input. Insert the filter just before your waveform and spectrum displays

and the audio player. Include a Boolean control to selectively enable the filter, i.e., include a bypass path and a `Select` node from the `Programming | Comparison` subpalette.

$\square_8$ Adjust the audio subrange start and duration times to your first Quindar tone. Collect screen shots of the spectrum with and without the notch filter enabled. Turn off the vertical axis auto-scaling to use a common amplitude scale. Report the amount by which the Quindar tone spectral component is reduced.

$\square_9$ Repeat for the second Quindar tone.

► Study the tutorial video `http://youtu.be/unuRYWUYQy4` to learn how to temporarily disable a portion of the block diagram with a Diagram Disable structure, much as you would "comment out" a portion of code in a text-based programming language. Place the two spectrum-related VIs inside the Diagram Disable structure, and disable them for very long duration subregions, e.g., greater than 45 s. The spectrum-related VIs consume too much memory when the subregion is long.

$\square_{10}$ Now, extend the duration to critically listen to a longer audio clip; record the start and duration times. Experiment with values of R, starting with a value of 0.9. Discuss the degree to which the single notch filter suppresses the Quindar tone sound.

$\square_{11}$ Capture a screen shot of the waveform and spectrum displays for the longer audio clip with and without filtering.

10.8 Comb Filter Design

You most likely can hear some harmonic residue of the Quindar tones, even with the notch filters engaged. In this section you will extend the notch filter to a comb filter to deal with these harmonics, also called *overtones*.

$\square_{12}$ Create new spectrum plots similar to those that you collected earlier for the individual Quindar tones, and enable the notch filter. Look for the second harmonic (at twice the frequency of the fundamental) and estimate its amplitude as a percentage of the fundamental's amplitude.

► Extend your MathScript structure to add two more notch filters at precisely twice the frequencies of the two existing notch filters.

$\square_{13}$ Capture a screen shot of the pole-zero diagram and frequency response indicators for your comb filter system function.

$\square_{14}$ Set the start and duration times to those of your earlier critical listening study. Experiment with different values of R; try to find the highest value that just causes the Quindar tones to become inaudible, and then report this value. Discuss the degree to which the comb filter suppresses the Quindar tone sound.

$\square_{15}$ Capture a screen shot of the waveform and spectrum displays for the longer audio clip with comb filtering enabled.

$\square_{16}$ Select an audio subregion of approximately 15 s duration that demonstrates the effectiveness of your comb filter. Create one .wav file with the filter disabled, and a second .wav file with the filter enabled. Submit these two files as part of your deliverables.

10.9 Discussion

$\square_{17}$ Discuss the overall performance of your two filter designs (notch and comb).

Laboratory Project 11

Image Deblurring

11.1 Synopsis

Photographic image capture requires a finite amount of time to gather sufficient light energy to expose the image sensor. Relative motion between the camera and the subject during exposure causes motion blur that smears image details. To the casual observer the blur renders the captured image useless. However, *deconvolution* of the blurring mechanism can recover a close approximation of the original non-blurred image.

In this project you will be presented with an image that suffers from *uniform motion blur*, the special case of constant relative velocity between the camera and subject. With knowledge of the blurring mechanism you will create and implement an *inverse filter* to deblur the image.

Ulaby/Yagle Section 8-4

11.2 Objectives

1. Work with image files; process and display images
2. Estimate motion blur with a mouse cursor measurement
3. Implement a stable inverse filter for a non-minimum phase system
4. Deblur an image with deconvolution

11.3 Deliverables

1. Hardcopy of all LabVIEW block diagrams and front panels that you create
2. Screen shots of requested images and plots
3. Deblurred image file (PNG format)
4. Lab report or notebook formatted according to instructor's requirements

11.4 Required Resources

1. NI LabVIEW with MathScript RT Module
2. Blurred image supplied by instructor (PNG format)

11.5 Preparation

Image handling in LabVIEW

Images are two-dimensional signals with horizontal and vertical dimensions. LabVIEW can work with images as either 2-D or 1-D arrays. For the sake of simplicity in this project the image signal will be treated as a 1-D signal — much like an audio signal — in which the image rows are laid end to end beginning at the top of the image. In this scheme the first array element is the upper left corner image pixel and the last array element is the lower right corner image pixel.

► Study the tutorial video `http://youtu.be/Z_81P1d_rCg` on image handling in LabVIEW; topics include reading and writing PNG image files, image display, obtaining image information, and extracting the image pixels as a 1-D array. You need only watch through time 5:00.

► Study the tutorial video `http://youtu.be/pw7-smkZh_U` to learn how to make the picture indicator automatically size itself to the image.

Image blurring

Your instructor has supplied you with a blurred image of a subject that was originally sharp and detailed. The blurred image resulted from uniform motion in the horizontal direction by an unknown distance of D pixels where D is an integer. Uniform motion blur is identical to the *moving average* (MA) system described in Section 7-3.4 of your text; see Equation 7.34 for the difference equation of the MA system. Substituting blurring distance D for the constant M, setting the b_i coefficients to 1, and $\mathbf{z}$-transforming this equation leads to the blurring system

$$\mathbf{H}(\mathbf{z}) = \frac{1}{D+1} \sum_{i=0}^{D} \mathbf{z}^{-i}. \tag{11.1}$$

Dividing by $D + 1$ ensures that the system has unity DC gain to keep the blurred image at the same average intensity as the original image.

Image analysis

The motion blur distance can be estimated by measuring the length of streaks traced out by the blur of small but distinct features.

► Create a new LabVIEW VI as a starting point for your image analysis application, and then load and display the blurred image provided by your instructor.

► Study the tutorial video `http://youtu.be/nVIOnz2WYyQ` to learn how to display mouse coordinates. Note that the while-loop and event structure need not enclose your existing code that reads the PNG image, but should instead be placed in its own area of the block diagram.

$\square_1$ Update your VI to implement the mouse coordinate display, and then estimate and report the blurring distance D.

11.6 Inverse Filter Design

The inverse filter (also called image deblurring filter) seeks to undo the effects of the image blurring filter. The inverse filter $\mathbf{G}(\mathbf{z})$ satisfies the property $\mathbf{G}(\mathbf{z})\mathbf{H}(\mathbf{z})$=1.

Basic inverse filter

► Create a new LabVIEW VI as a starting point for your image deblurring application, and then load and display the blurred image provided by your instructor.

$\square_2$ Study the tutorial video `http://youtu.be/JJZ-shHiayU` to learn how to write the blurring system $\mathbf{H}(\mathbf{z})$ (Equation 11.1 on the preceding page) in closed form, i.e., without a summation symbol.

$\square_3$ Write the inverse filter system $\mathbf{G}(\mathbf{z})$ for your closed-form version of $\mathbf{H}(\mathbf{z})$.

► Study the video `http://youtu.be/IsbItFzzOdA` to learn how to translate a system transfer function into a MathScript structure; also study the section near the end of the video that shows how to connect the filter coefficient arrays for the a and b coefficient arrays to the LabVIEW `IIR_Filter.vi` subVI.

► Implement the system $\mathbf{G}(\mathbf{z})$ as a MathScript structure. Create an input for pixel distance D and two outputs for the a and b filter coefficient arrays. The MathScript `zeros` function comes in handy to implement the denominator coefficients; use the form `zeros(1,N)` to create a 1-D array of zeros of length N.

$\square_4$ Place the `Plot Pole-Zero` and `Plot Frequency Response` VIs (available at `https://decibel.ni.com/content/docs/DOC-24032` and `https://decibel.ni.com/content/docs/DOC-24021`) and connect these to the outputs of your MathScript structure for $\mathbf{G}(\mathbf{z})$. Study the video tutorials `http://youtu.be/S8WmBxIlnEc` and `http://youtu.be/IsjXautfUXs` to learn how to use these subVIs. Create default indicators for the frequency response magnitude and pole-zero plots of $\mathbf{G}(\mathbf{z})$. Study the effect of blurring distance D on the two plots and discuss your observations.

$\square_5$ Where are the poles of $\mathbf{G}(\mathbf{z})$ located in the pole-zero diagram? How do these pole locations manifest themselves in the frequency response plot?

Stable inverse filter

The basic inverse filter $\mathbf{G}(\mathbf{z})$ is not BIBO stable because the image blurring system $\mathbf{H}(\mathbf{z})$ is not minimum phase. Pulling the poles of $\mathbf{G}(\mathbf{z})$ slightly inside the unit circle makes the deblurring filter stable at the expense of introducing a slight mismatch between the original blurring filter and its inverse filter.

$\square_6$ Apply **z**-transform property #5 ("Multiplication by a^n", also known as the **z**-scaling property) of Table 7-6 in your text to $\mathbf{G}(\mathbf{z})$ by substituting each instance of $\mathbf{z}$ with $\mathbf{z}/R$ to form the stable inverse filter $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$ where R is the pole radius with $|R| < 1$.

► Update the MathScript structure to create the filter coefficients for $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$; add an additional input for R.

$\square_7$ Plot the pole-zero diagram and frequency response of $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$. Investigate values of R in the vicinity of 1.

Unit DC gain

The inverse filter requires a DC gain of 1 to ensure that the average brightness of the deblurred image remains the same as the original image.

$\square_8$ Calculate the DC gain of $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$, i.e., evaluate $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$ at $\mathbf{z} = e^{j0} = 1$.

► Update the MathScript structure to divide the existing $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$ by the DC gain you calculated in the previous step, thereby ensuring that $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$ has unit DC gain for all values of D and R.

11.7 Image Deblurring

► Use the `IIR_Filter.vi` and the a and b coefficients produced by your MathScript structure to calculate the deblurred image.

► Place your deblurring system inside a combined event-structure and while-loop structure so that you can interactively adjust the blurring distance D and the pole radius R while observing the deblurred image. Refer to the tutorial video `http://youtu.be/eGlvOiqYVxg` for details.

► Begin with D set to your estimated blur distance and R set to 0.95. Iteratively adjust both parameters to deblur the image to the best of your ability. Strive for sharp (non-blurry) features, low noise, and minimum artifacts.

► Study the tutorial video `http://youtu.be/IDVHarNkJSU` to learn how to modify your event structure to save your image results to a file when you stop the while-loop. The video describes a general-purpose *load–process–save* design pattern that applies to a wide variety of signal processing tasks.

$\square_9$ Save your deblurred image as a PNG file, and report the values of D and R for this image.

11.8 Discussion

$\square_{10}$ Discuss your results, especially the nature of the noise and artifacts, and their relation to the parameters R and D. Consider including additional example images in your report to substantiate your discussion.

$\square_{11}$ Explain how the original image blurring process actually removes some information from the image, and how this impacts the design of the deblurring filter.

Laboratory Project 12

DTMF Decoder

12.1 Synopsis

Listen to the sound of your telephone keypad when you dial a number. These distinctive sounds known as *DTMF tones* provide a way for your phone to communicate dialing information to the telephone switching system. No doubt you are familiar with the numbers 0 to 9 and the asterisk and pound keys; the DTMF standard defines four additional keys to provide 16 unique tones altogether.

In this project you will create a complete DTMF decoder system based on FFT spectral analysis. Your decoder system will accept an audio recording of a DTMF tone sequence as input and display the detected tones on a grid of visual indicators.

Ulaby/Yagle Sections 8.6–8.7

12.2 Objectives

1. Interpret a signal's magnitude spectrum to search for periodic signals
2. Investigate windowing as a means to reduce spectral leakage
3. Process audio signals and work with audio files
4. Implement an edge detector to automatically identify the location of DTMF tone bursts in an audio waveform

12.3 Deliverables

1. Hardcopy of all LabVIEW block diagrams and front panels that you create
2. Screen shots of requested spectrum/time plots
3. Lab report or notebook formatted according to instructor's requirements

12.4 Required Resources

1. NI LabVIEW with MathScript RT Module
2. Audio file of a mystery DTMF digit sequence provided by instructor

12.5 Preparation

DTMF decoder block diagram

Figure 12.1 shows the high-level block diagram of the DTMF decoder system that you will create in this project. The decoder loads an audio file that contains a DTMF tone sequence, extracts an audio subrange for a single tone, calculates the discrete Fourier transform (DFT) magnitude spectrum for the tone, and decodes the spectral information to determine the corresponding DTMF digit.

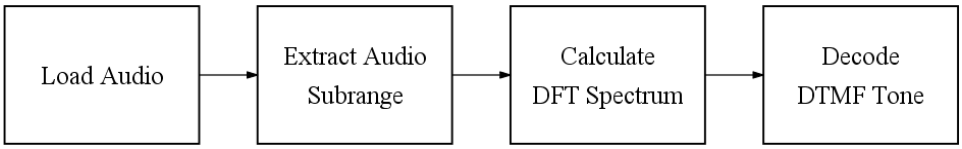


Figure 12.1: High-level block diagram DTMF decoder system.

DTMF tones

Learn about DTMF tones and create your own audio test sequence.

$\square_1$ Do an Internet search to determine the following information about DTMF tones:

1. Meaning of the acronym "DTMF,"
2. The year in which telephones first began to use DTMF,
3. Labels of the 16-key DTMF keypad,
4. Frequencies of the keypad rows, and
5. Frequencies of the keypad columns.

Include URLs for the articles from which you obtained this information.

► Install the Audacity software available at `http://audacity.sourceforge.net`. Run Audacity to create a new project. Set the "Project Rate" (system sampling frequency) to 8 kHz; see the lower left corner of the application. Select the menu command `Generate | DTMF Tones` and then enter the following information into the dialog box:

- "DTMF Sequence" = `123A456B789C*0#D`,
- "Amplitude" = 0.8,
- "Duration" = 5 seconds, and
- "Tone/silence ratio" = 50% (you will see a tone duration of 161 ms with these settings).

Select `File | Export` to save the audio as a WAV 16-bit audio file.

► Alternatively, search for a web service that converts the DTMF digits and letters to a .wav audio file, e.g., `http://www.dialabc.com/sound/generate`. Remember to select 8 kHz for the sampling frequency.

DFT-based spectral analysis

In this project you will use DFT spectral analysis as the heart of your DTMF decoder.

► Study Sections 8-6 and 8-7 in your text.

► Study the tutorial video at `http://youtu.be/z7X6jgFnB6Y` for an overview of DFT spectral analysis.

► Study the tutorial video at `http://youtu.be/UGUceuhSuUE` to learn more about the relationship between the time-domain waveform and its frequency-domain spectrum computed by the *fast Fourier transform* (FFT) algorithm. Note that the terms "DFT" and "FFT" are often used interchangeably.

12.6 Spectral Analysis

In this section you will analyze individual DTMF tones to better understand how to design your DTMF decoder.

► Create an empty top-level VI to serve as the starting point of your DTMF decoder application.

► Follow along with the video `http://youtu.be/ChSVf9lZ44k` to learn how to read a .wav audio file, extract the 1-D signal array, and play the audio to your soundcard. Use the DTMF test sequence audio you created earlier.

► Add block diagram code to calculate the audio file length in seconds and the sampling rate in Hz, and display these values on the front panel.

► Create a waveform indicator to display the complete audio signal.

► Study the help page for "Array Subset." Add controls to specify the start time (in seconds) and the length N (in *samples* – note the difference) of a subrange of audio. The "Horizontal Pointer Slide" control style works well. Enable the digital display for fine control of the desired time and sample values; right-click on the control, choose "Visible Items," and then check "Digital Display."

► Use "Build Waveform" to convert the 1-D array subset to waveform datatype, and connect the waveform data type to a new waveform indicator that displays the audio subrange; ensure that the horizontal time axis is properly calibrated in seconds. Re-connect the "Play Waveform" Express

subVI to listen to the audio subrange instead of the complete signal. Confirm that you can select any desired audio subrange.

► Study the tutorial video `http://youtu.be/9SL5_jY87IA` to learn how to use the "FFT" built-in subVI located in the `Signal Processing | Transforms` subpalette. Place and connect this VI to your audio subrange; use the *same* subrange length control value for the FFT length.

► Add an "Array Subset" node to extract the lower half of the FFT output array (the upper half contains a mirror image of the lower half and does not need to be plotted).

► Divide the FFT output by the subrange length N to normalize the values.

► Place a "Complex to Polar" node from the `Programming | Numeric | Complex` subpalette to compute the FFT magnitude, and then display this array on a waveform plot. Set the plot type to "stem" style by clicking the plot legend on the upper right corner of the indicator. Turn off autoscaling on the Y axis and set the maximum value to 0.12.

$\square_2$ Set the subrange length to 128, and then adjust the start time to isolate a single DTMF tone. Run the VI to see the tone's waveform and spectrum, and also to listen to the tone. Create a screenshot of the spectrum and state the associated DTMF tone digit. Discuss the appearance of the DFT spectrum.

$\square_3$ Repeat the previous step for another DTMF tone, noting the differences that appear in the DFT spectrum.

$\square_4$ Try increasing the FFT length to values 256, 512, and 1024; note that the FFT algorithm works most efficiently with array lengths that are a power of two. Discuss all effects that you observe resulting from longer FFT lengths.

$\square_5$ Derive an equation to calculate the FFT spectral bin number for a given frequency f_0. Recall that the bin number is an integer. Calculate the frequency bin numbers for the two DTMF frequencies for a selected DTMF tone digit with $N = 1024$. Zoom in on the spectral lines in the spectrum plot (right-click on the waveform graph indicator, enable the "Graph Palette" option of "Visible Items," and then use the zoom controls). Create a screen shot of the zoomed plot and mark up the screenshot to demonstrate that your equation properly calculates the FFT bin number for a given DTMF frequency.

► Note the abundance of spectral lines that cluster around the base of the tall spectral line that corresponds to each DTMF frequency. This *spectral leakage* effect can be reduced by *windowing* the waveform array before it is applied to the FFT. Study the tutorial video `http://youtu.be/UGUceuhSuUE` to learn more about spectral leakage and windowing.

$\square_6$ Select the Hamming window subVI from the `Signal Processing | Windows` subpalette and insert this before the FFT. Discuss the reduction in spectral leakage that you observe.

12.7 Tone Detector and Display

By now you have developed a good feel for the spectral representation of DTMF tones, and you also know how to translate any particular DTMF tone frequency to an FFT bin number. In this section you will complete the tone detector and display panel.

$\square_7$ Determine a suitable threshold to distinguish between the presence and absence of all 32 DTMF tone frequencies. Use a "Greater?" node from the `Programming | Comparison` subpalette to threshold the FFT magnitude array; use a front-panel control for the threshold value. Report the threshold value you selected.

► Use the "Index Array" node from the `Programming | Array` subpalette to extract the thresholded frequency bins of the four DTMF keypad row frequencies; pull on the bottom handle of "Index Array" to create four index inputs and outputs. Create four constants for the DTMF keypad row frequencies (in Hz) and then add additional mathematical nodes to convert

the frequencies to bin numbers for the "Index Array" inputs. In this way your DTMF decoder will work properly for any subrange length N that you choose. The end result of this step is four Boolean wires that indicate which DTMF keypad row is active, if any.

► Repeat the previous step to create four Boolean wires for the DTMF keypad columns. Arrange your block diagram layout to make a 4-by-4 grid of Boolean wires.

► The Boolean "AND" gate uniquely detects the two frequencies for a given DTMF digit. Use sixteen AND gates from the `Programming | Boolean` subpalette placed at the intersections of the row and column Boolean wires. Create a Boolean indicator at the output of each AND gate. Label the indicators according to the sixteen DTMF digits and arrange the front-panel indicators in the shape of a standard DTMF keypad.

► Test your finished tone decoder display for each of the sixteen DTMF tones in your audio test sequence.

12.8 Complete DTMF Decoder

At this point in the project you have demonstrated that your design can successfully detect each of the sixteen DTMF tones in an audio sequence. However, you must manually set the start position of each tone. In this section you will add an *edge detector* to automatically find the leading edges of all of the DTMF tone bursts and then loop over all of the tone bursts detected in the audio signal.

Figure 12.2 on the following page shows the block diagram of the DTMF decoder system with a tone-burst edge detector to automatically identify the starting index of each tone burst in the sequence. The edge detector creates an array of start times that control a for-loop structure to repeatedly run the block diagram that you have developed so far.

► Enter the edge detector block diagram code shown in Figure 12.3 on the next page. The edge detector consists of an envelope detector, comparator, differentiator, and threshold detector. Place the needed components by typing "Ctrl+Space" for the "Quick Drop" menu and typing the following names:

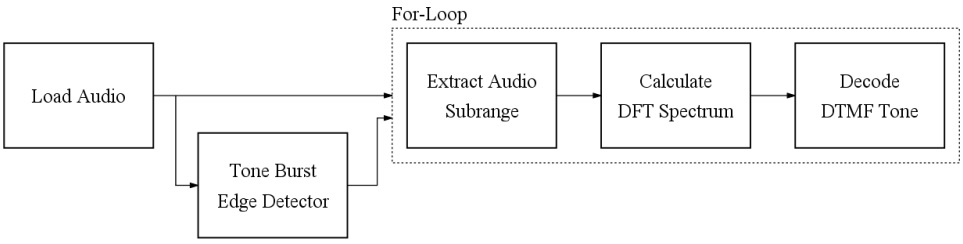


Figure 12.2: High-level block diagram DTMF decoder system with edge detector to automatically identify the starting times of the DTMF tone bursts.

- Absolute Value
- Butterworth Filter
- Greater?
- Boolean To (0,1)
- To Double Precision Float
- Derivative x(t)
- Threshold Detector

Right-click on each component and select "Help" to learn more details.

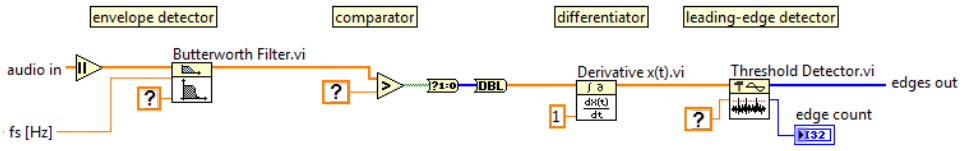


Figure 12.3: LabVIEW block diagram of the edge detector subsystem.

$\square_8$ Display the envelope detector output as a waveform plot. Experiment with the Butterworth filter's "low cutoff frequency" value. You want the

filter output to accurately track the outline (envelope) of the DTMF tone bursts. Create a screen shot of the envelope detector output and report the final value you selected for the cutoff frequency.

$\square_9$ Display the comparator output as a waveform plot. Select a threshold value that produces a clean squarewave representation of the DTMF tone burst envelope. Create a screen shot of the comparator output and report the final value you selected for the threshold value.

$\square_{10}$ Display the differentiator output as a waveform plot; create a screen shot of the plot.

$\square_{11}$ Add the "Threshold Detector" to determine the index values of the *leading edges* of the tone bursts. Select a threshold value to detect leading edges only; report your final value. Create an indicator for the "count" output to display the number of detected edges.

► Determine which parts of the existing block diagram to enclose in a for-loop structure. Wire the output of the "Threshold Detector" subVI into the for-loop structure using the default "Auto-Indexed Tunnel;" in this way the loop will execute one time for each DTMF tone burst. Include a "Wait" in the for-loop to set the pace of the tone playback and display.

► Test your complete DTMF decoder with your own audio test sequence and confirm that it works properly.

$\square_{12}$ Run your decoder with the instructor's mystery DTMF sequence. Report the DTMF digit sequence you discovered.

12.9 Discussion

$\square_{13}$ Discuss the overall performance of your DTMF decoder system.

Appendix A

Parts List

Resistors

The following resistors are standard-value 5% tolerance 1/4 watt carbon film devices. All listed resistors are available in resistor kits from Digi-Key, Jameco, and RadioShack with the following exceptions: (1) the Digi-Key kit does not contain a 10 MΩ resistor, and (2) none of the kits contain a 2 MΩ resistor (this can be created with two series-connected 1 MΩ resistors):

Resistor Kit Description	Supplier	Part #
365 pcs, 5 ea of 1.0Ω to 1.0MΩ	Digi-Key	RS125-ND
540 pcs, 30 values, 10Ω to 10MΩ	Jameco	103166
500 pcs, 64 values, 1.0Ω to 10MΩ	RadioShack	271-312

See *Resistor Color Codes* at `http://www.allaboutcircuits.com/vol_5/chpt_2/1.html` to learn how to read the color bands on carbon film resistors.

Qty	Value (kΩ)	Color Code
4	1.0	Brown - Black - Red
1	10	Brown - Black - Orange
2	15	Brown - Green - Orange

Capacitors

Qty	Value (μF)	Type
3	0.01	Ceramic
4	0.1	Ceramic

Inductors

Qty	Description	Digi-Key #
1	33-mH inductor (Murata Power Solutions 22R336C)	811-1294-ND

Active Devices and Integrated Circuits

NOTE: Texas Instruments offers free samples. Go to `http://www.ti.com` and click "Sample & Buy" to get started.

Qty	Description	Digi-Key #
1	TL072CP dual op amp	296-1775-5-ND

Jumper Wire Kit

Circuit Specialists part number WK-1 (350 pieces, pre-formed, 22 AWG), `http://www.circuitspecialists.com/prod.itml/icOid/6920`

Circuit Specialists part number MJW-70B (140 pieces, pre-formed, 22 AWG), `http://www.circuitspecialists.com/prod.itml/icOid/7590`

Appendix B

TL072 Operational Amplifier

The Texas Instruments TL072 dual operational amplifier ("op amp") provides two op amp devices in a single 8-pin package. For more details see the datasheet available at `http://www.ti.com`; enter "tl072" in the "Search by Part Number" field.

Figure B.1 on the following page shows the pinout diagram for the TL072. Note the requirement for a dual power supply; the NI NI ELVIS II ± 15 V supply serves this purpose. Also note that the op amp device itself does not have a ground terminal. Instead the NI ELVIS II GROUND (analog ground) establishes the ground reference.

NI Multisim provides a circuit model for the TL072: place the `TL072CP` device.

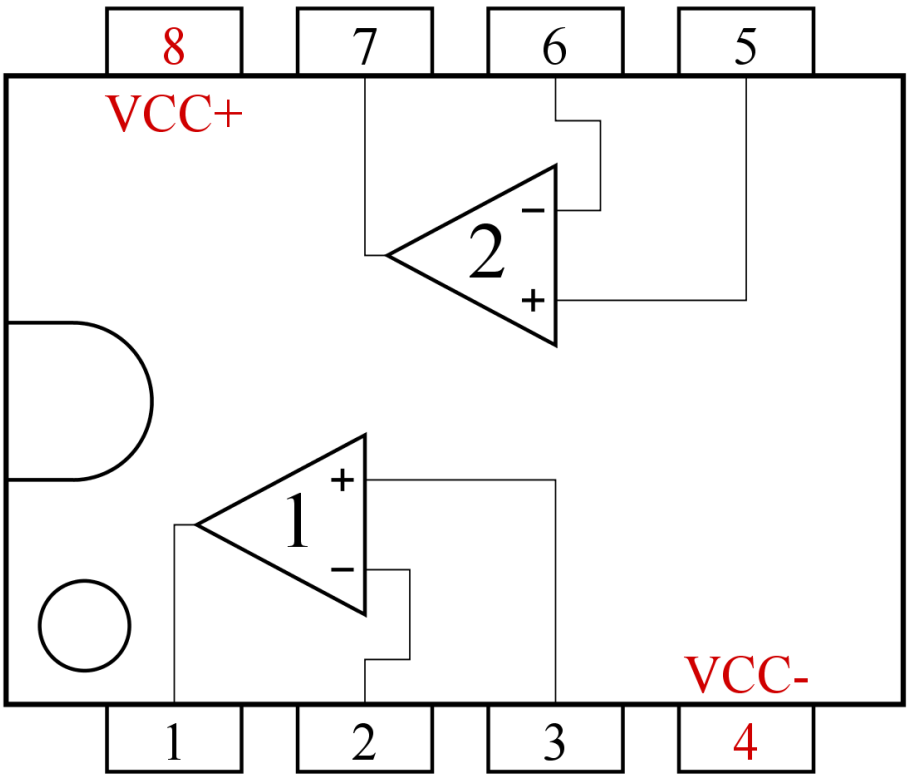


Figure B.1: Texas Instruments TL072 dual op amp pinout diagram (top view). The plastic package uses either a U-shaped cutout to indicate the left side or an indented circle to indicate pin 1. NOTE: In this book the positive supply "VCC+" is called V_{CC} and the negative supply "VCC-" is called V_{EE}.

Appendix C

Video Tutorial Links

Signals and Systems Concepts

- Work through a complete graphical convolution example:
  http://youtu.be/zoRJZDiPGds
- Understand discrete-time convolution with an example:
  http://youtu.be/yyTu0SXeW1M
- Study the Butterworth filter magnitude response:
  http://youtu.be/QJdNVdL4iBQ
- Apply the 2-D FFT as an image filter:
  http://youtu.be/X53SrMnu91A
- Create a 2-D Butterworth image filter:
  http://youtu.be/qpwUHt-DEbs

NI LabVIEW Video Tutorials

General-purpose techniques:

- Configure "look and feel" of LabVIEW 2012:
  http://youtu.be/M6gDpso6ysI
- Create a subVI:
  http://youtu.be/C2zYeC_RpM8
- Create a load-process-save design pattern:
  http://youtu.be/IDVHarNkJSU

- Display mouse coordinates:
  http://youtu.be/nVIOnz2WYyQ
- Disable a portion of block diagram code:
  http://youtu.be/unuRYWUYQy4
- Use global variables to reduce wiring clutter:
  http://youtu.be/PquslTj69f4
- Create a menu control:
  http://youtu.be/tImN4C03W1k
- Create an interactive front panel with event structure:
  http://youtu.be/eGlvOiqYVxg
- Execute an event structure one time on run:
  http://youtu.be/UFEvmmggqLQ

Digital filtering:

- Use the "Convolution" subVI:
  http://youtu.be/Fsf88Rmimz4
- Create a bandpass filter subVI:
  http://youtu.be/gBVluWy_UGs
- Implement H(z) as a digital filter:
  http://youtu.be/IsbItFzzOdA
- Use the "Plot Frequency Response" subVI:
  http://youtu.be/IsjXautfUXs
- Use the "Plot Pole-Zero" subVI:
  http://youtu.be/S8WmBxIlnEc

Audio:

- Work with .wav audio files:
  http://youtu.be/ChSVf9lZ44k
- View an audio signal and its spectrum:
  http://youtu.be/R3h4_uR9Dcc

Images:

- Basic image handling techniques:
  http://youtu.be/Z_81P1d_rCg
- Auto-resize an image picture indicator:
  http://youtu.be/pw7-smkZh_U

Fast Fourier Transform (FFT):

- Compute magnitude spectrum with the "FFT" subVI:
  http://youtu.be/9SL5_jY87IA
- Demonstrate basic properties of the FFT:
  http://youtu.be/UGUceuhSuUE
- Study the basic concepts of the FFT:
  http://youtu.be/z7X6jgFnB6Y

NI LabVIEW MathScript Video Tutorials

- Get started with the MathScript Window:
  http://youtu.be/Tvi1oY0lV2g
- Plot two functions of time:
  http://youtu.be/XQlAai1-YVc
- Take cursor measurements on a plot:
  http://youtu.be/bgK1p5O6OXc
- Work with images in the MathScript Window:
  http://youtu.be/oM_KSYIDrE4

NI Multisim & NI ELVISmx Video Tutorials

- Compare simulated and physical DMM measurements:
  http://youtu.be/MZiZ_C-ngkY

NI Multisim Video Tutorials

Place components:

- Find commonly-used circuit components:
  http://youtu.be/G6ZJ8C0ja9Q
- Find components by name:
  http://youtu.be/5wlFweh4n-c
- Place and operate linear potentiometer:
  http://youtu.be/oazwGLzWvhs
- Place dual op amp second device:
  http://youtu.be/-QDFEf-KdEw

Sources:

- Function generator:
  http://youtu.be/CeOl6EzD-_c
- AC (sinusoidal) voltage source:
  http://youtu.be/CXbuz7MVLSs
- ABM (Analog Behavioral Model) voltage source:
  http://youtu.be/8pPynWRwhO4
- Pulse voltage source:
  http://youtu.be/RdgxVfr28C8
- Piecewise linear (PWL) voltage source:
  http://youtu.be/YYU5WuyebD0
- VDD and VSS power supply voltages:
  http://youtu.be/XrPVLgYsDdY
- VCC and VEE power supply voltages:
  http://youtu.be/XkZTwKD-WjE

Measure DC power:

- Use a Parameter Sweep analysis to plot resistor power as a function of resistance:
  http://youtu.be/3k2g9Penuag

Measure frequency response:

- Measure frequency response with AC Analysis:
  http://youtu.be/tgCPDBtRcso

Net names:

- Display and change net names:
  http://youtu.be/0iZ-ph9pJjE

Grapher View and oscilloscope cursor measurements:

- Find the maximum value of a trace in Grapher View:
  http://youtu.be/MzYK60mfh2Y
- Set cursor to a specific value:
  http://youtu.be/48sQja58I10

Grapher View environment:

- Add data label to Grapher View plot:
  http://youtu.be/uibNqFlod_Y
- Plot mathematical expression:
  http://youtu.be/qYVf_rkTF-o
- Adjust axis limits and tick marks on a Grapher View plot:
  http://youtu.be/nxHHMpJd5Ow
- Plot second variable on its own axis in Grapher View:
  http://youtu.be/rULFKRTphcI
- Export simulation data to an Excel spreadsheet:
  http://youtu.be/s6ezjb8Xrhc

Oscilloscope:

- Basic operation of the two-channel oscilloscope:
  http://youtu.be/qnRK6QyqjvQ
- Waveform cursor measurements with the two-channel oscilloscope:
  http://youtu.be/snBRFq1Y1q4

- Distinguish oscilloscope traces by color:
  http://youtu.be/bICbjggcTiQ
- Stabilize the oscilloscope display with edge triggering:
  http://youtu.be/d69zYYSEG7E
- Basic operation of the four-channel oscilloscope:
  http://youtu.be/iUqs_c1Bc4Y

Transient response:

- Plot time-domain circuit response with Transient Analysis:
  http://youtu.be/waKnad_EXkc

Combined Multisim / ELVISmx measurements:

- Combine Multisim simulation and myDAQ measurements in the same instrument – Bode Analyzer:
  http://youtu.be/3UmTmUj4h1g

NI ELVISmx Video Tutorials

See *Electrical Circuits with NI myDAQ* for more video tutorials and projects:
http://decibel.ni.com/content/docs/DOC-12654

NI ELVISmx Instruments:

- DMM ohmmeter:
  http://decibel.ni.com/content/docs/DOC-12938
- DMM voltmeter:
  http://decibel.ni.com/content/docs/DOC-12937
- DMM ammeter:
  http://decibel.ni.com/content/docs/DOC-12939
- Function Generator (FGEN):
  http://decibel.ni.com/content/docs/DOC-12940
- Arbitrary Waveform Generator (ARB):
  http://decibel.ni.com/content/docs/DOC-12941

- Oscilloscope:
  http://decibel.ni.com/content/docs/DOC-12942
- Bode Analyzer:
  http://decibel.ni.com/content/docs/DOC-12943
- Digital Reader (DigIn):
  http://decibel.ni.com/content/docs/DOC-12944
- Digital Writer (DigOut):
  http://decibel.ni.com/content/docs/DOC-12945

Measurement techniques:

- Increase current drive of analog output (AO) channels with an op amp voltage follower:
  http://decibel.ni.com/content/docs/DOC-12665