Appendix D

LabVIEW Tutorial

D.1 Introduction

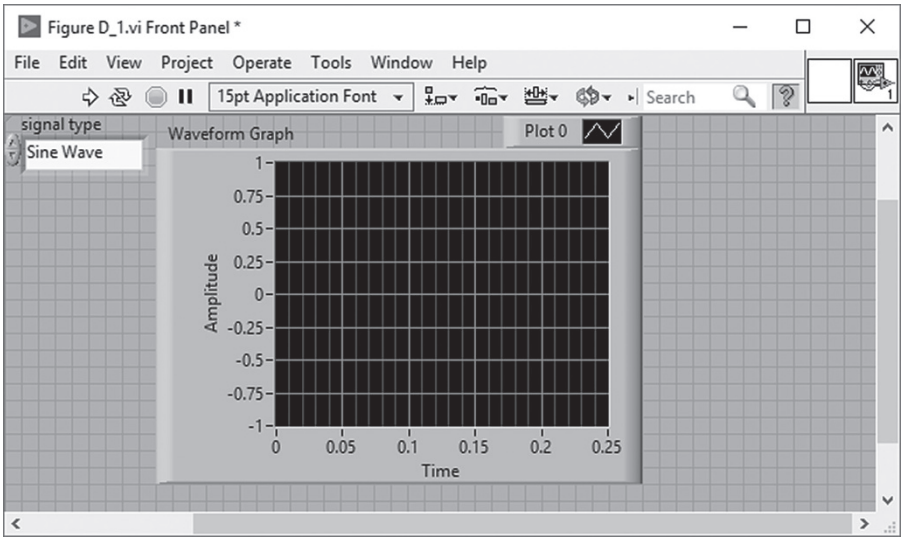
LabVIEW is a programming environment that is presented here as an alternative to MATLAB. Although not necessary, the reader is encouraged to become acquainted with MATLAB before proceeding, since familiarity with MATLAB can enhance the understanding of the relationship between textual (MATLAB) and graphical (LabVIEW) programming languages and extend the functionality of LabVIEW. In this tutorial, we will show how to use LabVIEW to **(1)** analyze and design control systems, and **(2)** simulate control systems. This appendix was developed using LabVIEW 2017.

LabVIEW is a graphical programming environment that produces virtual instruments (VI's). A VI is a pictorial reproduction of a hardware instrument on your computer screen, such as an oscilloscope or waveform generator. The VI can consist of various controls and indicators, which become inputs and outputs, respectively, to your program. Underlying each control and indicator is an associated block of code that defines its operation. The LabVIEW model thus consists of two windows: **(1) Front Panel**, which is a replica of the hardware front panel showing the controls and indicators, and **(2) Block Diagram**, which contains the underlying code for the controls and indicators on the **Front Panel**.

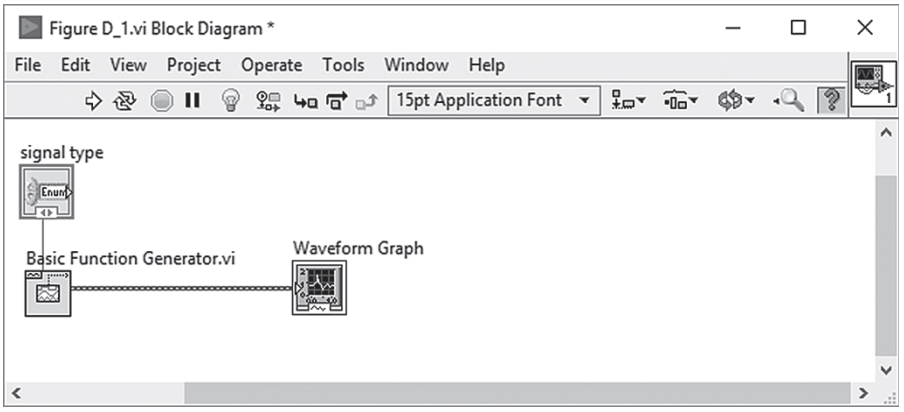
Associated with the **Front Panel** window is a **Controls** palette window containing numerous icons representing controls and indicators. The icons can be dragged onto a **Front Panel** window to create that control or indicator. Simultaneously, the associated code block is formed on the **Block Diagram** window.

Alternately, the block diagram can be formed first, and then the front panel is created from the block diagram. Associated with the **Block Diagram** window is a **Functions** palette window containing numerous icons representing a wide range of functions. Icons can be dragged onto a **Block Diagram** window to create that code block.

For example, Figure D.1(*a*) is the front panel of a signal generator. The generator consists of a control to select the signal type and a waveform graph that shows the output waveform. Figure D.1(*b*) shows the underlying code, which is contained in the code blocks. Here, the signal type selector is a control, while the waveform graph is an indicator. Later we will show how to make connections to other VI's. The palette windows for the front panel and block diagram are shown respectively in Figures D.1(*c*) and (*d*). VI's in this appendix can be found in the Control Systems Engineering Toolbox.



(*a*)



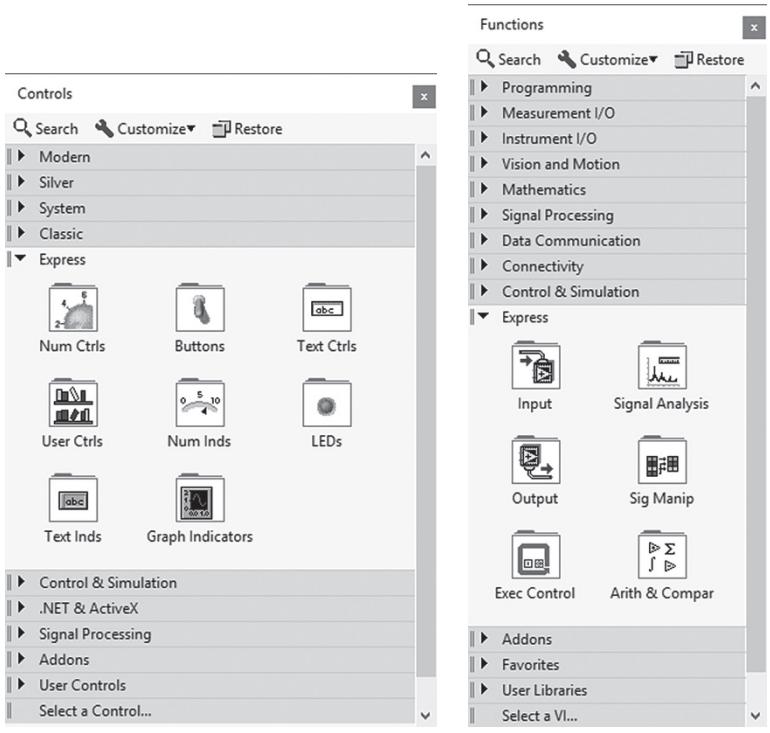
(*b*)

FIGURE D.1 A LabVIEW function generator VI: **a. Front Panel** window; **b. Block Diagram** window; (*figure continues*)

D.2 Control Systems Analysis, Design, and Simulation

LabVIEW can be used as an alternative to or in conjunction with MATLAB to analyze, design, simulate, build, and deploy control systems. In addition to LabVIEW, you will need the LabVIEW Control Design and Simulation Module. Finally, as an option that will be explained later, you may want to install the MathScript RT Module.

Analysis and design can be thought of as similar to writing MATLAB code, while simulation can be thought of as similar to Simulink. In LabVIEW, analysis and design, as opposed to simulation, are handled from different subpalettes of the **Functions** window's **Control & Simulation** palette. See Figure D.1(*d*). Analysis and design, and simulation will typically begin with the **Block Diagram** window, where icons representing code blocks will be interconnected. Parameters used by the code can be conveniently selected, changed, and passed to the code through VI controls on the **Front Panel** window created from the code icons. Any results, such as time response, can be displayed through VI indicators on the **Front Panel** window created from the code icons.



(*c*) (*d*)

FIGURE D.1 (*continued*) **c. Controls** palette; **d. Functions** palette

D.3 Using LabVIEW

The following steps start you on your way to using LabVIEW for control systems analysis, design, and simulation. These steps will be illustrated in the examples that follow.

1. **Start LabVIEW** LabVIEW starts with the window shown in Figure D.2, where you can select a **New VI** or **Open** an existing VI from the **File** menu. Alternatively, existing

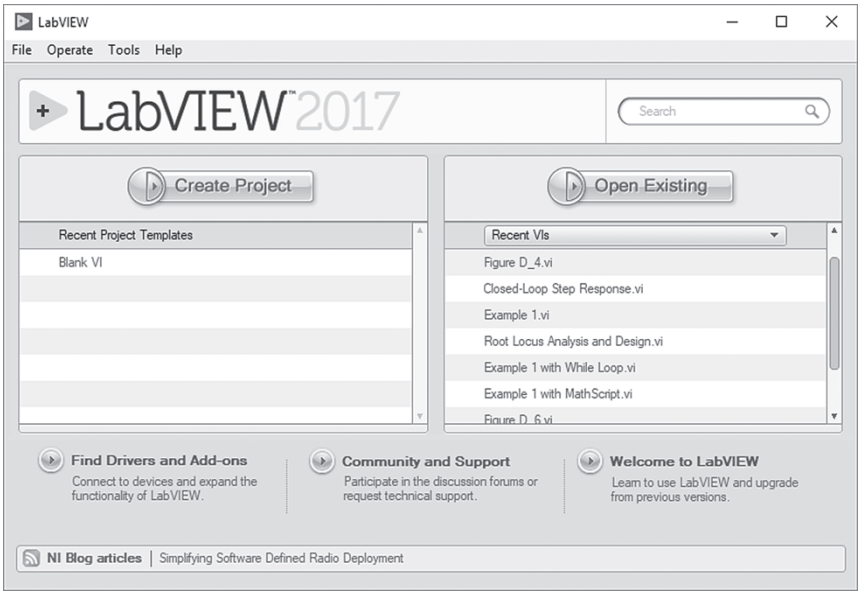


FIGURE D.2 **LabVIEW** window

VI's can be opened from the **Open Existing** table on the right. Selecting a new or existing VI brings up the **Front Panel** and **Block Diagram** windows shown in Figure D.1. If necessary, a window can be opened from the **Window** tab on the menu bar of the **Front Panel** and **Block Diagram**.

Right-click the **Block Diagram** window to bring up the **Functions** palette and click the thumb tack in the upper left-hand corner to dock the window. Repeat for the **Front Panel** window to access the **Controls** palette.

2. **Select blocks** Make the **Block Diagram** window active, or access it from **Window** on the menu bar. Right-click the **Block Diagram** window or use the **View** menu to bring up the **Functions** palette. Expand the palette window by clicking the double-up arrows at the bottom. At the top of the palette window click **Customize**, and select **View This Palette As/Category (Icons and Text)** to add a text description below each icon. For control systems analysis, design, and simulation, expand **Control & Simulation** in the **Functions** palette by clicking the arrow to the left of this category.

   If you are performing a simulation, click the subpalette **Simulation**. If you are performing control system analysis or design, click the subpalette **Control Design**. A small tab on the upper-left above the subpalette indicates additional underlying palettes or blocks.

   If the name of the icon is incomplete, resting the mouse over the icon will bring up its complete identification. To obtain detailed help about an icon, right-click the icon and select **Help**.

3. **Move blocks to the block diagram window** To move the icon to the **Block Diagram**, left-click the mouse to attach the icon (some icons take a little time to complete this operation). When the pointer turns into a hand, click the spot on the **Block Diagram** where you want to place the icon.

4. **Obtain information about the block** You will now want to obtain information about how to interconnect the block to other blocks and pass parameters to the block as well as other characteristics about the block. Select the yellow question mark at the right of the **Block Diagram** toolbar to turn on the **Context Help window**. This window will provide help about a particular icon if you rest your mouse over that icon. Additional help is available under the **Help** menu on the **Block Diagram** menu bar. Finally, right-click the icon to bring up a menu with additional choices, such as **Properties**, if any. In particular, you will use this menu to create the block's front panel's controls and indicators. This front panel will be your interface with the block to choose parameters and see responses.

5. **Interconnect and label blocks** Once blocks are placed on the **Block Diagram** they can be moved about by clicking on them or dragging your mouse across several of them to establish a selection pattern. After the selection pattern has been established, depress the mouse left button and drag to a new location. To delete a block, select the block and press the **Backspace** button on your keyboard.

   The context help for the block includes a description of the block's terminals. Let your mouse rest on a terminal until the mouse pointer turns into a spool of wire. Click the terminal and then move the mouse to the next icon's terminal where you want to make the connection. Click the destination terminal to complete the wiring. Notice that the terminal in the **Context Help** window blinks when your mouse resides above that terminal on the block, ensuring that you are on the correct terminal. If you make an error in wiring, click on the wire and press the **Backspace** button on your keyboard or right-click the wire and select **Delete Wire Branch**.

   Block labels can be displayed or hidden. Right-click on the block to bring up the pop-up menu and check or uncheck **Visible Items/Label** to display or hide, respectively, the label. Double-clicking on the label above some blocks will allow you to select and change the text. One click of the mouse on the label will place a selection pattern around the label and allow you to hold down the left key of the mouse and move the label to a different location.

6. **Create the interface to your block** You will now want to create the interface to your block in order to control or select functions, specify parameters, or view responses. This interface will be accessed via the **Front Panel** window. Right-click a terminal on a block for which you want to create an interface. On the pop-up menu, choose **Create/Control** to be able to interact with the block or **Create/Indicator** to view a response or setting.
7. **Set the controls** Switch to the **Front Panel** window and set your controls. For example, enter parameter values, select functions, etc. If you want to change values and at some future time return to the current values, click on **Edit** on the **Block Diagram** menu bar and select **Make Current Values Default**. To return to the default values in the future, click on **Edit** on the **Block Diagram** menu bar and select **Reinitialize Values to Default**.
8. **Run the program** Click on the arrow at the left of the toolbar on either the **Block Diagram** or **Front Panel** window to run the program. The program can be run continuously by clicking the curved arrows button on the toolbar second from the left. Continuously running your program permits changing functions and parameter values during execution.

   In order to identify the buttons, let your mouse rest on a button to bring up a context menu. Stop your simulation by pressing the red-dot button, third from the left. If you are performing control systems analysis and design, another way to continuously run the program is to place a **While Loop** around your block diagram. The loop is available in the **Functions** palette at **Express/Execution Control/While Loop**. This loop also places a **Stop** button on the **Front Panel**. The program executes until you press the stop button. In lieu of the **Stop** button, any true/false Boolean can be wired to the condition block (red dot) created inside the **While Loop**.

   If you are performing simulation, you can use a **Simulation Loop** available in the **Functions** palette at **Control & Simulation/Simulation/Control & Simulation Loop**. Place the **Control & Simulation Loop** around your simulation block diagram by dragging the mouse. Right-click on the **Control & Simulation Loop** outline and choose **Configure Simulation Parameters**... to determine the parameters for executing the simulation. The **Front Panel** indicators and controls are also configurable. Right-click on the indicator or control and select **Properties**.

D.4 Analysis and Design Examples

In this section, we will present some examples showing the use of LabVIEW for the analysis and design of control systems. In the next section, examples of the use of LabVIEW for simulation will be presented.

Analysis and design examples use icons selected from the **Control Design** subpalette under the **Control & Simulation** palette. In the next section showing examples of simulation, we will use icons taken from the **Simulation** subpalette under the **Control Design and Simulation** palette.

Example D.1

Open-Loop Step Response

Analysis and design usually begins by selecting icons from the **Control Design** subpalette and dragging them to the **Block Diagram** window. The icons represent blocks of code and the cascading of code blocks can be thought of as a sequence of lines of code. Thus, an advantage of LabVIEW over MATLAB is that the programmer does not need to memorize coding language. For example, consider the MATLAB code shown in TryIt D.1 that produces the step response of $G(s) = 100/(s^2 + 2s + 100)$.

TryIt D.1

```
numg=100;
deng=[1 2 100];
'G(s)'
G=tf(numg,deng)
step(G);
title('Angular Velocity')
```

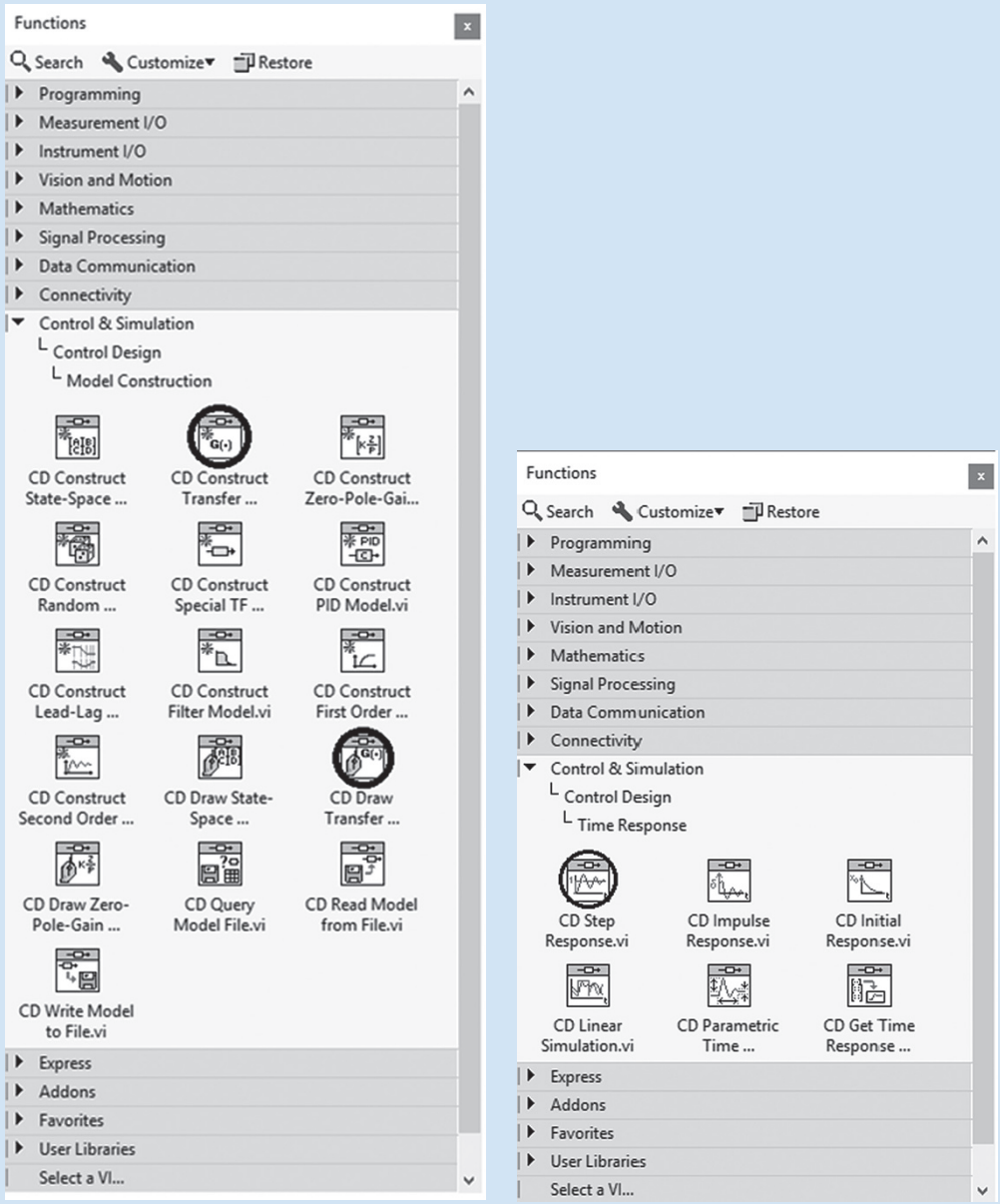


FIGURE D.3 Selecting **a. CD Construct**… and **CD Draw**…; **b. CD Step Response**

This step response can be produced in LabVIEW without knowing any coding language. We now demonstrate by following each step of Section D.3:

1. **Start LabVIEW** Start LabVIEW and select **New VI** from the **File** menu shown in Figure D.2.
2. **Select blocks** From the **Functions** palette, select the blocks shown in Figure D.3(*a*) and (*b*).
3. **Move/blocks to the Block Diagram window** Drag your icons one at a time to the **Block Diagram** window, Figure D.4.
4. **Obtain information about the block** Right-click each of the blocks and be sure the first two items under **Visible Items** are checked. Look at the **CD Construct Transfer Function Model.vi.** A **Polymorphic VI Selector** is shown at the bottom of the icon.

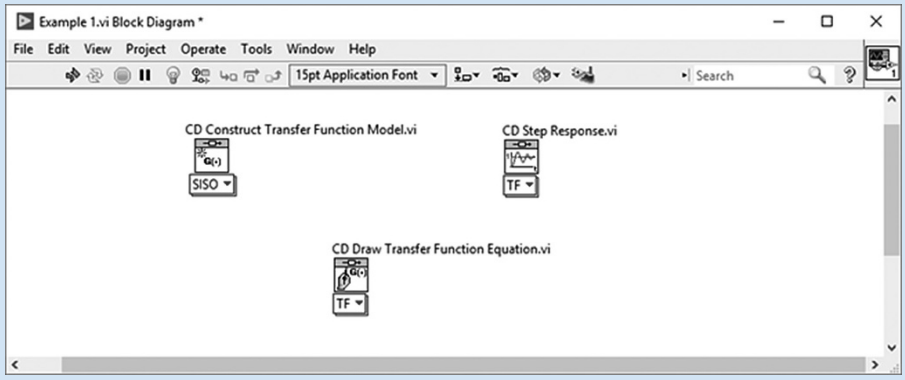


FIGURE D.4 **Block Diagram** window

Click the selector to bring up the menu. Select Single-Input-Single-Output **(SISO)**. This block effectively creates the transfer function shown in the first four steps of the MATLAB code in TryIt D.1.

Repeat for the **CD Draw Transfer Function Equation.vi** and select Transfer Function **(TF)** from the **Polymorphic VI Selector**. This block will write the transfer function symbolically in the display. Your selection from the polymorphic vi selector should match the format of the transfer function created by the **CD Construct Transfer Function Model.vi**.

Repeat for the **CD Step Response.vi.** and select **TF** from the **Polymorphic VI Selector**. This block will collect the data for the step response and permit plotting the data. This block effectively creates the last two commands of the MATLAB code shown in TryIt D.1.

5. **Interconnect and label blocks** You should now have the **Block Diagram** window shown in Figure D.4. Interconnect the code blocks. Click on the question mark on the right side of the toolbar to bring up the context menu. As your mouse passes above an icon, its context menu appears, showing the terminals. See Figure D.5. Interconnect the terminals by letting the mouse rest on a terminal until it becomes a spool of wire.

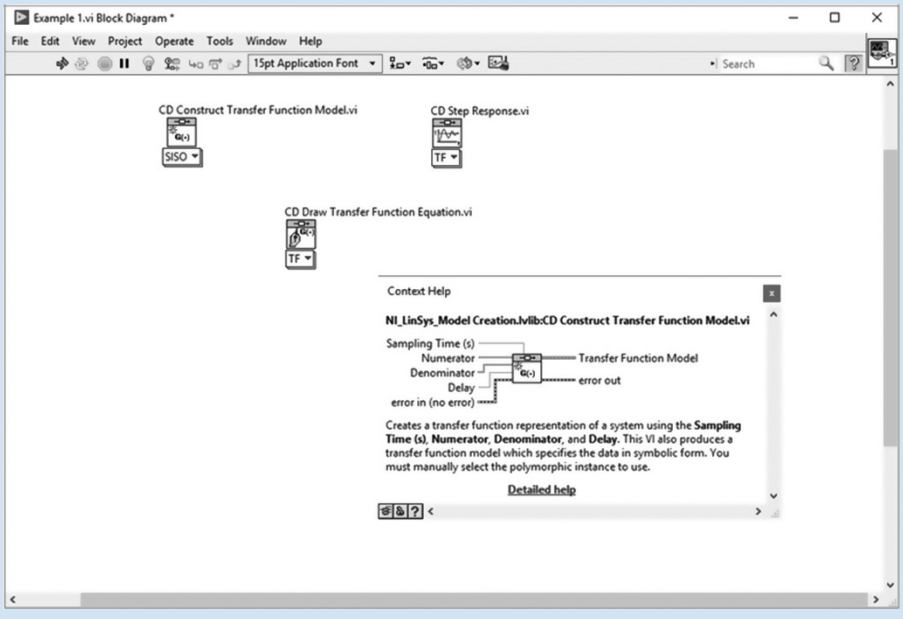


FIGURE D.5 **Context Help** for **CD Construct Transfer Function Model.vi**

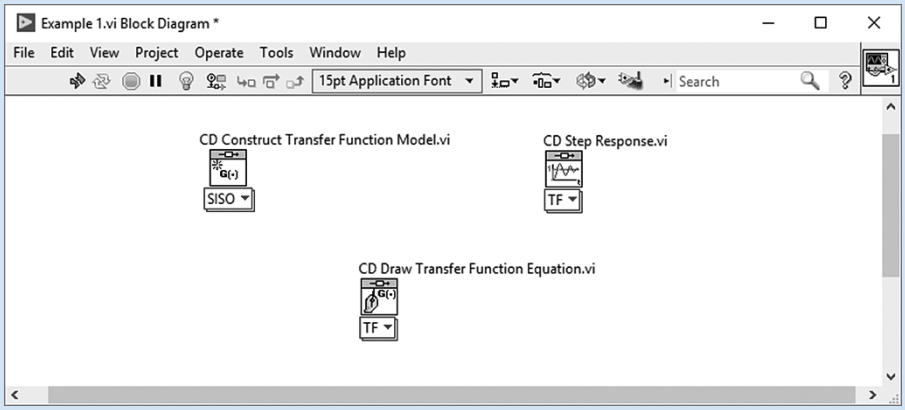


FIGURE D.6 Interconnected blocks

Click on the terminal and then click on the destination terminal. The two terminals will appear as wired together. Continue wiring terminals until you have the **Block Diagram** window shown in Figure D.6. Mid-wire connections as shown can be made by letting your mouse rest at the connection point until it becomes a spool of wire.

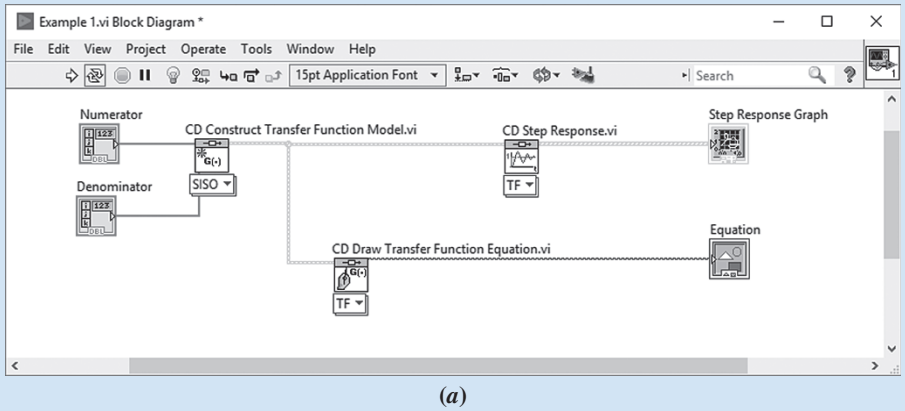
6. Create the interface to your block You will now want to create the interface to specify parameters and view responses. This step will create the interface that will be accessed on the **Front Panel** window. The interfaces we will create are:

CD Construct Transfer Function Model.vi input parameter controls. Right-click on the numerator terminal shown in Figure D.5 and select **Create/Control**. Repeat for the denominator.

CD Step Response.vi response plot indicator. Right-click on the **Step Response Graph** terminal and select **Create/Indicator**.

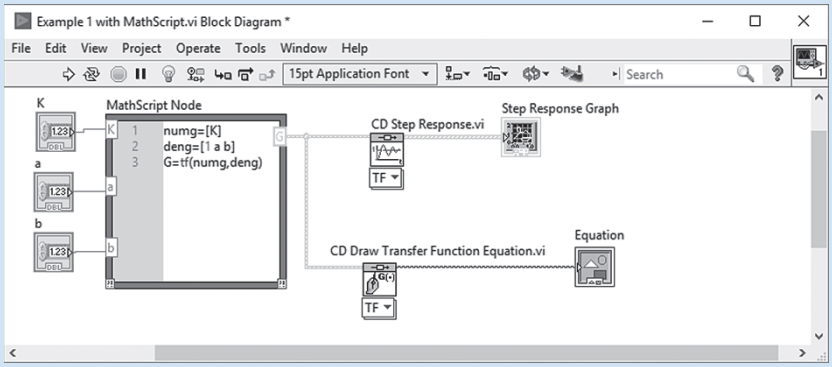
CD Draw Transfer Function Equation.vi symbolic transfer function indicator. Right-click on the **Equation** terminal and select **Create/Indicator**. Your **Block Diagram** should now look similar to Figure D.7(*a*).

As an option, you can create transfer functions using a **MathScript** block if the MathScript RT Module is installed. This option is generally compatible with MATLAB's M-file code statements for creating your transfer function. Interfaces are then created to pass parameters to and from the M-file code. You should be familiar with MATLAB to use this option. The **MathScript** block is found in **Functions** in the



(*a*)

FIGURE D.7 **Block Diagram** window: **a.** with **Control Design** blocks and interfaces; (*figure continues*)



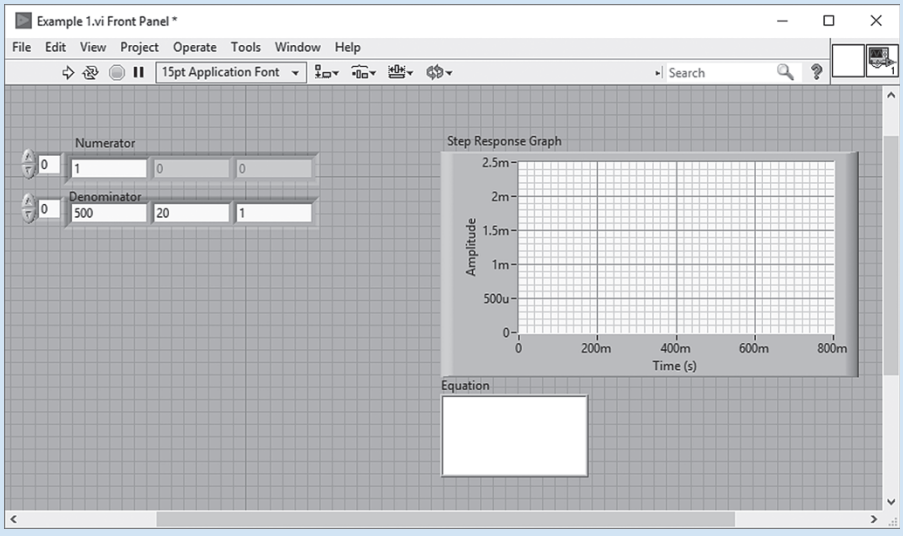
(*b*)

FIGURE D.7 (*continued*) **b.** with **MathScript** block

Programming/Structures/MathScript palette. You create M-file code inside the **MathScript** block. Inputs, outputs, and controls are created as follows. Right-click on the left side of the **MathScript Node** and select **Add Input**. Name the input **K**. Right-click your terminal **K** and select **Create/Control**. A control is formed on both the **Block Diagram** and **Front Panel**. Repeat the same process to create inputs and controls for parameters **a** and **b**. Now create the output to the **MathScript Node**. Right-click the right-hand side of the **MathScript Node** and select **Add Output/G**. After wiring your inputs and outputs, your Block Diagram will be that shown in Figure D.7(*b*).

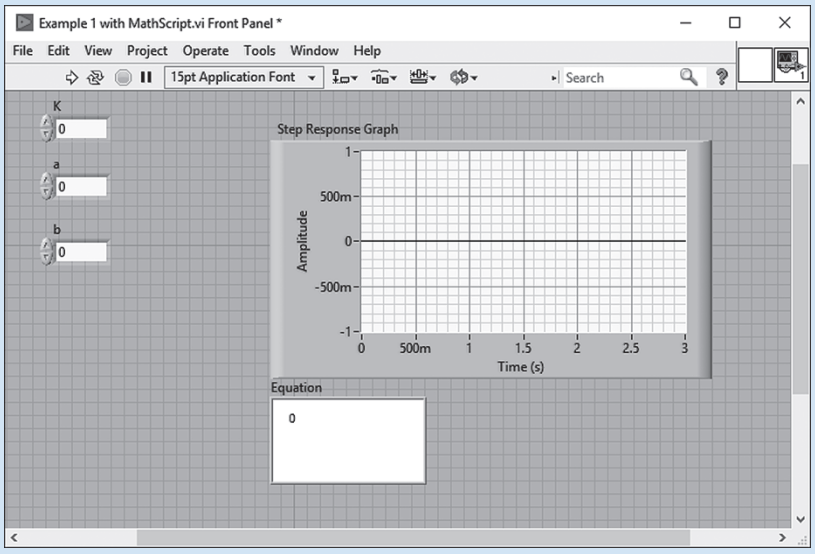
On the **Block Diagram** window menu bar, select **Window/Show Front Panel**. You will see the **Front Panel** shown in Figure D.8 created by your interfaces. You can double-click the labels above your interfaces either in the **Front Panel** window or the **Block Diagram** window to change the label to be more descriptive of your project.

7. Set the controls Using the **Front Panel** window, enter polynomial coefficients for the numerator and denominator in ascending order—lowest to highest. The selector to the left of the numerator and denominator shows the power of *s* for the left-most coefficient. Increasing the counter allows entry of higher-order coefficients not visible originally. To make all coefficients of a polynomial visible, let the mouse move on the right-hand edge of the polynomial indicator until the pointer becomes a double arrow



(*a*)

FIGURE D.8 **Front Panel: a.** for **Block Diagram** shown in Figure D.7(*a*); (*figure continues*)



(*b*)

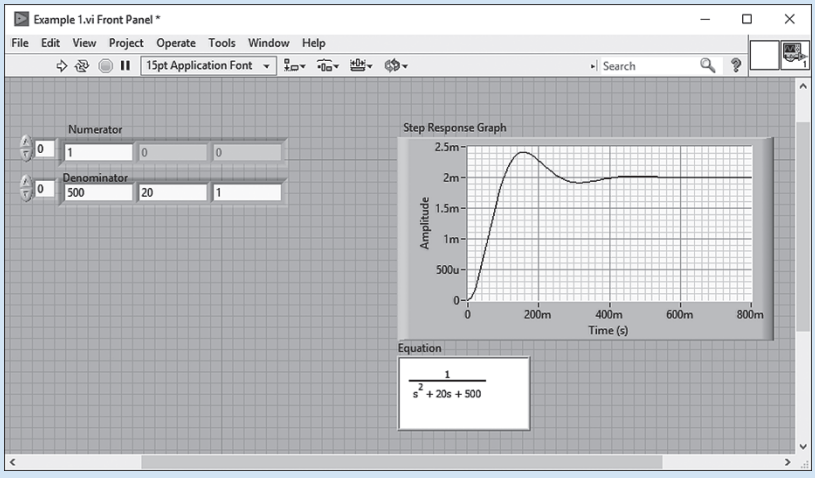
FIGURE D.8 (*continued*) **b.** for **Block Diagram** shown in Figure D.7(*b*)

and blue dots appear at the left and right edges of the entire polynomial indicator. You can then drag the right blue dot to expose more cells.

Familiarize yourself with the choices on the menu bar as well as those on the pop-up menus created when you right-click on any indicator or control. For example, under the **Edit** menu, among other choices, you can **Make Current Values Default** or **Reinitialize Values to Default**. Right-clicking the indicators or controls brings up a menu from which, among other choices, **Properties** can be selected to configure the indicator or control as desired.

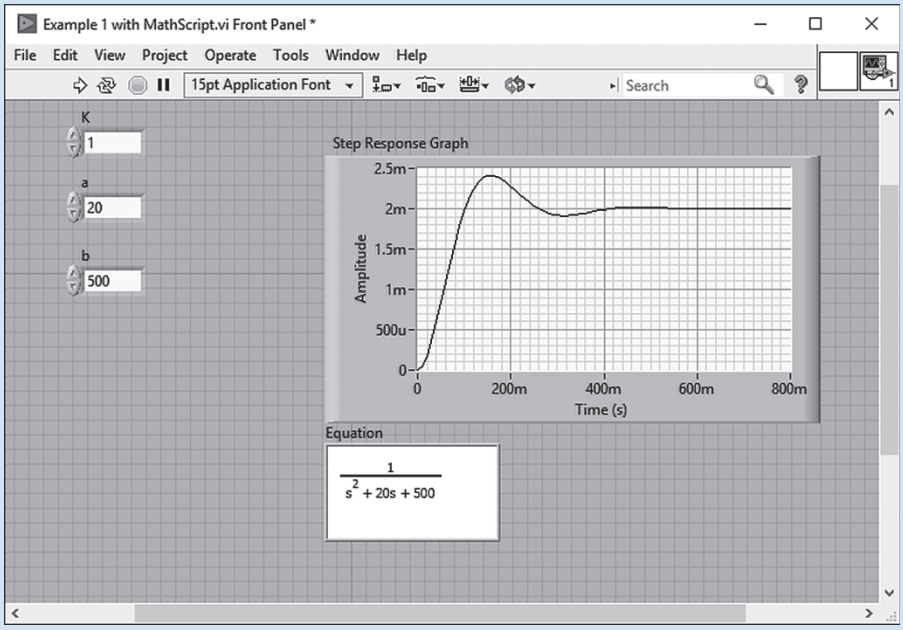
8. Run the program Figure D.9 shows Example D.1 after execution. The figure shows the values entered, the equation, and the step response. Execution was initiated by clicking the arrow at the left of the toolbar.

The program can run continuously by clicking the curved arrows on the toolbar. Now, change values; hit the **Enter** key and see the results immediately displayed. Stop the program execution by clicking on the red hexagon on the toolbar. Another way of



(*a*)

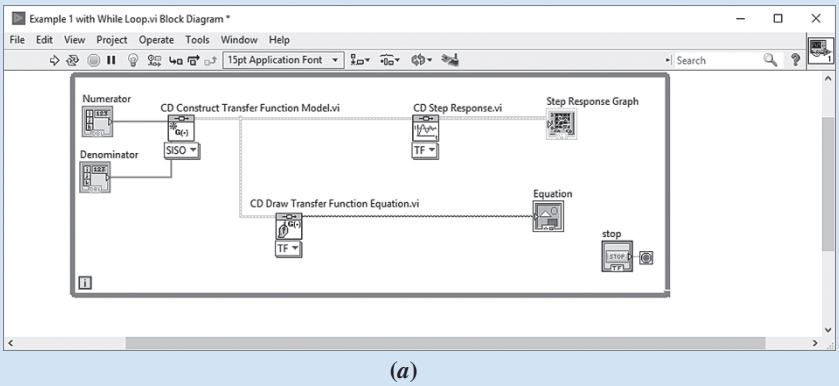
FIGURE D.9 **Front Panel** after execution: **a.** for block diagram in Figure D.7(*a*); (*figure continues*)



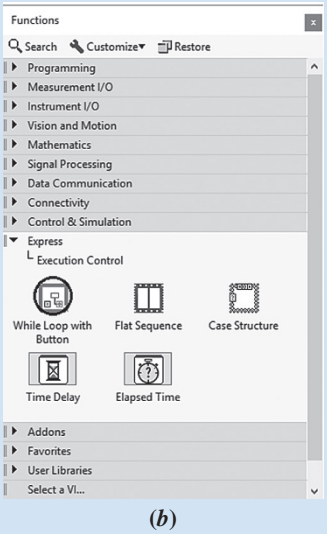
(*b*)

FIGURE D.9 (*continued*) **b.** for block diagram in Figure D.7(*b*)

continuously running the program is to place a **While Loop** around the block diagram as shown in Figure D.10(*a*). The loop is accessed from **Functions/Express/Execution Control** as shown in Figure D.10(*b*). After selecting the **While Loop**, drag the cursor across the block diagram to create the continuous loop. A **stop** button will appear on the block diagram as well as on the **Front Panel**. At the lower right is a **Conditional Terminal** icon, which can be used to control the **While Loop**. The reader should consult the on-line documentation for further information.



(*a*)



(*b*)

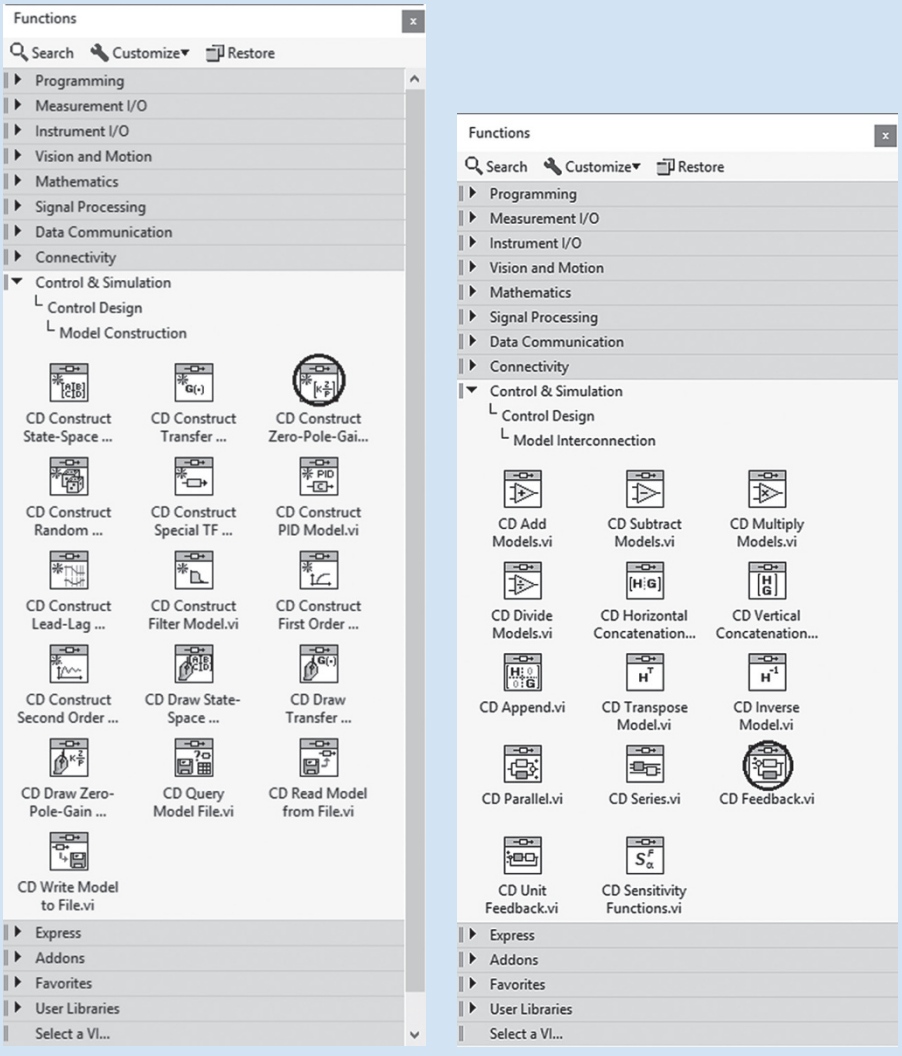
FIGURE D.10 **a.** Block diagram with **While Loop**; **b. Functions** palette showing **While Loop** location

Example D.2

Closed-Loop Step Response

In this example, we show how to display the step response of a unity-feedback system. For variety, we represent the open-loop system as a ratio of zeros over poles with a multiplying gain, analogous to MATLAB's **zpk** function. In the previous example, we represented the system as a ratio of polynomials, analogous to MATLAB's **tf** function.

1. **Select blocks** The zero-pole-gain transfer function is obtained from the **Functions** palette as shown in Figure D.11(*a*). We place this transfer function in the forward path



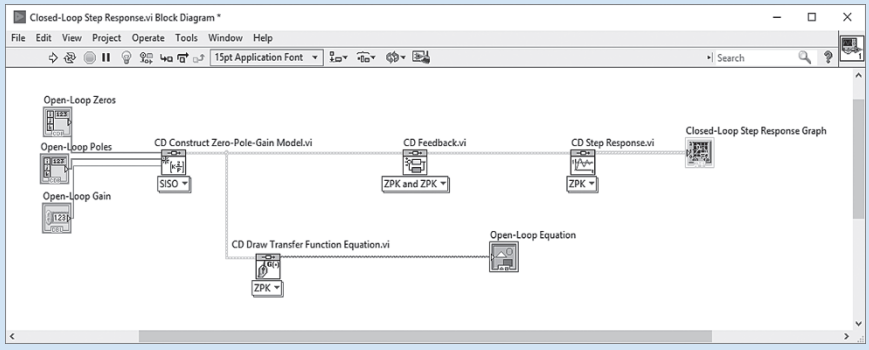
(*a*) (*b*)

FIGURE D.11 **a.** Obtaining zero-pole-gain transfer function from the **Functions** palette; **b.** obtaining **Feedback** interconnection from **Functions** palette

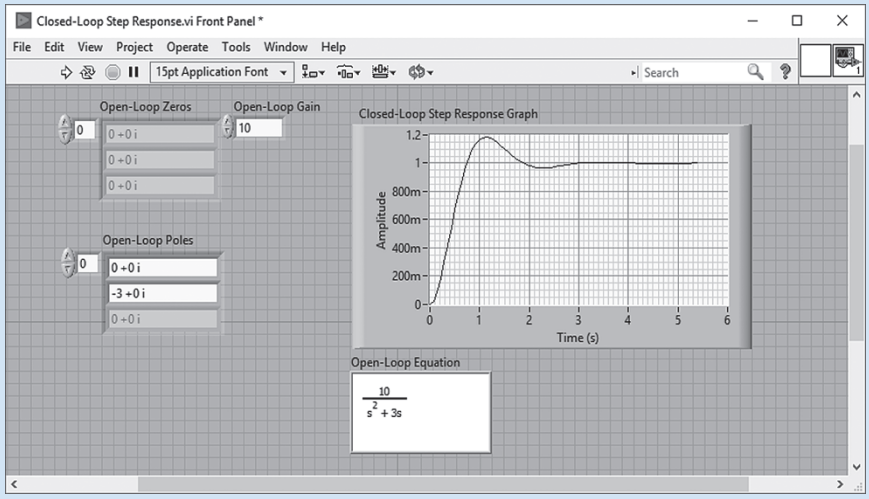
of a unity-feedback system by following its block with a **Feedback** block obtained from the **Functions** palette as shown in Figure D.11(*b*). If the **Model 2** input to the **Feedback** block is left unconnected, then a unity-feedback interconnection is assumed. Other options for interconnection, such as parallel and series, are shown on the palette of Figure D.11(*b*).

2. **Interconnect and label blocks** Producing the closed-loop step response is similar to Example D.1, except the step-response blocks are placed at the output of the **Feedback** block. The equation writer is wired to the system output as in Example D.1. All data types must be compatible and are shown selected with the pull-down menu at the base of the blocks. If you select **Automatic** in the pull-down menu, LabVIEW will select the correct form for you as you connect the blocks.

   The final **Block Diagram** and **Front Panel** for this example are shown in Figure D.12 (*a*) and (*b*), respectively. Notice that you enter open-loop poles, zeros, and gain on the **Front Panel** in place of polynomial open-loop numerator and denominator coefficients.



(*a*)



(*b*)

FIGURE D.12 **a. Block Diagram** for Example D.2; **b. Front Panel** for Example D.2

Example D.3

Root Locus Analysis and Design

We can obtain root locus plots by adding the **Root Locus** block obtained from the **Functions** palette as shown in Figure D.13. The **Root Locus** block is connected to the output of the open-loop system and a **Root Locus Graph** indicator is formed at the output of the **Root Locus** block. The resultant **Block Diagram** and **Front Panel** are shown in Figure D.14(*a*) and (*b*) respectively.

Figure D.13 shows other characteristic blocks that can be added. For example, closed-loop poles and zeros, as well as damping ratio and natural frequency, can be displayed.

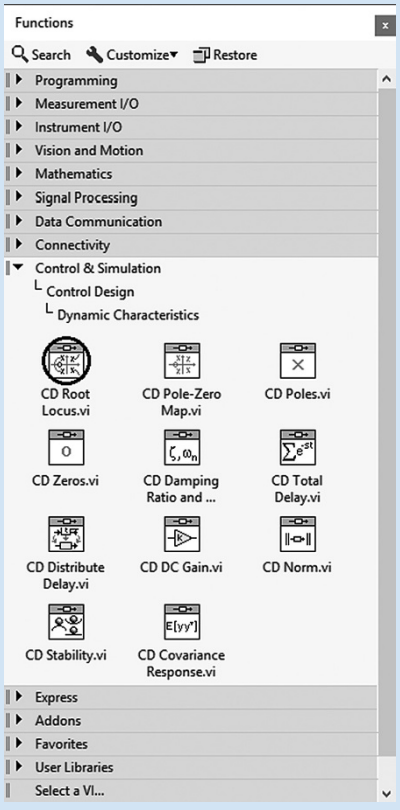
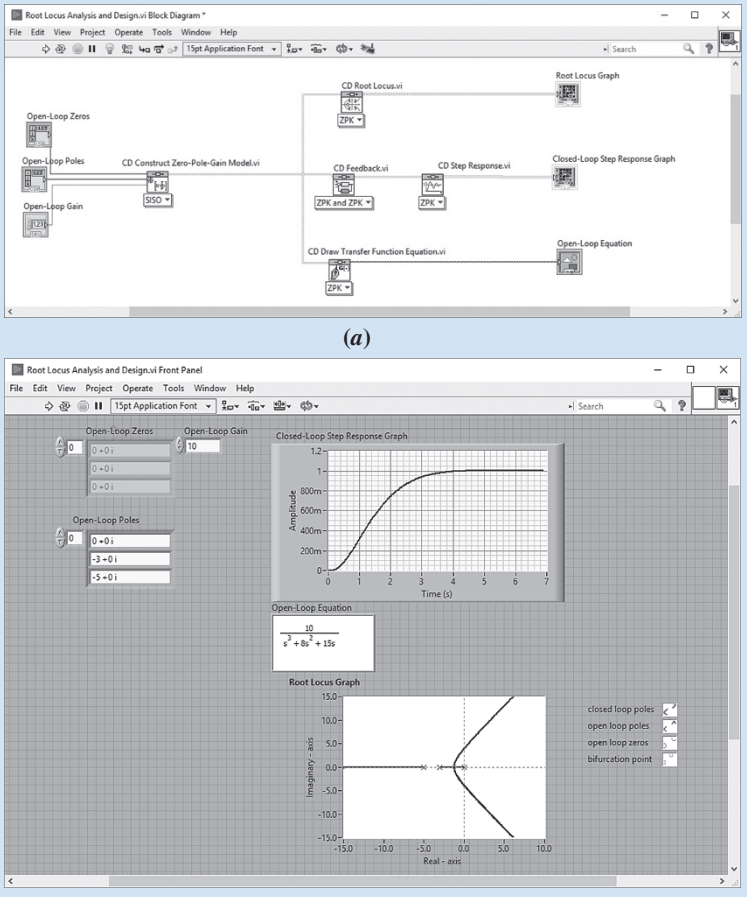


FIGURE D.13 **Functions** palette showing location of **Root Locus** block



(*a*)

(*b*)

FIGURE D.14 Windows showing root locus analysis: **a. Block Diagram**; **b. Front Panel**

Example D.4

Open- and Closed-Loop Sinusoidal Frequency Analysis and Design

We can obtain open- and closed-loop sinusoidal frequency response curves by replacing the **Root Locus** block with the **Bode** block to yield the open-loop frequency response. A copy of the **Bode** block can be added at the output of the **Feedback** block to obtain the closed-loop frequency response. Figure D.15 shows where to obtain the **Bode** block.

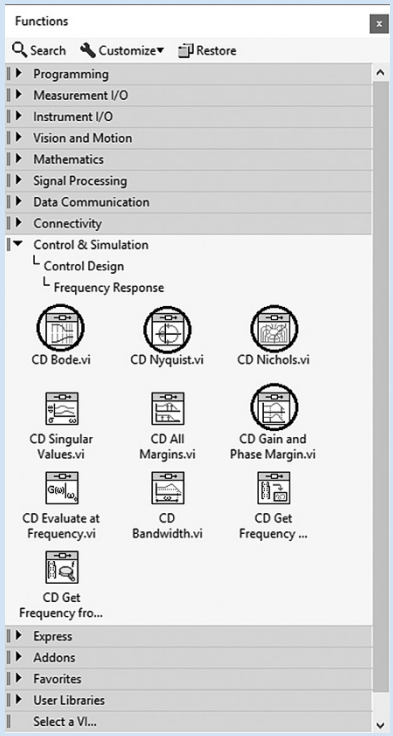
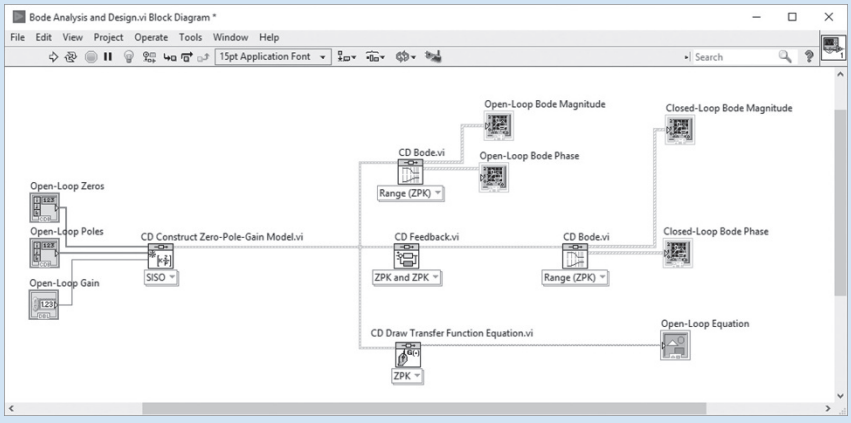


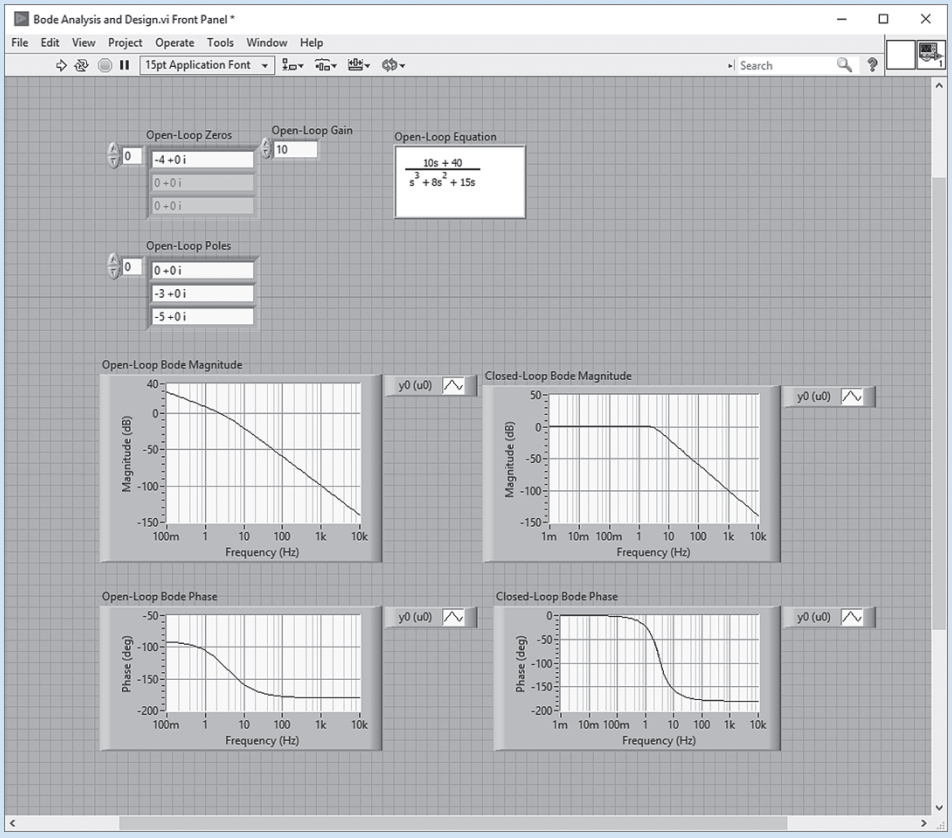
FIGURE D.15 **Functions** window showing frequency response blocks, such as **Bode**, **Nyquist**, **Nichols**, and **Gain and Phase Margin** blocks

Figure D.16 shows the **Block Diagram** and **Front Panel** with open- and closed-loop Bode analysis. In order to display the plots, the indicators shown at the outputs of the **Bode** blocks were created.



(*a*)

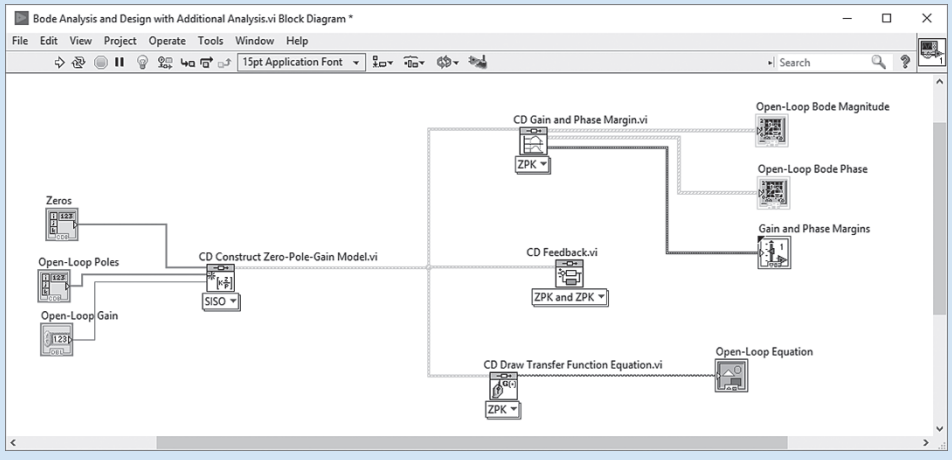
FIGURE D.16 Bode analysis via LabVIEW: **a. Block Diagram**; (*figure continues*)



(*b*)

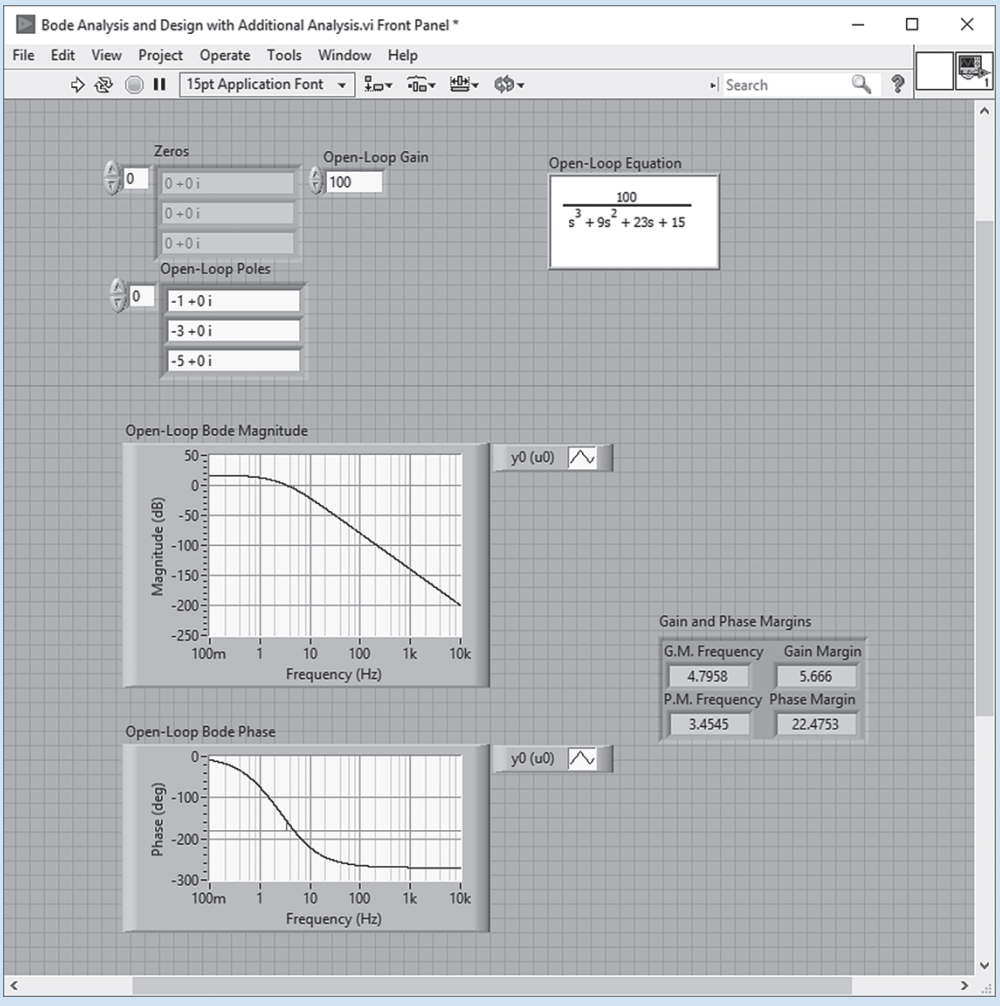
FIGURE D.16 (*continued*) **b. Front Panel**

Figure D.15 shows other alternatives for frequency response analysis. For example, in addition to the Bode plots, you can create an indicator telling you the gain and phase margins by using the **Gain and Phase Margin** block. Figure D.17 shows that result.



(*a*)

FIGURE D.17 Bode analysis with gain and phase margin: **a. Block Diagram**; (*figure continues*)



(*b*)

FIGURE D.17 (*continued*) **b. Front Panel**

Finally, if you need to use Nyquist or Nichols charts, the associated blocks are shown in Figure D.15 and can replace the **Bode** blocks.

D.5 Simulation Examples

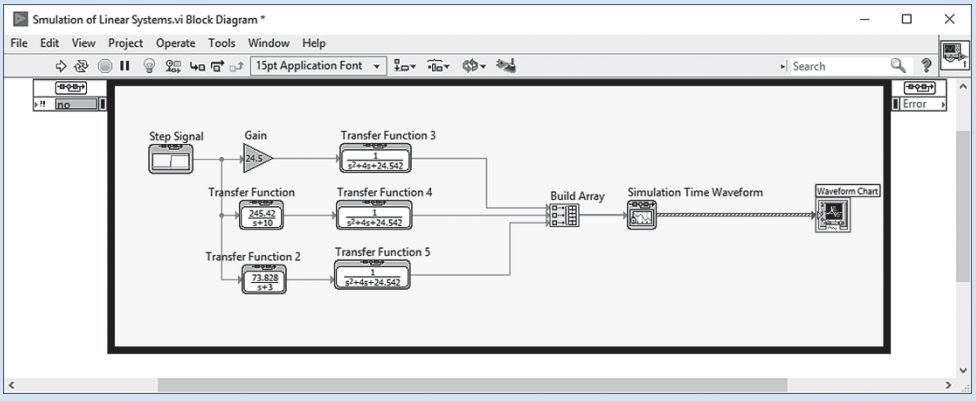
Whereas the LabVIEW block sequence for design and analysis is analogous to following the code statement sequence in a MATLAB M-file, the LabVIEW block sequence for simulation is analogous to following the block sequence of a Simulink diagram.

In this section, we show examples of simulation using LabVIEW. For control system simulation, icons for the block diagram are taken from the **Simulation** subpalette under the **Control & Simulation** palette. Our examples will parallel the examples shown in Appendix C which uses Simulink.

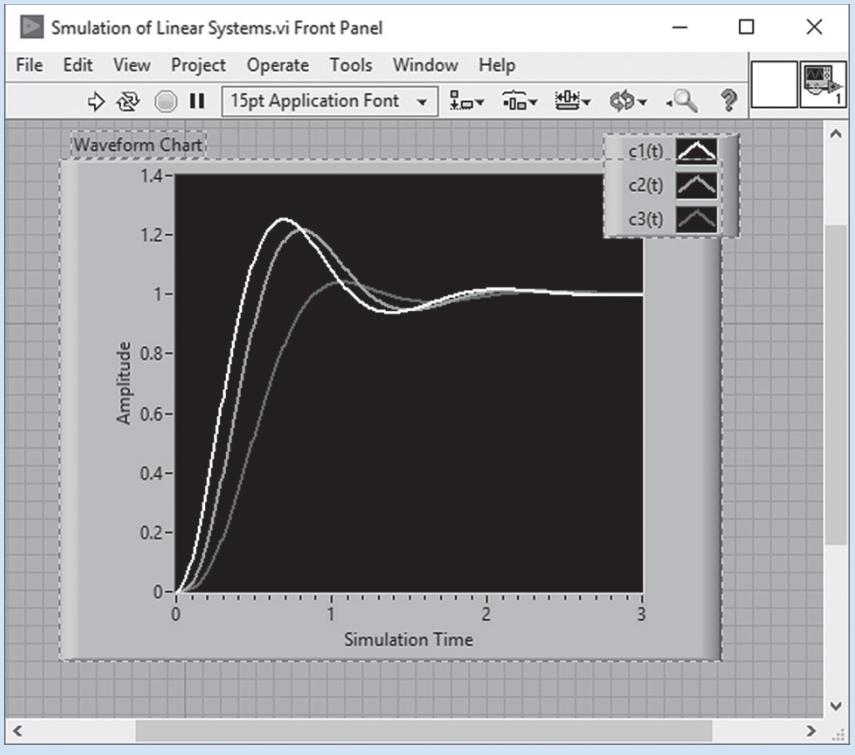
Example D.5

Simulation of Linear Systems

Create Block Diagram and Front Panel Figure D.18 shows the **Block Diagram** and **Front Panel** for simulating a linear system. The simulation reproduces Example C.1 in Appendix C, which uses Simulink. Blocks are selected from the **Simulation** subpalette under the **Control & Simulation** palette and must be placed within the **Simulation Loop** obtained from **Functions/Control & Simulation/Simulation/Control & Simulation Loop**. We now enumerate the detailed steps required to create the **Block Diagram** and **Front Panel:**



(*a*)



(*b*)

FIGURE D.18 Simulation of linear systems: **a. Block Diagram**; **b. Front Panel**

1. Transfer functions are obtained from **Functions/Control & Simulation/Simulation/Continuous Linear Systems/Transfer Function**. Right-click on each transfer function and select **Configuration** to enter the parameter values shown in Figure D.18(*a*) or equivalently in Figure C.5.
2. The gain block is obtained from **Functions/Control & Simulation/Simulation/Signal Arithmetic/Gain**. Right-click on the **Gain** block and select **Configuration** to enter the parameter value.
3. The step-input block is obtained from **Functions/Control & Simulation/Simulation/Signal Generation/Step Signal**. Right-click on the **Step Signal** block and select **Configuration** to enter the parameter value.
4. In order to display the three step-response curves simultaneously, we use a **Build Array** block obtained from **Functions/Programming/Array/Build Array**. Drag the bottom of the icon to expose the correct number of inputs (three for this case).
5. To create the display, we use the **Simulation Time Waveform** block obtained from **Functions/Control & Simulation/Simulation/Graph Utilities/Simtime Waveform**. Right-click the output of the **Simtime Waveform** block and select **Create/Indicator** to produce the **Waveform Chart** icon and the **Front Panel** display.

Configure simulation loop Finally, set the simulation parameters by right-clicking the **Simulation Loop** and selecting **Configure Simulation Parameters**.... Set the parameters as shown in Figure D.19.

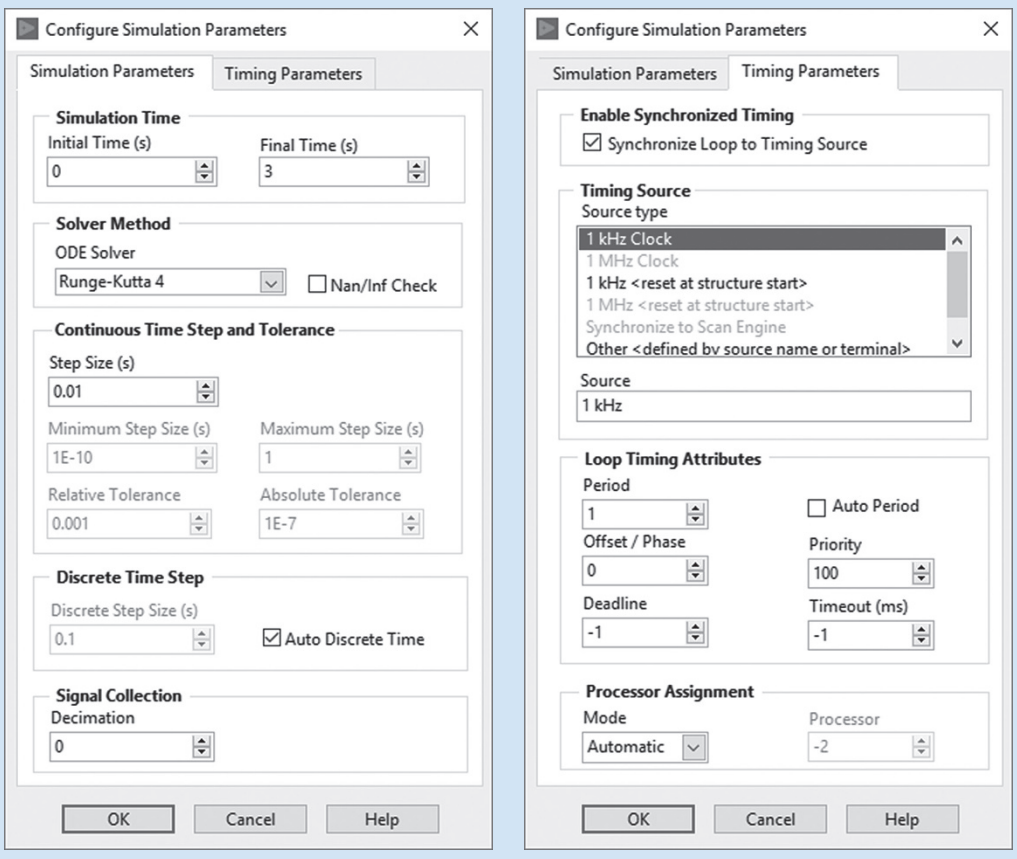


FIGURE D.19 Configuring the **Simulation Loop** parameters: **a. Simulation parameters**; **b. Timing parameters**

Configure graph parameters On the **Front Panel**, right-click the graph and select **Properties** to configure graph parameters if required. Select the legend and expand it vertically to expose all three plot identities. The titles in the legend can be changed to reflect meaningful labels for the plots.

Run the simulation Perform the simulation by clicking the arrow at the extreme left of the toolbar on the **Front Panel** window. You can erase curves between trials by right-clicking the display and selecting **Data Operations/Clear Chart**.

Example D.6

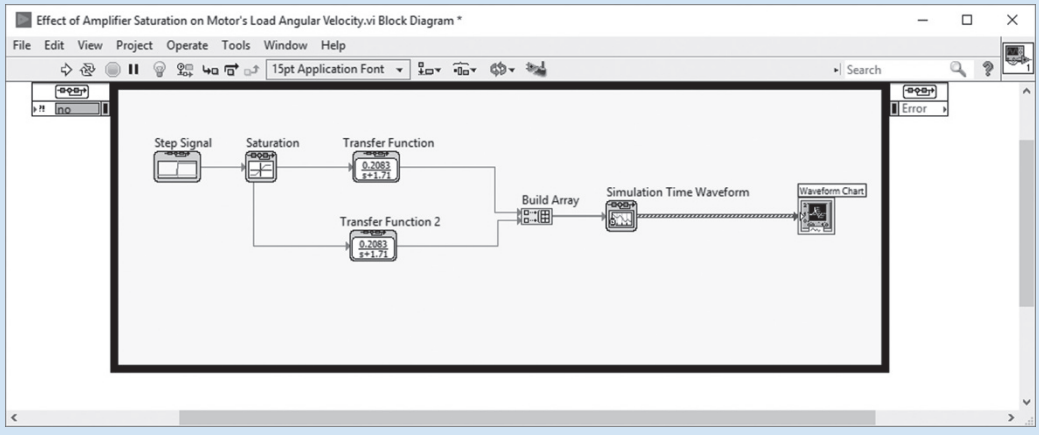
Effect of Amplifier Saturation on Motor's Load Angular Velocity

Create Block Diagram and Front Panel The **Block Diagram** and **Front Panel** for simulating a dc motor with and without saturation are shown in Figure D.20. The **Saturation** block is obtained from **Control & Simulation/Simulation/ Nonlinear Systems/Saturation**.

Configure simulation loop Configure the simulation loop as shown in figure D.19, except change the **Final Time (s)** in Figure D.19(*a*) to 10.

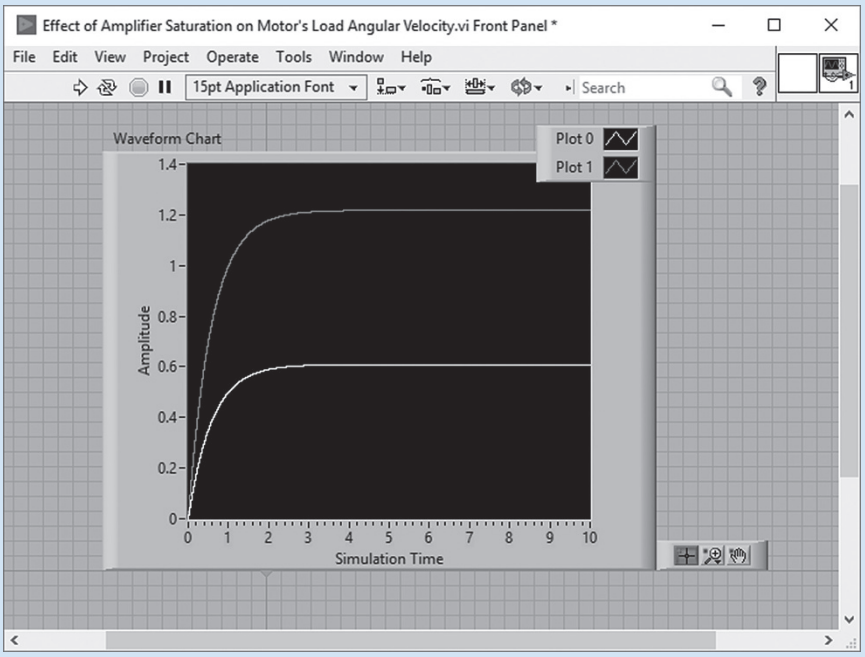
Configure graph parameters On the **Front Panel**, right-click the graph and select **Properties** to configure graph parameters. Select the **Scales** tab and enter 10 in the **Maximum** box as shown in Figure D.21. Select the legend and expand it vertically to expose both plot identities. The titles in the legend can be changed to reflect meaningful labels for the plots.

Run the simulation Perform the simulation by clicking the arrow at the extreme left of the toolbar on the **Front Panel** window. You can erase curves between trials by right-clicking the display and selecting **Data Operations/Clear Chart**.



(*a*)

FIGURE D.20 Simulation of a dc motor with and without saturation: **a. Block Diagram**; (*figure continues*)



(*b*)

FIGURE D.20 (*continued*) **b. Front Panel**

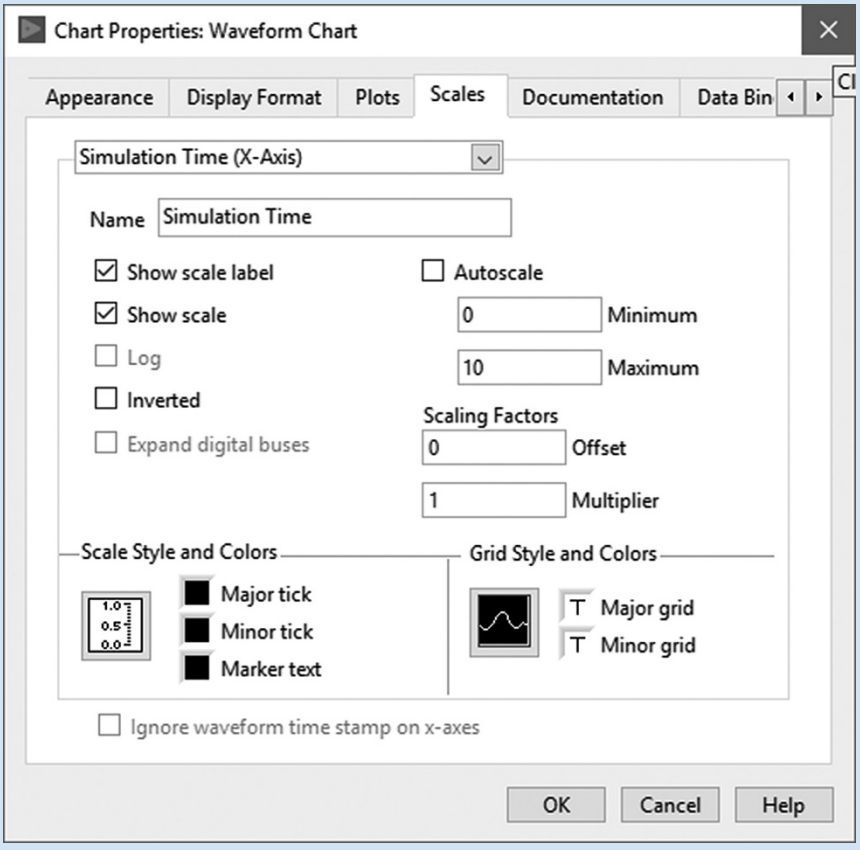


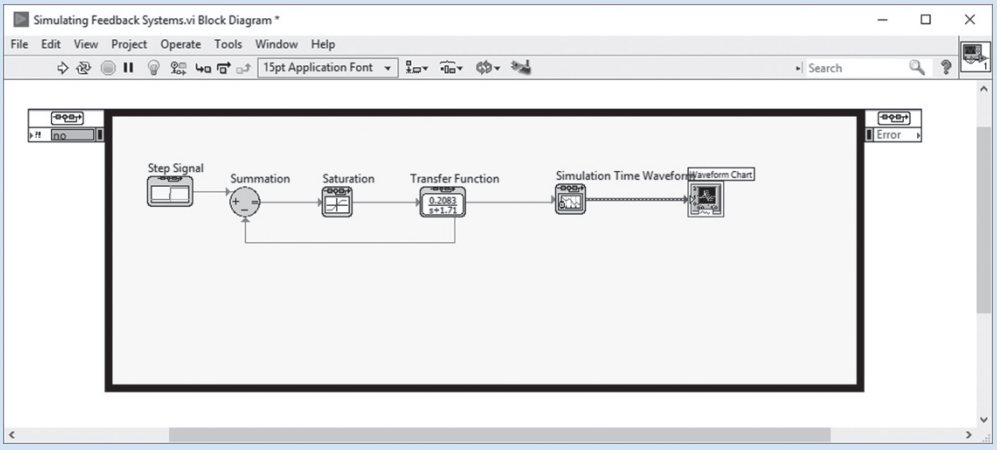
FIGURE D.21 **Chart Properties: Waveform Chart** Window

Example D.7

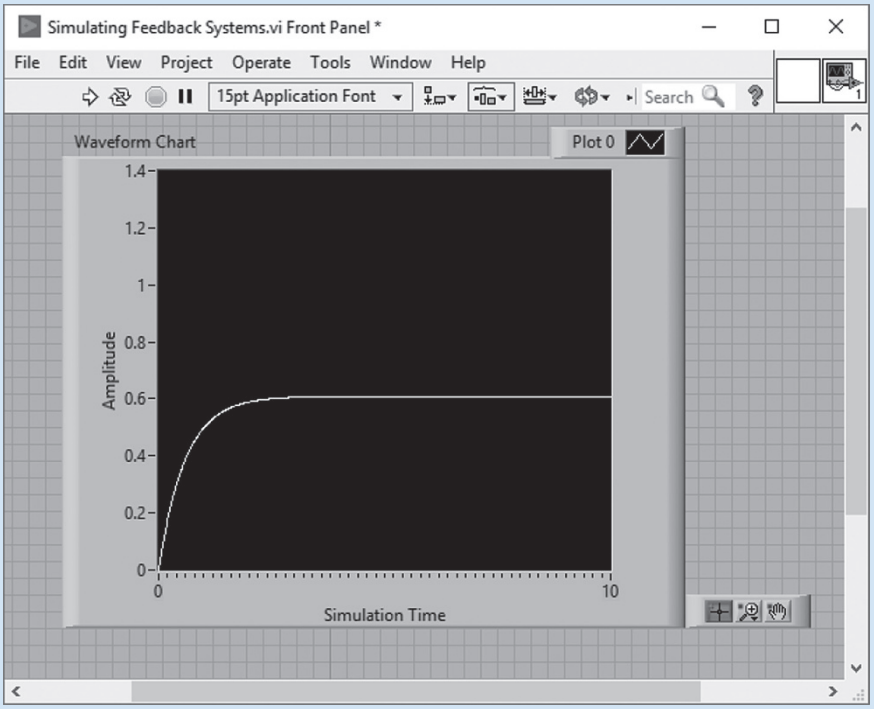
Simulating Feedback Systems

Create Block Diagram and Front Panel The **Block Diagram** and **Front Panel** for simulating feedback systems is shown in Figure D.22. The **Summation** block is obtained from **Control & Simulation/Simulation/Signal Arithmetic/Summation**.

Configure Summation and other blocks Right-click the **Summation** block and select **Configuration**.... Repeat for other blocks.



(*a*)



(*b*)

FIGURE D.22 Simulation of feedback systems: **a. Block Diagram**; **b. Front Panel**

Configure simulation loop Configure the simulation loop as shown in Figure D.19, except change the **Final Time (s)** in Figure D.19(*a*) to 10.

Configure graph parameters On the **Front Panel**, right-click the graph and select **Properties** to configure graph parameters. Select the **Scales** tab and enter 10 in the **Maximum** box as shown in Figure D.21.

Run the simulation Perform the simulation by clicking the arrow at the extreme left of the toolbar on the **Front Panel** window. You can erase curves between trials by right-clicking the display and selecting **Data Operations/Clear Chart**.

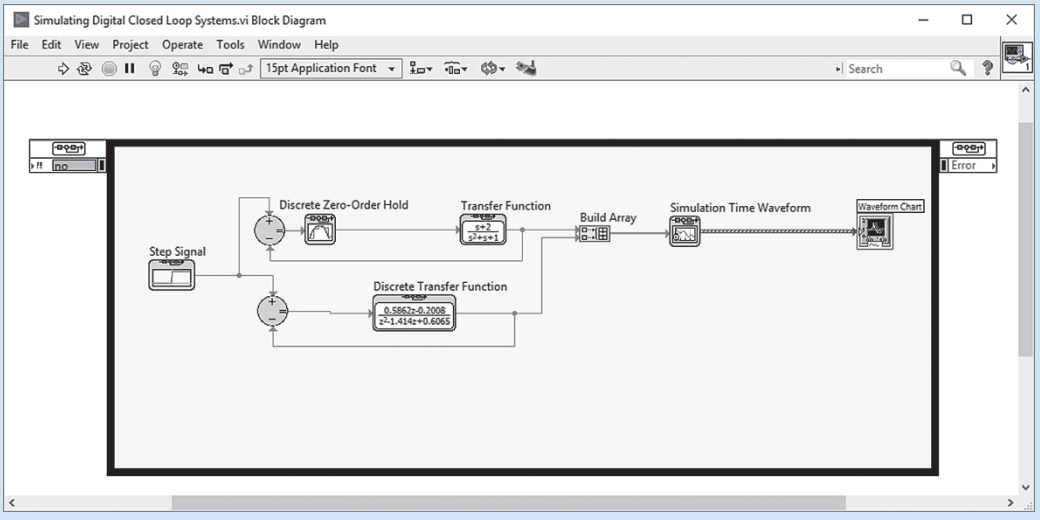
Example D.8

Simulating Digital Systems with the Simulation Palette

Digital systems, such as Example C.4 in Appendix C, can be simulated using LabVIEW. However, there are restrictions on the transfer functions used in the simulation. LabVIEW requires that all inputs to the transfer functions be present at the beginning of the simulation or else a cycle error will result. Unfortunately, this requirement limits the use of transfer functions to those with a denominator of higher order than the numerator. Under these conditions, the reader is advised to use either MATLAB or the **Control Design** palette rather than the **Simulation** palette of the **Control & Simulation** function.

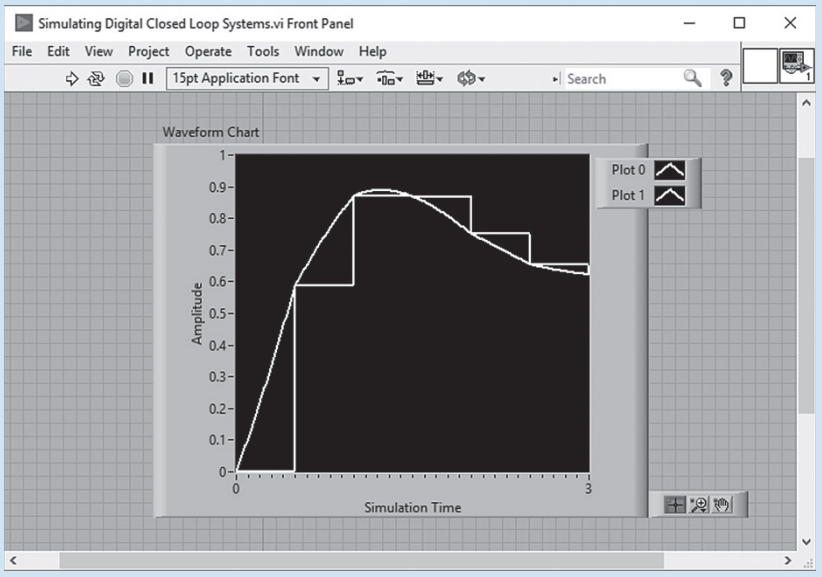
Our first digital example will simulate a digital feedback system using the **Simulation** palette with proper transfer functions. The next example will simulate Example C.4 in Appendix C, which does not have proper transfer functions, using LabVIEW's **Control Design** palette.

Create Block Diagram and Front Panel The **Block Diagram** and **Front Panel** for simulating digital systems is shown in Figure D.23. The **Discrete Zero-Order Hold** block is obtained from **Control Design & Simulation/Simulation/Discrete Linear**



(*a*)

FIGURE D.23 Simulation of digital systems with **Simulation** palette: **a. Block Diagram**; (*figure continues*)



(*b*)

FIGURE D.23 (*continued*) **b. Front Panel**

Systems/**Discrete Zero-Order Hold**. The **Discrete Transfer Function** is obtained from **Control & Simulation/Simulation/Discrete Linear Systems/Discrete Transfer Function**.

Configure Discrete Zero-Order Hold and other blocks Right click the **Discrete Zero-Order Hold** block and select **Configuration…**. Set the sample period to 0.5 second. Configure the transfer functions as shown on the **Block Diagram**. Configure the **Step Signal** to be a unit step.

Configure simulation loop Configure the simulation loop as shown in Figure D.19.

Configure graph parameters On the **Front Panel**, right-click the graph and select **Properties** to configure graph parameters. Select the **Scales** tab and enter 3 in the **Maximum** box for the *x* axis and 1 for the *y* axis. Select the legend and expand it vertically to expose both plot identities. The titles in the legend can be changed to reflect meaningful labels for the plots.

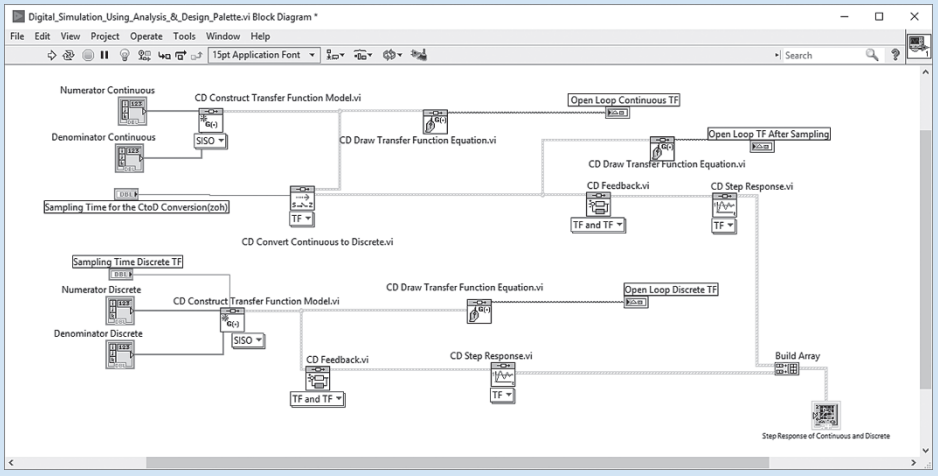
Run the simulation Perform the simulation by clicking the arrow at the extreme left of the toolbar on the **Front Panel** window. You can erase curves between trials by right-clicking the display and selecting **Data Operations/Clear Chart**.

The simulation shows the difference in responses obtained by (1) modeling the digital system as a zero-order hold cascaded with a linear system (Plot 0), or (2) modeling the system with a digital transfer function (Plot 1).

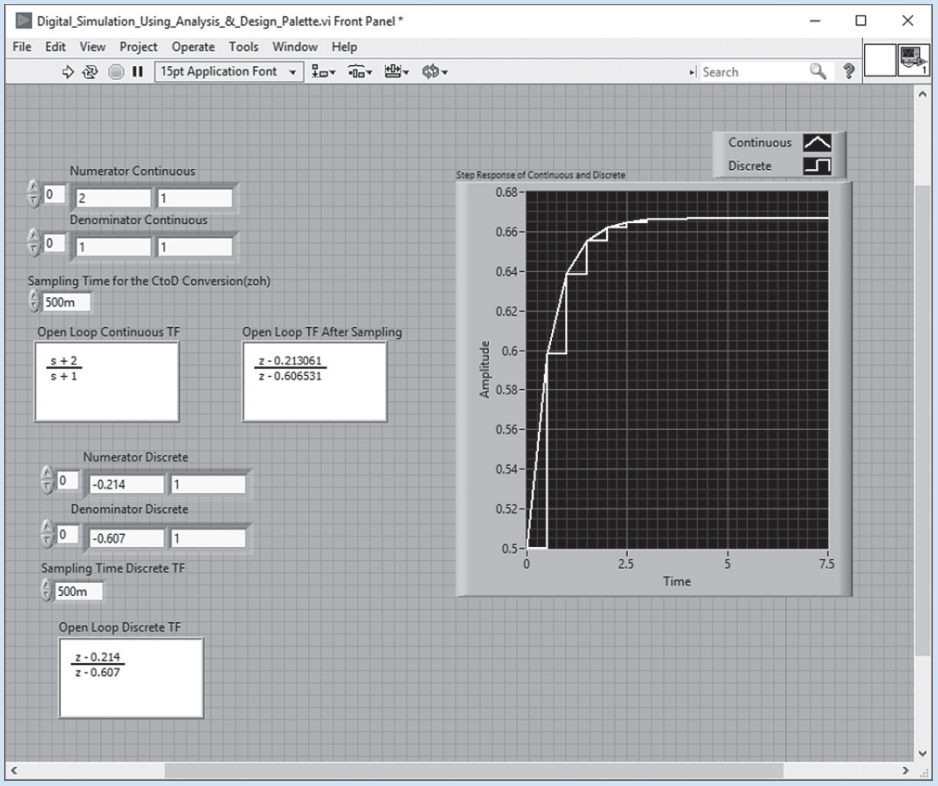
Example D.9

Simulating Digital Systems with the Control Design Palette

In order to avoid cycle errors in LabVIEW, we use the **Control Design** palette when we have transfer functions for which the numerator and denominator are of the same order. This example reproduces Simulink Example C.4.



(*a*)



(*b*)

FIGURE D.24 Simulation of digital systems with **the Control Design** palette: **a. Block Diagram**; **b. Front Panel**

Create Block Diagram and Front Panel The **Block Diagram** and **Front Panel** for this example are shown in Figure D.24. Wire the blocks as shown.

Most of the blocks were previously discussed in Examples D.1 and D.2. Digital transfer functions are created using the same blocks as continuous systems, but with a nonzero **Sampling Time(s)** input.

The **CD Convert Continuous to Discrete.vi**, is obtained from **Functions/ Control Design & Simulation/Control Design/Model Conversion/CD Convert Continuous to Discrete.vi**.

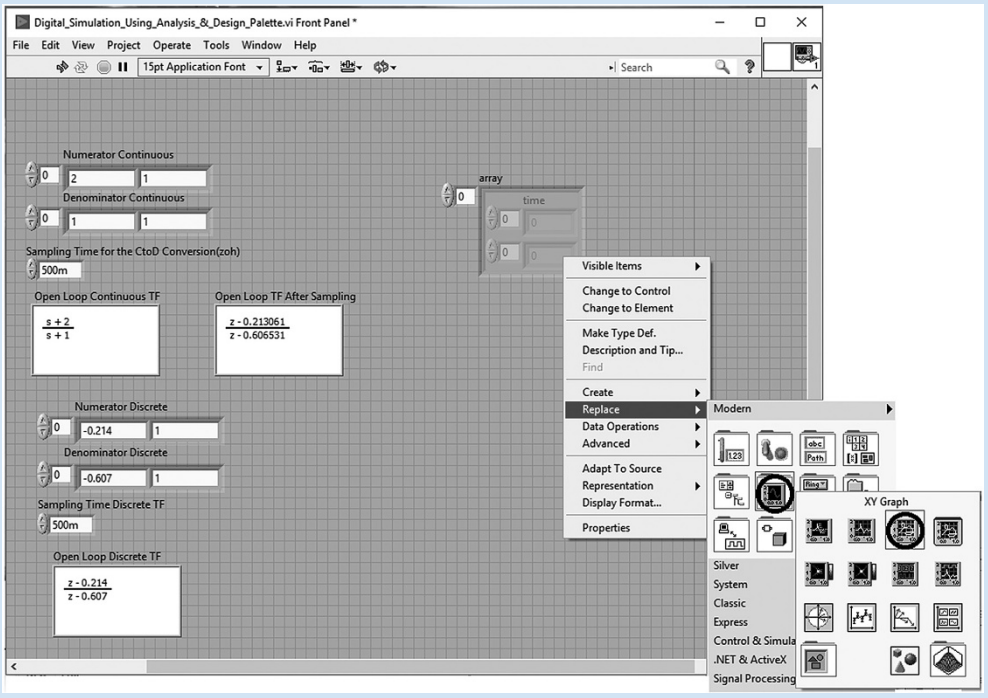


FIGURE D.25 Choosing **XYGraph**

The **Build Array** is obtained from **Functions/Programming/Array/Build Array**. Expand the **Build Array** block to show two inputs.

Configure parameters for Build Array Right-click on **Build Array** and select **Concatenate Inputs**. Right-click again on **Build Array** and select **Create/Indicator**.

Select and then right-click the indicator on the front panel and choose **Replace**.[1] Using the resulting palettes as shown in Figure D.25, select the **XY Graph**.[2]

On the front panel expand the legend to show two graphs. Title the legend components as shown. Change the x- and y-axes' starting and ending points as desired by right-clicking the graph and selecting **Properties**. In the **Properties** window, select **Scales** and enter the desired information. Finally, select **Plots** and enter your choices.

Right-click the graph on the front panel and select **Data Operations** and make your current values the default. Also, right-click again and choose to reinitialize to your default values. You may also choose to clear the current plot.

Configure parameters for CD Convert Continuous to Discrete.vi Right-click and create a control for **Sample Time(s), Numerator**, and **Denominator** as described in Example D.1. Set the values as shown on the **Front Panel**.

Configure parameters for CD Construct Transfer Function Model.vi as a discrete model Right-click and create a control for **Sample Time(s)**, **Numerator**, and **Denominator** as described in Example D.1. Set the values as shown on the **Front Panel**.

Configure parameters for all CD Draw Transfer Function Equation.vi Right-click and create a control for **Equation** as described in Example D.1. Set the values as shown on the **Front Panel**.

Run simulation See Example D.1 for a description. The results are shown in Figure D.24(*b*).

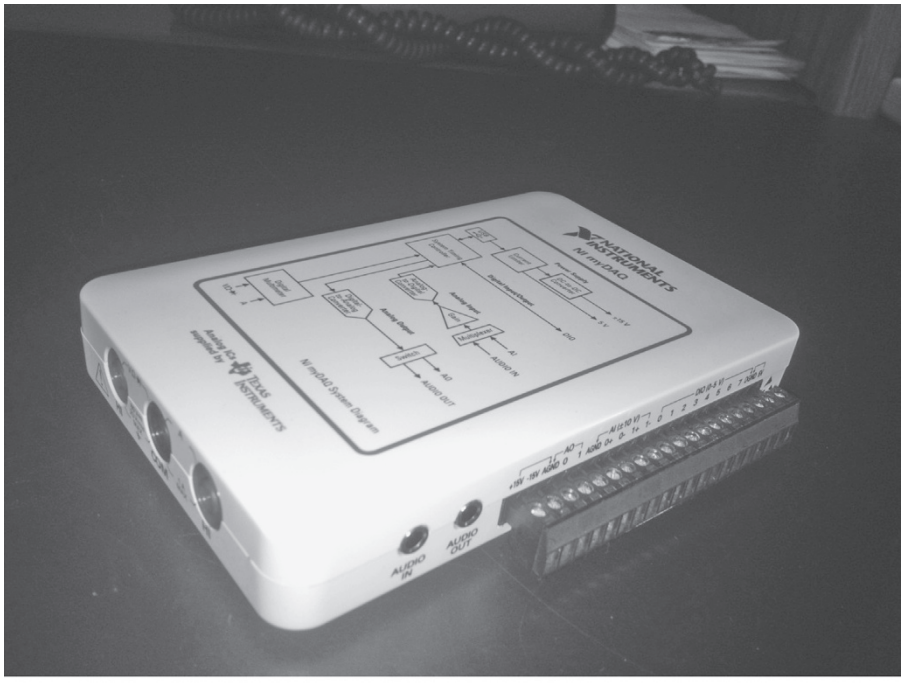
[1] Be sure to select the entire indicator. Then place your mouse on the outer edge and be sure it is a pointer before selecting **Replace**.

[2] Right click the **XY Graph** on the front panel and open **Properties**. On the **Appearance** tab, select 2 for **Plots shown**.

D.6 Interfacing with External Hardware

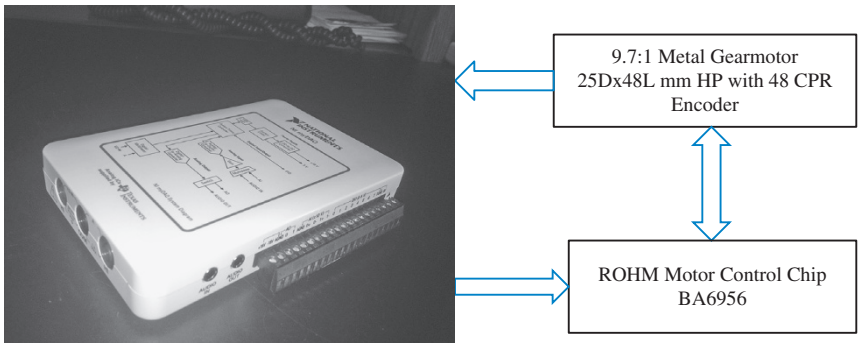
This section provides an introduction to the use of LabVIEW virtual instruments to control external hardware with the **NI myDAQ** for students. Specifically, we concentrate on using the **NI myDAQ** to analyze and design an actual feedback system used in the Hardware Exploration Laboratory file located at www.wiley.com/go/Nise/ControlSystemsEngineering8e.

1. **Required Hardware** **NI myDAQ** is a data acquisition module that is ideal for student experimentation, since it is portable and of low cost. In order to perform the experiments, you will need a motor control chip and a gearmotor. Although other alternatives exist, we use the following: (1) ROHM motor control chip, BA6956; and (2) 9.7:1 metal gearmotor 25Dx48L mm HP with 48 CPR encoder, which can be obtained from www.pololu.com.
2. **Required Software** The software required to support the experiments are: (1) LabVIEW or the Student Version of LabVIEW and (2) **NI ELVISmx Soft Front Panel (SFP) Instruments**, which comes bundled with the **NI myDAQ. NI ELVISmx** provides virtual instruments that will generate input signals to and acquire output signals from your external control system.
3. **Basic Configuration** Figure D.26 shows the **NI myDAQ** and the basic configuration that will be used to perform control system design and analysis. Detailed wiring diagrams will accompany specific experiments.
4. **Launching NI myDAQ** The **NI myDAQ** kit comes with the following cables and connectors: (1) USB cable; (2) 20-position screw terminal connector; (3) audio cable; and (4) DMM banana cable. The following steps will launch the **NI myDAQ** and **NI ELVISmx**, which provides convenient virtual instruments for control and data acquisition.

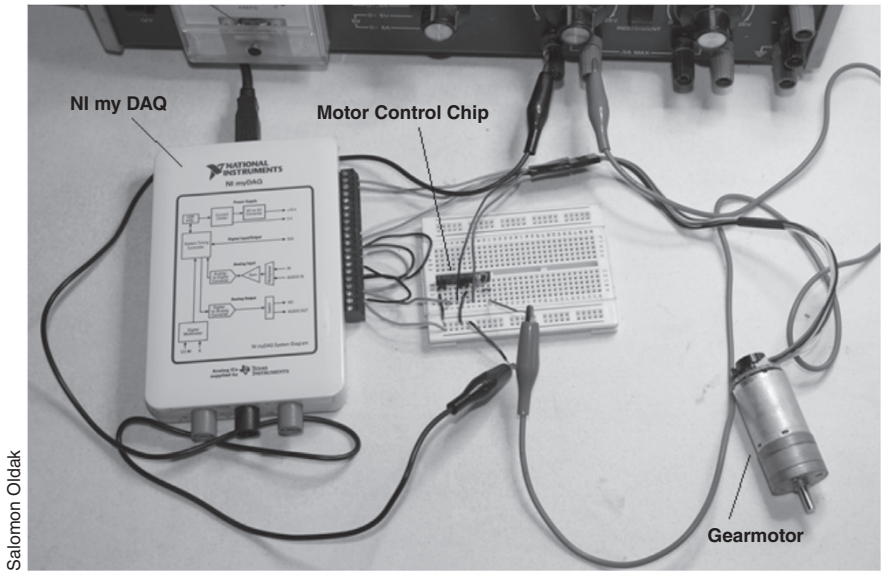


(*a*)

FIGURE D.26 **a. NI myDAQ**; (*figure continues*)



(*b*)



(*c*)

FIGURE D.26 (*continued*) **b.** basic configuration showing **NI myDAQ** interfaced with motor control chip and gearmotor; **c.** interconnected hardware

Step 1: Be sure LabVIEW is installed on your computer.

Step 2: Install the **NI myDAQ** software.

Step 3: Connect the **NI myDAQ** to your computer via the USB cable.

Step 4: After **NI myDAQ** is recognized, **NI ELVISmx** should launch automatically. If **NI ELVISmx** does not launch, then manually launch from the **Start** menu **National Instruments/NI ELVISmx Instrument Launcher**. Figure D.27 shows the window containing the virtual instruments available from **NI ELVISmx**. Clicking on any instrument will bring up that VI.

Simple Experiments Using the NI myDAQ

1. **Measuring battery voltage using the NI myDAQ Digital Multimeter (DMM)** With myDAQ connected to your computer, launch the DMM shown in Figure D.27. Attach the DMM probes between the myDAQ and a battery as shown in Figure D.28(*a*). Press the green **Run** arrow.

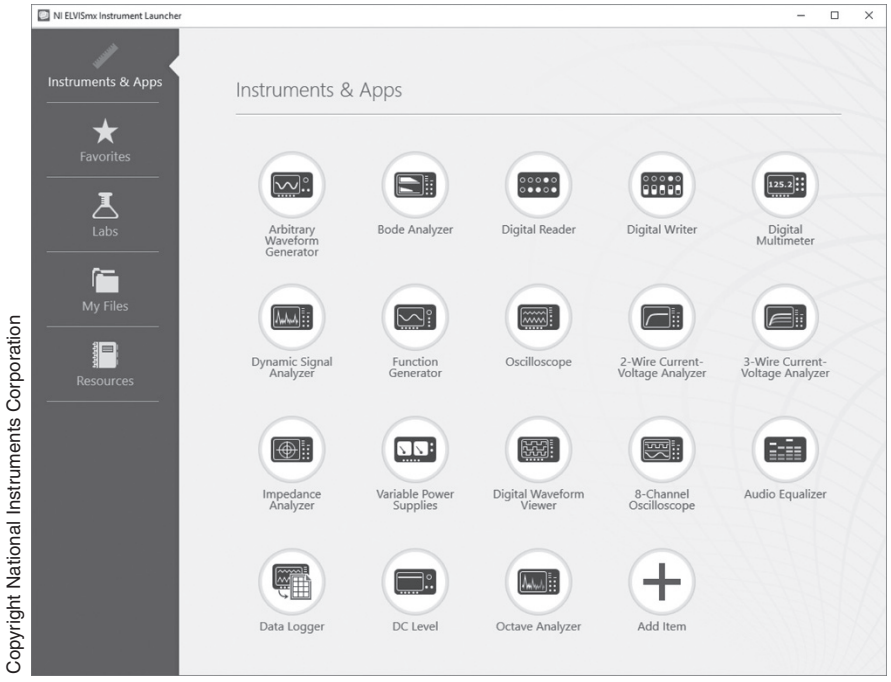
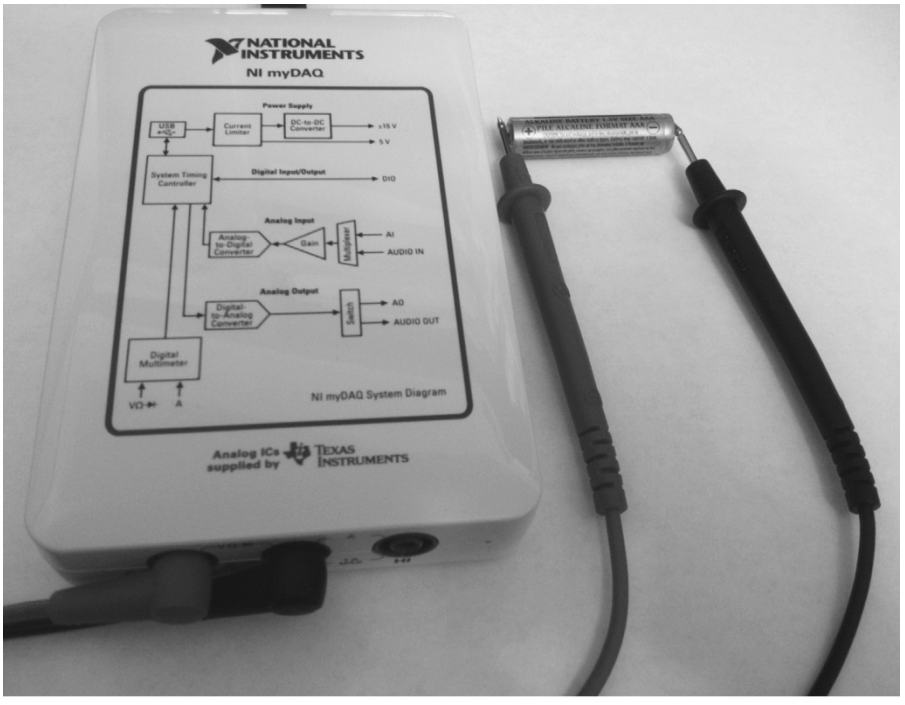
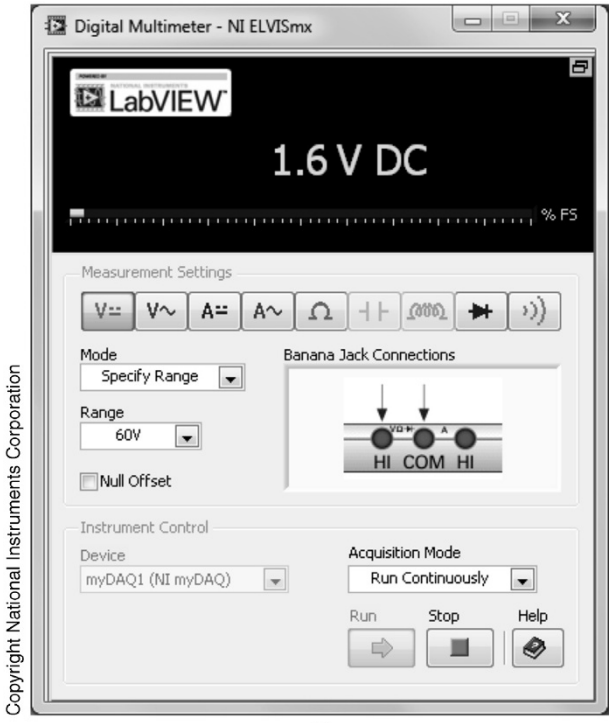


FIGURE D.27 The **NI ELVISmx Instrument Launcher** window



(*a*)

FIGURE D.28 Battery voltage measurement: **a.** Connections to myDAQ; (*figure continues*)



(*b*)

FIGURE D.28 (*continued*)
b. DMM reading

2. The NI myDAQ audio equalizer With myDAQ connected to your computer, launch the **Audio Equalizer** in the **Instruments & Apps** menu shown in Figure D.27. Attach an audio cable from your music source to **AUDIO IN** on your myDAQ. Similarly, attach speakers or an earphone to the **AUDIO OUT**. Press the blue **Start** arrow. You can now adjust volume, bass, midtone, and treble as well as watch the audio waveform in time or frequency. The **Audio Equalizer** is shown in Figure D.29.

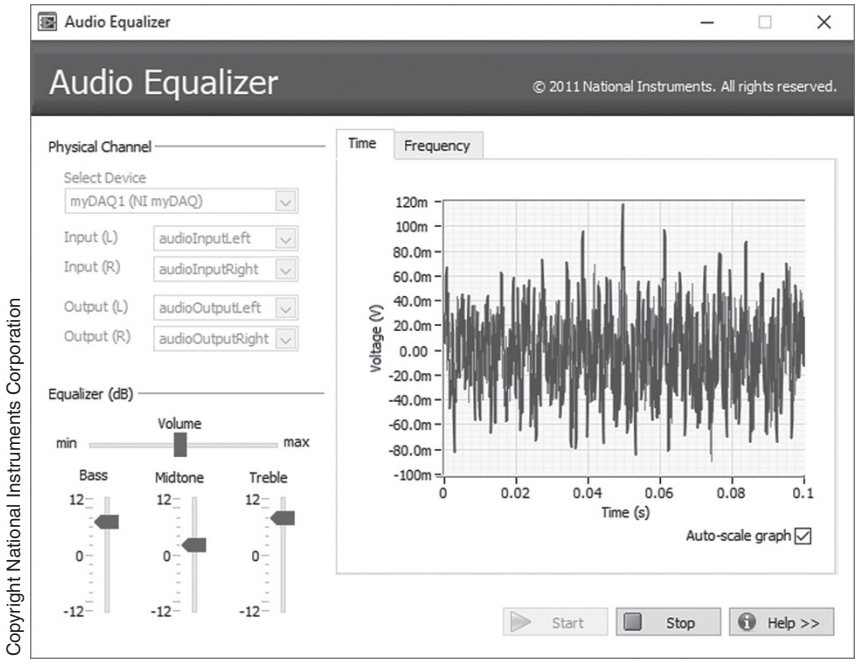


FIGURE D.29 The NI myDAQ Audio Equalizer

Summary

This appendix presented LabVIEW as an alternative to MATLAB for analysis, design, and simulation. Our discussion was divided into analysis and design, and simulation.

Analysis and design is performed by interconnecting code blocks, which is analogous to writing in-line code for MATLAB M-files. Since the LabVIEW code blocks are represented by icons, an advantage of using LabVIEW is that you do not have to know specific code statements.

Simulation is performed by interconnecting code blocks and is analogous to Simulink flow diagrams.

LabVIEW has more extensive applications. You can create virtual instruments on your computer monitor that can operate external hardware and transmit and receive telemetric data. We covered a few of these applications in this appendix using the NI myDAQ. It is left to the interested reader to pursue more of these applications.

Bibliography

National Instruments. *LabVIEW 2017 Help*. National Instruments, Austin, TX. 2017. See http://zone.ni.com/reference/en-XX/help/371361P-01/

National Instruments. *LabVIEW Fundamentals*. National Instruments, Austin, TX. 2005.

National Instruments. *LabVIEW™ Control Design User Manual*. National Instruments, Austin, TX. 2004–2009.

National Instruments. *Introduction to LabVIEW in 3 Hours for Control Design and Simulation*. National Instruments Course Notes, Austin, TX. 2009.

National Instruments. *NI myDAQ User Guide*. National Instruments, Austin, TX. 2010–2016.

*All LabVIEW screenshots and equipment in this book are reprinted with permission of National Instruments.