LabVIEW™

Control Design Toolkit User Manual

January 2005
371057C-01

NATIONAL INSTRUMENTS™

Worldwide Technical Support and Product Information

`ni.com`

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599, Canada 800 433 3488, China 86 21 6555 7838, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11, France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, India 91 80 51190000, Israel 972 0 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400, Lebanon 961 0 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 0 348 433 466, New Zealand 0800 553 322, Norway 47 0 66 90 76 60, Poland 48 22 3390150, Portugal 351 210 311 210, Russia 7 095 783 68 51, Singapore 1800 226 5886, Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 0 8 587 895 00, Switzerland 41 56 200 51 51, Taiwan 02 2377 2222, Thailand 662 992 7519, United Kingdom 44 0 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at `ni.com/info` and enter the info code `feedback`.

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on `ni.com/legal` for more information about National Instruments trademarks.

Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/patents`.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Contents

About This Manual

Chapter 1
Introduction to Control Design

Chapter 2
Constructing Dynamic System Models

Chapter 3
Converting Models

Chapter 4
Connecting Models

Chapter 5
Working with Delay Information

Chapter 6
Time Response Analysis

Chapter 7
Frequency Response Analysis

Chapter 8
Analyzing Dynamic Characteristics

Chapter 9
Analyzing State-Space Characteristics

Chapter 10
Model Order Reduction

Chapter 11
Designing Classical Controllers

Chapter 12
Designing State-Space Controllers

Chapter 13
Defining State Estimator Structures

Chapter 14
Defining State-Space Controller Structures

Appendix A
Technical Support and Professional Services

About This Manual

This manual contains information about the purpose of control design and the control design process. This manual also describes how to develop a control design system using the LabVIEW Control Design Toolkit.

This manual requires that you have a basic understanding of the LabVIEW environment. If you are unfamiliar with LabVIEW, refer to the *Getting Started with LabVIEW* manual before reading this manual.

Note This manual is not intended to provide a comprehensive discussion of control design theory. Refer to the following books for more information about control design theory: *Modern Control Systems*[1], *Feedback Control of Dynamic Systems*[2], *Digital Control of Dynamic Systems*[3], *Control Systems Engineering*[4], and *Modern Control Engineering*[5].

Conventions

The following conventions appear in this manual:

» — The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.

This icon denotes a note, which alerts you to important information.

bold — Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

---

1 Dorf, Richard C., and Robert H. Bishop. *Modern Control Systems*, 9th ed. Upper Saddle River, NJ: Prentice Hall, 2001.

2 Franklin, Gene F., J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2002.

3 Franklin, Gene F., J. David Powell, and Michael L. Workman. *Digital Control of Dynamic Systems*, 3rd ed. Menlo Park, CA: Addison Wesley Longman, Inc., 1998.

4 Nise, Norman S. *Control Systems Engineering*, 3rd ed. New York: John Wiley & Sons, Inc., 2000.

5 Ogata, Katsuhiko. *Modern Control Engineering*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2001.

italic	Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

Related Documentation

The following documents contain information that you might find helpful as you read this manual:

- *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**
- *Getting Started with LabVIEW*
- *LabVIEW User Manual*
- *LabVIEW Simulation Module User Manual*
- *LabVIEW System Identification Toolkit User Manual*

1

Introduction to Control Design

Control design is a process that involves developing mathematical models that describe a physical system, analyzing the models to learn about their dynamic characteristics, and creating a controller to achieve certain dynamic characteristics. Control systems contain components that direct, command, and regulate the physical system, also known as the plant. In this manual, the control system refers to the sensors, the controller, and the actuators. The reference input refers to a condition of the system that you specify.

The dynamic system, shown in Figure 1-1, refers to the combination of the control system and the plant.

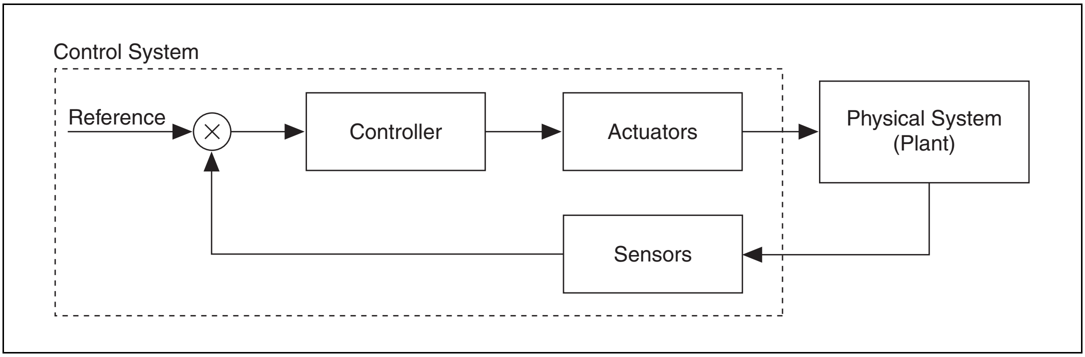

Figure 1-1. Dynamic System

The dynamic system in Figure 1-1 represents a closed-loop system, also known as a feedback system. In closed-loop systems, the control system monitors the outputs of the plant and adjusts the inputs to the plant to make the actual response closer to the input that you designate.

One example of a closed-loop system is a system that regulates room temperature. In this example, the reference input is the temperature at which you want the room to stay. The thermometer senses the actual temperature of the room. Based on the reference input, the thermostat activates the heater or the air conditioner. In this example, the room is the plant, the thermometer is the sensor, the thermostat is the controller, and the heater or air conditioner is the actuator.

Other common examples of control systems include the following applications:

- Automobile cruise control systems
- Robots in manufacturing
- Refrigerator temperature control systems
- Hard drive head control systems

This chapter provides an overview of model-based control design and describes how you can use the LabVIEW Control Design Toolkit to design a controller.

Model-Based Control Design

Model-based control design involves the following four phases: developing and analyzing a model to describe a plant, designing and analyzing a controller for the dynamic system, simulating the dynamic system, and deploying the controller. Because model-based control design involves many iterations, you might need to repeat one or more of these phases before the design is complete. Figure 1-2 shows how National Instruments provides solutions for each of these phases.

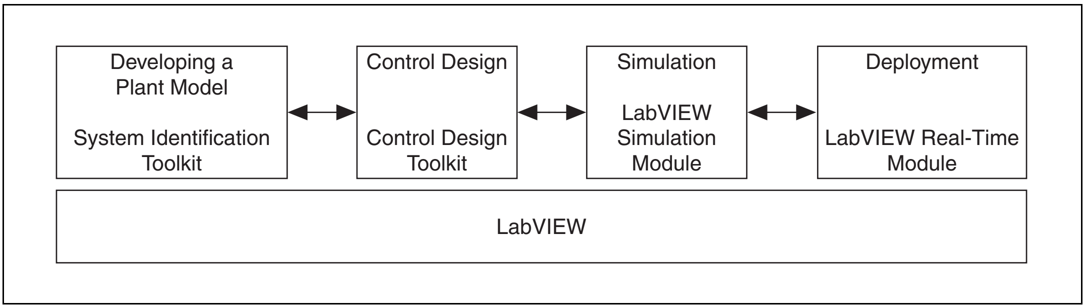

Figure 1-2. Using LabVIEW in Model-Based Control Design

National Instruments also provides products for I/O and signal conditioning that you can use to gather and process data. Using these tools, which are built on the LabVIEW platform, you can experiment with different approaches at each phase in model-based control design and quickly identify the optimal design solution for a control system.

Developing a Plant Model

The first phase of model-based control design involves developing and analyzing a mathematical model of the plant you want to control. You can use a process called system identification to obtain and analyze this model. The system identification process involves acquiring data from a plant and then numerically analyzing stimulus and response data to estimate the parameters and order of the model.

The system identification process requires a combination of the following components:

- **Signal generation and data acquisition**—National Instruments provides software and hardware that you can use to stimulate and measure the response of the plant.
- **Mathematical tools to model a dynamic system**—The LabVIEW System Identification Toolkit contains VIs to help you estimate and create accurate mathematical models of dynamic systems. You can use this toolkit to create discrete linear models of systems based on measured stimulus and response data.

Note This manual does not provide a comprehensive discussion of system identification. Refer to the following books for more information about developing a plant model: *Modern Control Systems*[1], *Feedback Control of Dynamic Systems*[2], *Digital Control of Dynamic Systems*[3], *Control Systems Engineering*[4], and *Modern Control Engineering*[5].

Designing a Controller

The second phase of model-based control design involves two steps. The first step is analyzing the plant model obtained during the system identification process. The second step is designing a controller based on that analysis. You can use the Control Design VIs and tools to complete these steps. These VIs and tools use both classical and state-space techniques.

---

[1] Dorf, Richard C., and Robert H. Bishop. *Modern Control Systems*, 9th ed. Upper Saddle River, NJ: Prentice Hall, 2001.

[2] Franklin, Gene F., J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[3] Franklin, Gene F., J. David Powell, and Michael L. Workman. *Digital Control of Dynamic Systems*, 3rd ed. Menlo Park, CA: Addison Wesley Longman, Inc., 1998.

[4] Nise, Norman S. *Control Systems Engineering*, 3rd ed. New York: John Wiley & Sons, Inc., 2000.

[5] Ogata, Katsuhiko. *Modern Control Engineering*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2001.

Figure 1-3 shows the typical steps involved in designing a controller.

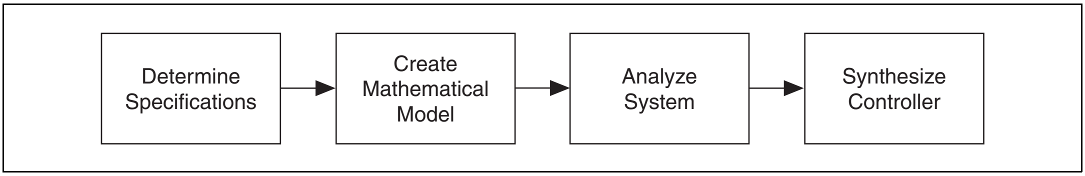

Figure 1-3. Control Design Process

You often iterate these steps to achieve an acceptable design that is physically realizable and meets specific performance criteria.

Simulating the Dynamic System

The third phase of model-based control design involves validating the controller design obtained in the previous phase. You perform this validation by simulating the dynamic system. For example, simulating a jet engine saves time, labor, and money compared to building and testing an actual jet engine.

You can use the Control Design Toolkit to simulate linear time-invariant systems. The LabVIEW Simulation Module, however, provides a variety of different numerical integration schemes for simulating more elaborate systems, such as nonlinear systems. Use the Simulation Module to determine how a system responds to complex, time-varying inputs.

Deploying the Controller

The fourth phase of model-based control design involves deploying the controller to a real-time (RT) target. LabVIEW and the LabVIEW Real-Time Module provide a common platform that you can use to implement the control system.

Refer to the National Instruments Web site at `ni.com` for information about the National Instruments products mentioned in this section.

Overview of the Control Design Toolkit

The Control Design Toolkit provides an interactive Control Design Assistant and a library of VIs for designing a controller based on a model of a plant. You can use both tools to complete the entire control design process from creating a model of the controller to synthesizing the controller on an RT target.

Control Design Assistant

You can use the Control Design Assistant to synthesize and analyze a controller for a user-defined model without knowing how to program in LabVIEW. You access the Control Design Assistant through the NI Express Workbench. The Express Workbench is a framework that can host multiple interactive National Instruments tools and assistants.

You also can use the Control Design Assistant to create a project. In one project, you can load or create a model of a plant into the Control Design Assistant, analyze the time or frequency response, and then calculate the controller parameters. Using the Express Workbench, you immediately can see the mathematical equation and graphical representation that describe the model. You also can view the response data and the configuration of the controller.

Using the Control Design Assistant, you can convert a project to a LabVIEW block diagram and customize that block diagram in LabVIEW. You then can use LabVIEW to enhance and extend the capabilities of the application. Refer to the *NI Express Workbench Help* for more information about using the Control Design Assistant to analyze models that describe a physical system and design controllers to achieve specified dynamic characteristics.

Control Design VIs

The Control Design Toolkit also provides VIs that you can use to create and develop control design applications in LabVIEW. You can use these VIs to develop mathematical models of a dynamic system, analyze the models to learn about their dynamic characteristics, and create controllers to achieve specified dynamic characteristics. You use these VIs to customize a LabVIEW block diagram to achieve specific goals. You also can use other LabVIEW VIs and functions to enhance the functionality of the application. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for information about the Control Design VIs.

Unlike creating a project with the Control Design Assistant, creating a LabVIEW application using the Control Design VIs requires basic knowledge about programming in LabVIEW. Refer to the *LabVIEW User Manual* and the *Getting Started with LabVIEW* manual for more information about the LabVIEW programming environment.

2

Constructing Dynamic System Models

Model-based control design relies upon the concept of a dynamic system model. A dynamic system model is a mathematical representation of the dynamics between the inputs and outputs of a dynamic system. You generally represent dynamic system models with differential equations or difference equations.

Obtaining a model of the dynamic system you want to control is the first step in model-based control design. You analyze this model to anticipate the outputs of the model when given a set of inputs. Using this analysis, you then can design a controller that affects the outputs of the dynamic system in a manner that you specify.

For example, consider the temperature-regulation example in the introduction of Chapter 1, *Introduction to Control Design*. You can analyze the open-loop dynamics of the plant to effectively design a controller for this closed-loop dynamic system. A model for this closed-loop dynamic system describes the input to the plant as the air flow from the vent. The output of the plant is the temperature of the room. By analyzing the relationship between the inputs and output of the plant, you can predict how the plant reacts when given certain inputs. Based on this analysis, you then can design a controller for this dynamic system.

This chapter provides information about using the LabVIEW Control Design Toolkit to create dynamic system models. This chapter also describes the different forms that you can use to represent a dynamic system model.

Constructing Accurate Models

To create a model of a system, think of the system as a black box that continuously accepts inputs and continuously generates outputs. Figure 2-1 shows the basic black-box model of a dynamic system.

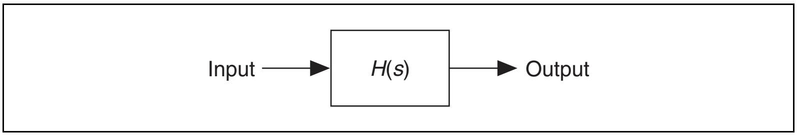

Figure 2-1. Black-Box Model of a Dynamic System

You refer to this model as a black-box model because you often do not know the relationship between the inputs and outputs of a dynamic system. The model you create, therefore, has errors that you must account for when designing a controller.

An accurate model perfectly describes the dynamic system that it represents. Real-world dynamic systems, however, are subject to a variety of non-deterministic fluctuating conditions and interacting components that prevent you from making a perfect model. You must consider many external factors, such as random interactions and parameter variations. You also must consider internal interacting structures and their fundamental descriptions.

Because designing a perfectly accurate model is impossible, you must design a controller that accounts for these inaccuracies. A robust controller is one that functions as expected despite some differences between the dynamic system and the model of the dynamic system. A controller that is not robust might fail when such differences are present.

The more accurate a model is, the more complex the mathematical relationship between inputs and outputs. At times, however, increasing the complexity of the model does not provide any more benefits. For example, if you want to control the interacting forces and friction of a mechanical dynamic system, you might not need to include the thermodynamic effects of the system. These effects are complicated features of the system that do not affect the friction enough to impact the robustness of the controller. A model that incorporates these effects can become unnecessarily complicated.

Model Representation

You can represent a dynamic system using several types of dynamic system models. You also can represent each type of dynamic system model using three different forms. The following sections provide information about the different types and forms of dynamic system models that you can construct with the Control Design Toolkit.

Model Types

You base the type of dynamic system model on the properties of the dynamic system on which the model represents. The following sections provide information about the different types of models you can create with the Control Design Toolkit.

Linear versus Nonlinear Models

Dynamic system models are either linear or nonlinear. A linear model obeys the principle of superposition. The following equations are true for linear models.

$$y_1 = f(x_1)$$

$$y_2 = f(x_2)$$

$$Y = f(x_1 + x_2) = y_1 + y_2$$

Conversely, nonlinear models do not obey the principle of superposition. Nonlinear effects in real-world systems include saturation, dead-zone, friction, backlash, and quantization effects; relays; switches; and rate limiters. Many real-world systems are nonlinear, though you can linearize the model to simplify a design or analysis procedure. You can use the LabVIEW Simulation Module to perform this linearization task.

The Control Design Toolkit supports linear models only.

Time-Variant versus Time-Invariant Models

Dynamic system models are either time-variant or time-invariant. The parameters of a time-variant model change with time. For example, you can use a time-variant model to describe an automobile. As fuel burns, the mass of the vehicle changes with time.

Conversely, the parameters of a time-invariant model do not change with time. For an example of a time-invariant model, consider a simple robot. Generally, the dynamic characteristics of robots do not change over short periods of time.

The Control Design Toolkit supports time-invariant models only.

Continuous versus Discrete Models

Dynamic system models are either continuous or discrete. Continuous models represent real-world signals that vary continuously with time. You use differential equations to describe continuous systems. For example, a model that describes the orbital motion of a satellite is a continuous model.

Conversely, discrete models represent signals that you sample at separate intervals in time. You use difference equations to describe discrete systems. For example, a digital computer that controls the altitude of the satellite uses a discrete model.

Continuous system models are analog, and discrete system models are digital. Both continuous and discrete system models can be linear or nonlinear and time-invariant or time-variant.

The Control Design Toolkit supports continuous and discrete models.

Model Forms

You can use the Control Design Toolkit to represent dynamic system models in the following three forms: transfer function, zero-pole-gain, and state-space. Refer to the *Constructing Transfer Function Models* section, the *Constructing Zero-Pole-Gain Models* section, and the *Constructing State-Space Models* section of this chapter for information about creating and manipulating these system models.

Table 2-1 shows the equations for the different forms of dynamic system models.

Table 2-1. Definitions of Continuous and Discrete Systems

Model Form	Continuous	Discrete
Transfer Function	$H(s) = \dfrac{b_0 + b_1 s + \ldots + b_{m-1}s^{m-1} + b_m s^m}{a_0 + a_1 s + \ldots + a_{n-1}s^{n-1} + a_n s^n}$ $\boldsymbol{H} = \left[H_{ij}\right]$	$H(z) = \dfrac{b_0 + b_1 z + \ldots + b_{m-1}z^{m-1} + b_m z^m}{a_0 + a_1 z + \ldots + a_{n-1}z^{n-1} + a_n z^n}$ $\boldsymbol{H} = \left[H_{ij}\right]$
Zero-Pole-Gain	$H(s) = \dfrac{k(s-z_1)(s-z_2)\ldots(s-z_m)}{(s-p_1)(s-p_2)\ldots(s-p_n)}$ $\boldsymbol{H} = \left[H_{ij}\right]$	$H(z) = \dfrac{k(z-z_1)(z-z_2)\ldots(z-z_m)}{(z-p_1)(z-p_2)\ldots(z-p_n)}$ $\boldsymbol{H} = \left[H_{ij}\right]$
State-Space	$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu}$ $\boldsymbol{y} = \boldsymbol{Cx} + \boldsymbol{Du}$	$\boldsymbol{x}(k+1) = \boldsymbol{Ax}(k) + \boldsymbol{Bu}(k)$ $\boldsymbol{y}(k) = \boldsymbol{Cx}(k) + \boldsymbol{Du}(k)$

Note Continuous models use the s variable to define time, whereas discrete models use the z variable.

You can use these forms to describe single-input single-output (SISO), single-input multiple-output (SIMO), multiple-input single-output (MISO), and multiple-input multiple-output (MIMO) systems. The number of sensors and actuators determines whether a dynamic system is a SISO, SIMO, MISO, or MIMO system.

The following sections provide information about an example dynamic system and how to represent this dynamic system using all three model forms.

RLC Circuit Example

Figure 2-2 shows an example circuit consisting of a resistor *R*, an inductor *L*, a current $i(t)$, a capacitor *C*, a capacitor voltage v_c, and an input voltage v_i.

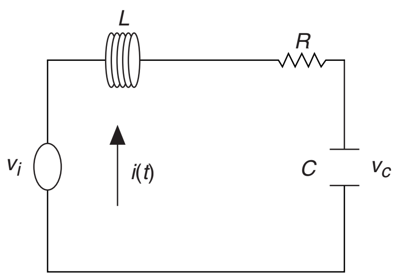

Figure 2-2. RLC Circuit

The following sections use this example to illustrate the creation of three forms of dynamic system models.

Constructing Transfer Function Models

Transfer function models use polynomial functions to define the dynamic relationship between inputs and outputs of a system. You analyze transfer function models in the frequency domain. The following equations define continuous and discrete transfer function models.

Continuous Transfer Function Model

$$H(s) = \frac{numerator(s)}{denominator(s)} = \frac{b_0 + b_1 s + \ldots + b_{m-1}s^{m-1} + b_m s^m}{a_0 + a_1 s + \ldots + a_{n-1}s^{n-1} + a_n s^n}$$

Discrete Transfer Function Model

$$H(z) = \frac{numerator(z)}{denominator(z)} = \frac{b_0 + b_1 z + \ldots + b_{m-1}z^{m-1} + b_m z^m}{a_0 + a_1 z + \ldots + a_{n-1}z^{n-1} + a_n z^n}$$

Numerators of transfer function models describe the locations of the zeroes of the system. Denominators of transfer function models describe the locations of the poles of the system.

Use the CD Construct Transfer Function Model VI to create continuous SISO, SIMO, MISO, and MIMO system models in transfer function form. This VI creates a data structure that defines the transfer function model and contains additional information about the system, such as the sampling time, input or output delays, and input and output names. Refer to the *Obtaining Model Information* section of this chapter for information about other properties of transfer function models.

SISO Transfer Function Models

Using the example in the *RLC Circuit Example* section of this chapter, you can describe the voltage of the capacitor v_c using the following second order differential equation:

$$LC\ddot{v}_c + RC\dot{v}_c + v_c = v_i$$

After taking the Laplace transform and rearranging terms, you then can write the transfer function between the input voltage V_i and the capacitor voltage V_c using the following equation.

$$\frac{V_c(s)}{V_i(s)} = \frac{\frac{1}{LC}}{s^2 + \frac{Rs}{L} + \frac{1}{LC}} = H(s)$$

You then can use H(s) to study the dynamic properties of the RLC circuit. The following equation defines a continuous transfer function where $R = 20\ \Omega$, $L = 50$ mH, and $C = 10\ \mu$F.

$$H(s) = \frac{2 \times 10^6}{s^2 + 400s + 2 \times 10^6}$$

Figure 2-3 shows how you use the CD Construct Transfer Function Model VI to create this continuous transfer function model.

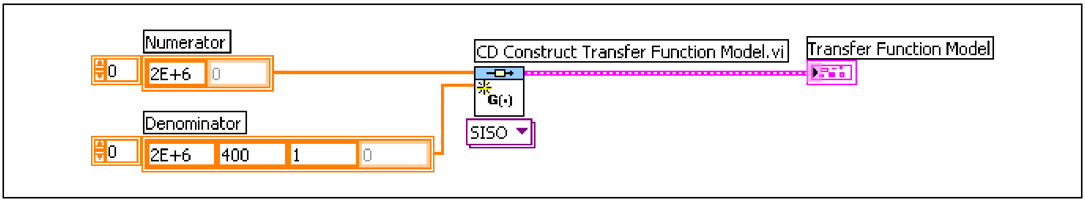

Figure 2-3. Creating a Continuous Transfer Function Model

The **Numerator** and **Denominator** inputs are arrays with zero-based indexes. The i^{th} element of the array corresponds to the i^{th} order coefficient of the polynomial. You define the coefficients in ascending order.

Note The CD Construct Transfer Function Model VI does not automatically cancel polynomial roots appearing in both the numerator and the denominator of the transfer function. Refer to Chapter 10, *Model Order Reduction*, for information about canceling pole-zero pairs.

The CD Construct Transfer Function Model VI creates a continuous model. You can create a discrete transfer function model in one of two ways. The method you use depends on whether you know the coefficients of the discrete transfer function model.

If you know the coefficients of the discrete transfer function model, you can enter in the appropriate values for **Numerator** and **Denominator** and set the **Sampling Time (s)** to a value greater than zero. Figure 2-4 shows this process using a sampling time of 10 μs.

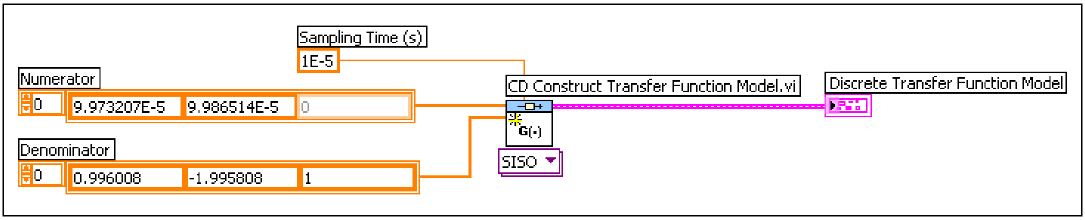

Figure 2-4. Using Coefficients to Create a Discrete Transfer Function Model

If you do not know the coefficients of the discrete transfer function model, you must use the CD Convert Continuous to Discrete VI for the conversion. Set the **Sampling Time (s)** parameter of this VI to a value greater than zero. Figure 2-5 shows this process using a sampling time of 10 μs.

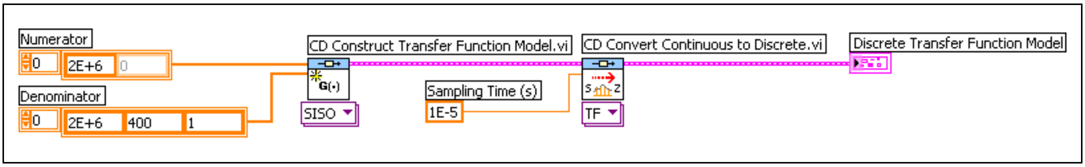

Figure 2-5. Using the CD Convert Continuous to Discrete VI to Create a Discrete Transfer Function Model

Converting from a continuous model to a discrete model results in the following equation:

$$H(z) = \frac{9.9865 \times 10^{-5} z + 9.9732 \times 10^{-5}}{z^2 - 1.9958z + 0.996}$$

Refer to the *Converting Continuous Models to Discrete Models* section of Chapter 3, *Converting Models*, for more information about converting continuous models to discrete models.

SIMO, MISO, and MIMO Transfer Function Models

You can use the CD Construct Transfer Function Model VI to create SIMO, MISO, and MIMO dynamic system models. This section uses a MIMO dynamic system model as an example.

Consider the two-input two-output system shown in Figure 2-6.

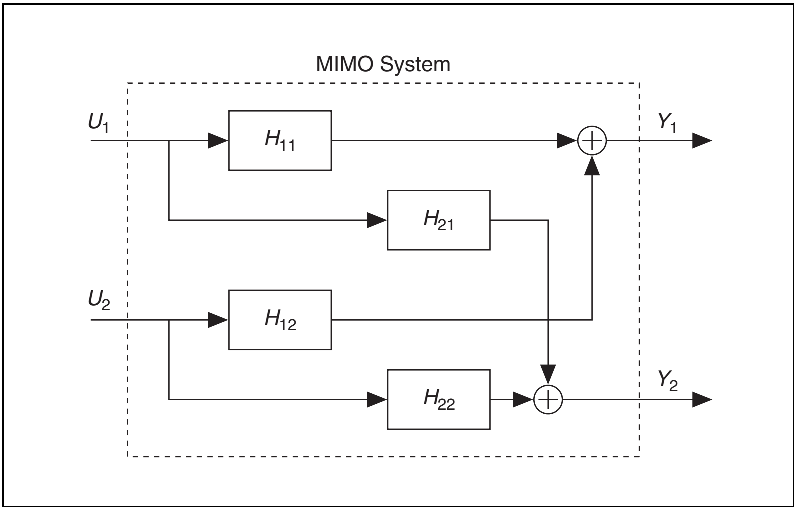

Figure 2-6. MIMO System with Two Inputs and Two Outputs

You can define the transfer function of this MIMO system by using the following transfer function matrix $\boldsymbol{H}$, where each element represents a SISO transfer function.

$$\boldsymbol{H} = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}$$

Suppose the following equations define the SISO transfer functions between each input-output pair.

$$H_{11} = \frac{1}{s} \qquad H_{12} = \frac{2}{s+1}$$

$$H_{21} = \frac{s+3}{s^2+4s+6} \qquad H_{22} = 4$$

Select the MIMO instance of the CD Construct Transfer Function Model VI to create a MIMO transfer function model. You then can specify each transfer function between the j^{th} input and the i^{th} output as the ij^{th} element of the two-dimensional **Transfer Function(s)** input array. Figure 2-7 shows that the numerator-denominator pair of the first row and first column corresponds to H_{11}, the numerator-denominator pair of the first row and second column corresponds to H_{12}, and so on.

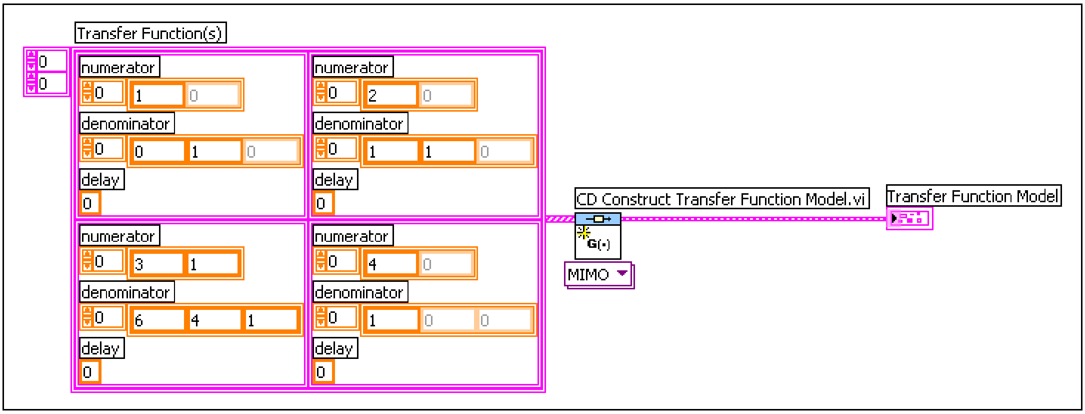

Figure 2-7. Creating a MIMO Transfer Function Model

The elements in the **numerator** and **denominator** arrays correspond to the coefficients, in ascending order, of the numerator and denominator in the H_{ij} transfer function model. For example, the numerator of H_{11} is 1, which corresponds to the zero-order coefficient. Therefore, the first element in the **numerator** array for H_{11} is 1. The denominator of H_{11} is s, which means the value 0 corresponds to the zero-order coefficient and the value 1 corresponds to the first-order coefficient. Therefore the first element in the **denominator** array for H_{11} is 0 and the second element is 1.

Symbolic Transfer Function Models

Symbolic models define the transfer function using variables rather than numerical values. If you want to change the value of R, for example, you only need to make the change in one location instead of several locations. Select the SISO (Symbolic) or MIMO (Symbolic) instance of the CD Construct Transfer Model VI to create a SISO or MIMO symbolic transfer function model, respectively.

The following equation is a symbolic version of the transfer function originally defined in the *SISO Transfer Function Models* section of this chapter.

$$H(s) = \frac{\frac{1}{LC}}{s^2 + \frac{Rs}{L} + \frac{1}{LC}}$$

Specify the **Symbolic Numerator** and **Symbolic Denominator** coefficients using the variable names *R*, *L*, and *C*. You then specify values of the numerator and denominator coefficients in the **variables** input, as shown in Figure 2-8.

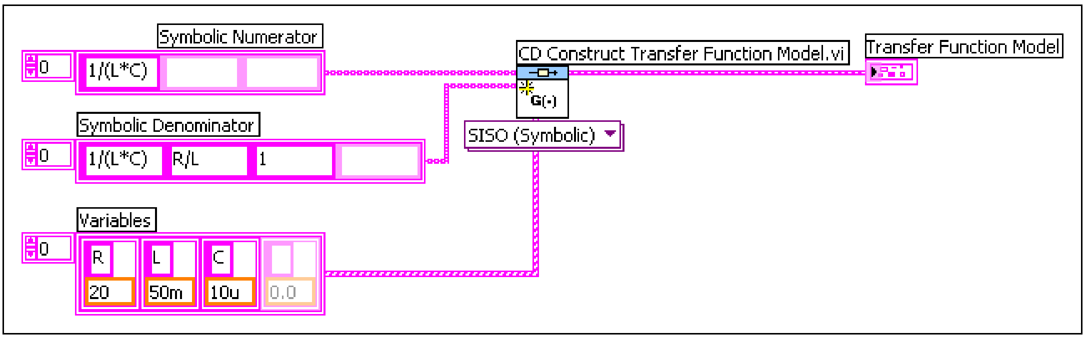

Figure 2-8. Creating a SISO Symbolic Transfer Function Model

Constructing Zero-Pole-Gain Models

Zero-pole-gain models are rewritten transfer function models. When you factor the polynomial functions of a transfer function model, you get a zero-pole-gain model. This factoring process shows the gain and the locations of the poles and zeroes of the system. The locations of these poles determine the stability of the dynamic system.

You analyze zero-pole-gain models in the frequency domain. The following equations define continuous and discrete zero-pole-gain models, where the numerators and denominators are products of first-order polynomials.

Continuous Zero-Pole-Gain Model

$$H_{ij}(s) = k\frac{\prod_{i=0}^{m} s + z_i}{\prod_{i=0}^{n} s + p_i} = \frac{k(s - z_1)(s - z_2)\dots(s - z_m)}{(s - p_1)(s - p_2)\dots(s - p_n)}$$

Discrete Zero-Pole-Gain Model

$$H_{ij}(z) = k\frac{\prod_{i=0}^{m} z + z_i}{\prod_{i=0}^{n} z + p_i} = \frac{k(z - z_1)(z - z_2)\dots(z - z_m)}{(z - p_1)(z - p_2)\dots(z - p_n)}$$

In these equations, k is a scalar quantity that represents the gain, z_i represents the locations of the zeroes, and p_i represents the locations of the poles of the system model.

Numerators of zero-pole-gain models describe the location of the zeroes of the system. Denominators of zero-pole-gain models describe the location of the poles of the system.

Use the CD Construct Zero-Pole-Gain Model VI to create SISO, SIMO, MISO, and MIMO system models in zero-pole-gain form. This VI creates a data structure that defines the zero-pole-gain model and contains additional information about the system, such as the sampling time, input

or output delays, and input and output names. Refer to the *Obtaining Model Information* section of this chapter for information about other properties of zero-pole-gain models.

SISO Zero-Pole-Gain Models

Using the example in the *RLC Circuit Example* section of this chapter, the following equation defines a continuous zero-pole-gain model where $R = 20\ \Omega$, $L = 50$ mH, and $C = 10\ \mu$F.

$$H(s) = \frac{2 \times 10^6}{(s + 200 + 1400i)(s + 200 - 1400i)} = \frac{2 \times 10^6}{(s + 200 \pm 1400i)}$$

This equation defines a model with one pair of complex conjugate poles at $-200 \pm 1400i$.

Figure 2-9 shows how you use the CD Construct Zero-Pole-Gain Model VI to create this continuous zero-pole-gain model.

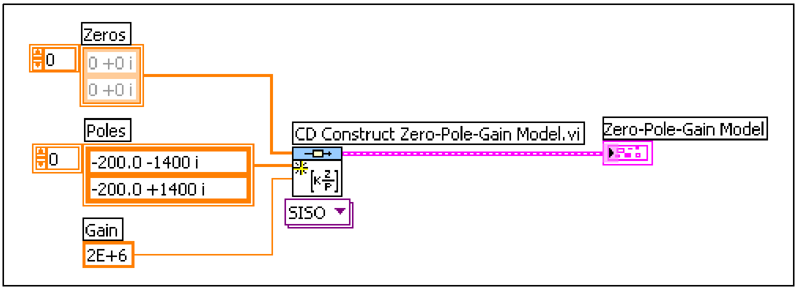

Figure 2-9. Creating a Continuous Zero-Pole-Gain Model

The CD Construct Zero-Pole-Gain Model VI creates a continuous model. You create a discrete zero-pole-gain model in the same way you create a discrete transfer function model. Refer to the *SISO Transfer Function Models* section of this chapter for more information about creating a discrete zero-pole-gain model.

SIMO, MISO, and MIMO Zero-Pole-Gain Models

You create SIMO, MISO, and MIMO zero-pole-gain models the same way you create SIMO, MISO, and MIMO transfer function models. Refer to the *SIMO, MISO, and MIMO Transfer Function Models* section of this chapter for information about creating these forms of system models.

Symbolic Zero-Pole-Gain Models

You create symbolic zero-pole-gain models the same way you create symbolic transfer function models. Refer to the *Symbolic Transfer Function Models* section of this chapter for information about creating a symbolic system model.

Constructing State-Space Models

Continuous state-space models use first-order differential equations to describe the system. Discrete state-space models use difference equations to describe the system. You analyze state-space models in the time domain. The following equations define a continuous and a discrete state-space model.

Continuous State-Space Model

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$
$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$$

Discrete State-Space Model

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)$$
$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k)$$

Table 2-2 describes the dimensions of the vectors and matrices of a state-space model.

Table 2-2. Dimensions and Names of State-Space Model Variables

Variable	Dimension	Name
k	—	discrete time
n	—	number of states
m	—	number of inputs
r	—	number of outputs
$\boldsymbol{A}$	$n \times n$ matrix	state matrix
$\boldsymbol{B}$	$n \times m$ matrix	input matrix
$\boldsymbol{C}$	$r \times n$ matrix	output matrix
$\boldsymbol{D}$	$r \times m$ matrix	direct transmission matrix

Table 2-2. Dimensions and Names of State-Space Model Variables (Continued)

Variable	Dimension	Name
$\boldsymbol{x}$	n-vector	state vector
$\boldsymbol{u}$	m-vector	input vector
$\boldsymbol{y}$	r-vector	output vector

Use the CD Construct State-Space Model VI to create SISO, SIMO, MISO, and MIMO system models in state-space form. This VI creates a data structure that uses matrices to define the state-space model. The matrices are zero-based two-dimensional arrays of numbers where the ij^{th} element of the array corresponds to the ij^{th} element of matrices in a state-space model. You can assume that an n^{th} order system with m inputs and r outputs has state, input, and output vectors as defined in the following equations:

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{m-1} \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{r-1} \end{bmatrix}$$

State-space models also contain additional information about the system, such as the sampling time, input or output delays, and input and output names. Refer to the *Obtaining Model Information* section of this chapter for information about other properties that state-space models contain.

SISO State-Space Models

Using the example in the *RLC Circuit Example* section of this chapter, the following equations define a continuous state-space model.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{v}_c \\ \ddot{v}_c \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} v_c \\ \dot{v}_c \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{LC} \end{bmatrix} v_i$$

$$\boldsymbol{y} = v_c = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v_c \\ \dot{v}_c \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} v_i$$

In these equations, $\boldsymbol{y}$ equals the voltage of the capacitor v_c, and u equals the input voltage v_i.

x equals the voltage of the capacitor and the derivative of that voltage $\begin{bmatrix} v_c \\ \dot{v}_c \end{bmatrix}$.

The following matrices define a state-space model where $R = 20\ \Omega$, $L = 50$ mH, and $C = 10\ \mu$F.

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -2 \times 10^6 & -400 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ 2 \times 10^6 \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} 0 \end{bmatrix}$$

When you plug these matrices into the equations for a continuous state space model defined in the *Constructing State-Space Models* section of this chapter, you get the following equations:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ -2 \times 10^6 & -400 \end{bmatrix} \begin{bmatrix} v_c \\ \dot{v}_c \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \times 10^6 \end{bmatrix} v_i$$

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v_c \\ \dot{v}_c \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} v_i$$

Figure 2-10 shows how you use the CD Construct State-Space Model VI to create this continuous state-space model.

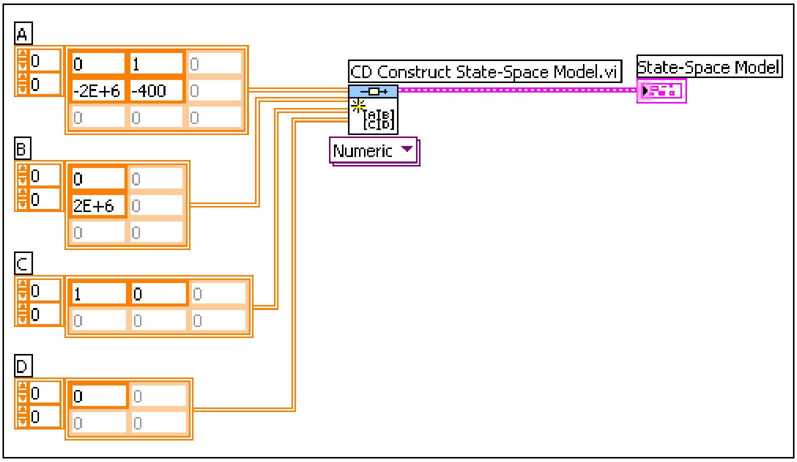

Figure 2-10. Creating a Continuous State-Space Model

Note Although ***B*** is a column vector, ***C*** is a row vector, and ***D*** is a scalar**,** you must use the 2D array data type when connecting these inputs to the VI.

The CD Construct State-Space Model VI creates a continuous model. You create a discrete state-space model in the same way you create a discrete transfer function model. Refer to the *SISO Transfer Function Models* section of this chapter for more information about creating a discrete state-space model.

SIMO, MISO, and MIMO State-Space Models

You construct a SIMO, MISO, or MIMO state-space model by ensuring the output matrix ***C*** and the input matrix ***B*** have the appropriate dimensions. For a SIMO system, construct an output matrix ***C*** with more than one row. For a MISO system, construct an input matrix ***B*** with more than one column. For a MIMO system, construct matrices ***C*** and ***B*** with more than one row and column, respectively.

When you create a SIMO, MISO, or MIMO system, ensure that the direct transmission matrix ***D*** has the appropriate dimensions. If you leave ***D*** empty or unwired, the Control Design Toolkit replaces the missing values with zeroes.

Symbolic State-Space Models

You create symbolic state-space models the same way you create a symbolic transfer function model. Refer to the *Symbolic Transfer Function Models* section of this chapter for more information about creating a symbolic system model.

Obtaining Model Information

Each Model Construction VI creates not only a data structure that defines the model, but also a set of properties that provide information about the system. These properties are common in all three model forms. Table 2-3 lists the properties and their corresponding data types.

Table 2-3. Model Properties

Property	Data Type	Description
Model Name	String	Assigns a name to a specific model.
Input Names	1D array of strings	The i^{th} element of the array defines the name of the i^{th} input to the model.
Output Names	1D array of strings	The i^{th} element of the array defines the name of the i^{th} output of the model.
Input Delays	1D array of double-precision, floating-point numeric values	The i^{th} element of the array defines the time delay of the i^{th} input of the model.
Output Delays	1D array of double-precision, floating-point numeric values	The i^{th} element of the array defines the time delay of the i^{th} output of the model.
Transport Delay	1D array of double-precision, floating-point numeric values	The ij^{th} element of the array defines the time delay between the i^{th} output and j^{th} input of the model.
Notes	String	A string for storing additional data. The string can contain comments or other information that you want to store with the model.

Table 2-3. Model Properties (Continued)

Property	Data Type	Description
Sampling Time	Double-precision, floating-point numeric value	Represents the sampling time, in seconds, of the system. If a model represents a continuous system, the value of **Sampling Time** is zero. For discrete system models, the value must be greater than zero.
State Names	Array of strings	The i^{th} element of the array defines the name of the i^{th} state of the model. This property is available with state-space models only.

You can use these data structures with every VI in the Control Design Toolkit that accepts a system model as an input.

Note Delay information exists in the model properties and not in the mathematical model. Any analysis, such as time- or frequency-domain analysis, you perform on the model does not account for delay present in the model. If you want the analysis to account for delay present in the model, you must incorporate the delay into the model itself. Refer to Chapter 5, *Working with Delay Information*, for more information about accounting for model delay.

You can use the Model Information VIs to get and set various properties of the model. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for more information about using the Model Information VIs to view and change the properties of a system model.

3

Converting Models

Model conversion involves changing the representation of dynamic system models. For example, you can convert a zero-pole-gain model to a state-space model. You also can convert a model between continuous and discrete types.

You can convert models you created using the LabVIEW Simulation Module into models you can use in the LabVIEW Control Design Toolkit and vice versa. Refer to the *LabVIEW Help*, available by selecting **Help» VI, Function, & How-To Help**, for more information about the VIs you can use to perform this conversion.

This chapter provides information about using the LabVIEW Control Design Toolkit to convert between model forms and to convert between continuous and discrete models.

Converting between Model Forms

You can use three different model forms—transfer function, zero-pole-gain, and state-space—to describe the same dynamic system. Refer to Chapter 2, *Constructing Dynamic System Models*, for more information about these model forms. You can use the Control Design Toolkit to convert from one form to another.

Converting between model forms is important because each form provides different information about the system. For example, state-space models use the states of a system to show physical information about the system. Thus, observing physical information about a dynamic system is less complicated when the model for that dynamic system is in state-space form.

You also can use different analysis and synthesis techniques depending on the form of the model. For example, if a model for a system is in transfer function form, you can synthesize a controller for that system using classical control design techniques such as the root locus technique. If the model is in state-space form, you can design a controller using state-space control design techniques such as the pole placement technique. Refer to Chapter 11, *Designing Classical Controllers*, and Chapter 12, *Designing*

State-Space Controllers, for more information about classical and state-space control design techniques.

The following sections discuss the Model Conversion VIs you can use to convert between model forms.

Converting Models to Transfer Function Models

Use the CD Convert to Transfer Function Model VI to convert a zero-pole-gain or state-space model to a transfer function model. This section uses a state-space model as an example.

Note Because transfer function models do not include state information, you lose the state vector $\boldsymbol{x}$ when you convert a state-space model to a transfer function model. Additionally, the Control Design Toolkit might not be able to recover the same states if you convert the model back to state-space form.

Consider the continuous state-space model defined in the *Constructing State-Space Models* section of Chapter 2, *Constructing Dynamic System Models*.

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{Ax} + \boldsymbol{Bu} \\ \boldsymbol{y} &= \boldsymbol{Cx} + \boldsymbol{Du} \end{aligned}$$

For continuous systems, you can use the Laplace transform to convert from the time domain to the Laplace domain model representation.

Note The equations in this section convert model forms within both the continuous and discrete domains. Refer to the *Converting between Continuous and Discrete Models* section of this chapter for information about converting between continuous and discrete domains.

Applying the Laplace transform to the state-space model results in the following equation:

$$Y(s) = [\boldsymbol{C}(\boldsymbol{I}s - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D}]U(s)$$

In this equation, s is the Laplace variable, and $\boldsymbol{I}$ is the identity matrix with the same dimensions as $\boldsymbol{A}$.

The ratio between the output $Y(s)$ and input $U(s)$ defines the following matrix transfer function model $H(s)$.

$$H(s) \equiv \frac{Y(s)}{U(s)} = \boldsymbol{C}(\boldsymbol{I}s - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D}$$

For example, consider the following second-order MISO state-space system model.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -1 & 2 \\ 0 & -1 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\boldsymbol{u}$$

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 0 & 0 \end{bmatrix}\boldsymbol{u}$$

Using the Laplace transform, you obtain the transfer function matrix $H(s)$.

$$H(s) = \begin{bmatrix} \frac{1}{s+1} & \frac{2}{s^2 + 2s + 1} \end{bmatrix}$$

Converting Models to Zero-Pole-Gain Models

Use the CD Convert to Zero-Pole-Gain Model VI to convert a transfer function or state-space model to a zero-pole-gain model. This section uses a transfer function model as an example.

Note When you convert a state-space model to a zero-pole-gain model, the CD Convert to Zero-Pole-Gain Model VI converts the state-space model to a transfer function model first.

To convert the transfer function matrix $H(s)$ to the zero-pole-gain form, the Control Design Toolkit calculates the numerator and denominator polynomial roots and the gain of each SISO transfer function in $H(s)$.

When you convert the transfer function matrix from the *Converting Models to Transfer Function Models* section of this chapter, you obtain the following zero-pole-gain model:

$$H(s) = \begin{bmatrix} \frac{1}{s+1} & \frac{2}{(s+1)^2} \end{bmatrix}$$

This zero-pole-gain model is numerically identical to the transfer function model. The zero-pole-gain form, however, shows the locations of the zeroes and poles of a system.

Converting Models to State-Space Models

Use the CD Convert to State-Space Model VI to convert a zero-pole-gain or transfer function model to a state-space model. This section uses a zero-pole-gain model as an example.

Note When you convert a zero-pole-gain model to a state-space model, the CD Convert to State-Space Model VI converts the zero-pole-gain model to a transfer function model first.

When converting a transfer function or zero-pole-gain model, you can specify whether you want the resulting state-space model to be full or minimal. A full state-space model does not reduce the number of states determined by a least common denominator calculation. A minimal state-space model reduces the number of states and produces a minimal representation of the original model. Use the **Realization Type** parameter of the CD Convert to State-Space Model VI to specify if you want the resulting model to be full or minimal. Refer to the *Obtaining the Minimal Realization of Models* section of Chapter 10, *Model Order Reduction*, for more information about minimizing state-space realizations.

Using the example in the *Converting Models to Transfer Function Models* section of this chapter, the following equation gives the minimal realization when converting a zero-pole-gain model to a state-space model.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -0.33 & 0.94 \\ -0.47 & -1.67 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} -0.41 & 0 \\ 0.29 & -0.87 \end{bmatrix} \boldsymbol{u}$$

$$\boldsymbol{y} = \begin{bmatrix} -2.45 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 & 0 \end{bmatrix} \boldsymbol{u}$$

This model numerically differs from the initial state-space model. From the input-output model perspective, however, the state-space models are identical.

Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for more information about the Model Conversion VIs.

Converting between Continuous and Discrete Models

Continuous models are analog and operate using physical components. Discrete models are digital and operate on a computer or real-time (RT) target. To determine how an analog model performs on a digital target, you can convert the continuous model to a discrete model. You also can convert a discrete model to a continuous model.

Additionally, you can resample a discrete model. Resampling involves converting a discrete model to a discrete model with a different sampling time. Resampling is useful when the sampling time of a model does not match the sampling time of the target on which that model operates. In this situation, you resample the model to use the sampling time of the target.

The Model Conversion VIs provide a number of mathematical methods that perform these conversions. Table 3-1 summarizes these methods, which are substitutions between the continuous Laplace-transform operator and the discrete z-transform operator.

Table 3-1. Mapping Methods for Converting between Continuous and Discrete

Method of Approximation	Continuous to Discrete	Discrete to Continuous
Forward Rectangular Method	$s \rightarrow \frac{z-1}{T}$	$z \rightarrow 1 + sT$
Backward Rectangular Method	$s \rightarrow \frac{z-1}{zT}$	$z \rightarrow \frac{1}{1-sT}$
Tustin's Method	$s \rightarrow \frac{2(z-1)}{T(z+1)}$	$z \rightarrow \frac{1+\frac{sT}{2}}{1-\frac{sT}{2}}$
Prewarp Method	$s \rightarrow \frac{z(z-1)}{T^*(z+1)}$ $T^* = \frac{2\tan\left(\frac{w \times T}{2}\right)}{w}$	$z \rightarrow \frac{1+sT^*}{1-sT^*}$ $T^* = \frac{2\tan\left(\frac{w \times T}{2}\right)}{w}$

In these equations, T represents the sample time and w represents the prewarp frequency. T^* is a modified sample time that the Prewarp method uses in converting between continuous and discrete models.

The following sections provide information about the methods that you can use to perform continuous to discrete conversions, discrete to continuous conversions, and discrete to discrete conversions.

Converting Continuous Models to Discrete Models

To convert a continuous model to a discrete one, first approximate the value of the derivative in the continuous equation over each change in time. Then find the area of the geometric region having width *dt* and height equal to the derivative.

For example, consider the following first-order continuous differential equation:

$$\dot{y} = f(t)$$

To convert this continuous model to a discrete model, evaluate the derivative function $f(t)$ at different points to approximate $\dot{y}$ at time t. Figure 3-1 illustrates the function $f(t)$ between t and $t + T$, where T is the sampling time.

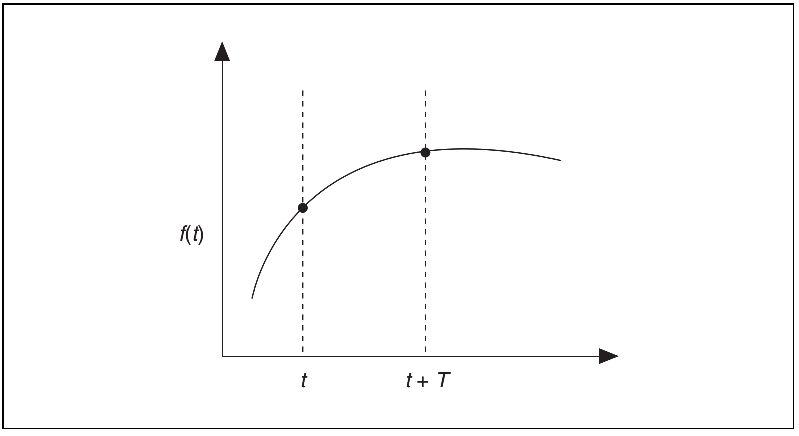

Figure 3-1. Discretizing a Differential Equation

Integrating between time t and $t + T$ results in the following difference equation:

$$\int_{t}^{t+T} \dot{y}\,d\tau = y(t+T) - y(t) = \int_{t}^{t+T} f(\tau)\,d\tau$$

Integrating $f(\tau)$ for $\tau = t$ to $t + T$ represents the area under the curve. The CD Convert Continuous to Discrete VI provides the following mathematical methods to approximate this area.

- Forward Rectangular
- Backward Rectangular
- Tustin's
- Prewarp
- Zero-Order-Hold
- First-Order-Hold

- Z Transform
- Matched Pole Zero

The following sections provide information about each of these methods.

Forward Rectangular Method

The Forward Rectangular method considers $f(\tau)$ constant and equal to $f(t + T)$ along the integration range. This consideration results in the following equation:

$$y(t+T) = y(t) + f(t+T)T$$

This method considers the incremental area term between sampling times t and $t + T$ as a rectangle of width T and height equal to $f(t + T)$, as shown in Figure 3-2.

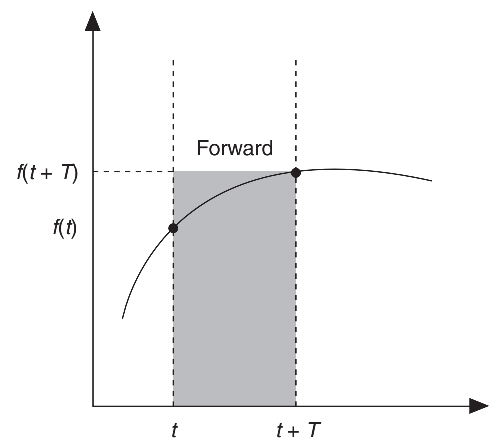

Figure 3-2. Forward Rectangular Method

Figure 3-2 shows that, for this example, the Forward Rectangular method overestimates the area under the curve. To minimize this overestimation, use a small sampling interval. Depending on the direction and size of the curve you are measuring, this overestimation might not occur.

Backward Rectangular Method

The Backward Rectangular method considers $f(\tau)$ constant and equal to $f(t)$ along the integration range. This consideration results in the following equation:

$$y(t+T) = y(t) + f(t)T$$

This method considers the incremental area term between sampling times t and $t + T$ as a rectangle of width T and height equal to $f(t)$, as shown in Figure 3-3.

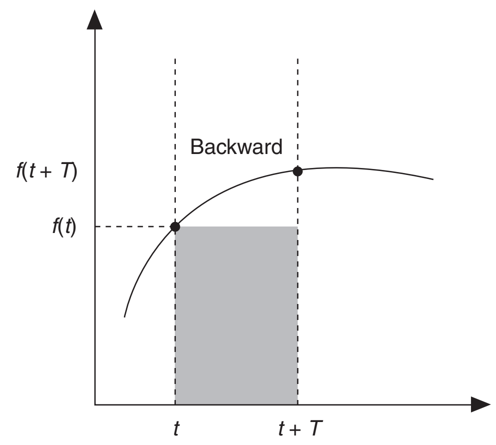

Figure 3-3. Backward Rectangular Method

Figure 3-3 shows that, for this example, the Backward Rectangular method underestimates the area under the curve. To minimize this underestimation, use a small sampling interval. Depending on the direction and size of the curve you are measuring, this underestimation might not occur.

Tustin's Method

Tustin's method, also known as the trapezoid method, uses trapezoids to provide a balance between the Forward Rectangular and Backward Rectangular methods. Tustin's method takes the average of the rectangles defined by the Forward and Backward Rectangular methods and uses the average value as the incremental area to approximate the area under the curve.

Tustin's method considers $f(\tau)$ constant and equal to the average between $f(t)$ and $f(t + T)$ along the integration range, which results in the following equation:

$$y(t+T) = y(t) + \frac{[f(t) + f(t+T)]}{2}T$$

The last term in this equation is identical to the area of a trapezoid of height T and bases $f(t)$ and $f(t + T)$. Figure 3-4 shows the area under a curve using Tustin's method.

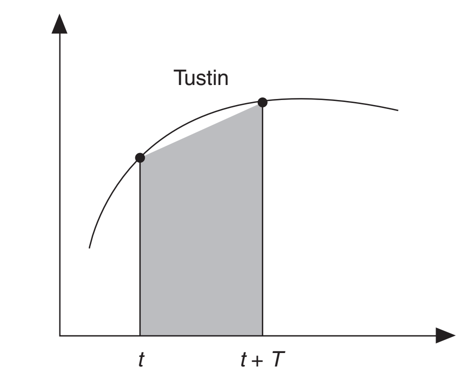

Figure 3-4. Tustin's Method

Figure 3-4 shows that, for this example, Tustin's method provides a balance between the overestimation of the Forward Rectangular and the underestimation of the Backward Rectangular method.

Prewarp Method

The Prewarp method is a trapezoidal type of transformation that uses the prewarp frequency ω to adjust the sampling time T. This adjustment results in a separate sampling time T^*. This adjustment also compensates for errors introduced in the discretizing process.

This method also considers $f(\tau)$ constant and equal to the average between $f(t)$ and $f(t + T^*)$ along the integration range, which results in the following equation:

$$y(t+T) = y(t) + \frac{[f(t) + f(t+T^*)]}{2}T$$

The last term in this equation is identical to the area of a trapezoid of height T and bases $f(t)$ and $f(t + T^*)$. Figure 3-5 shows the area under a curve using the Prewarp method.

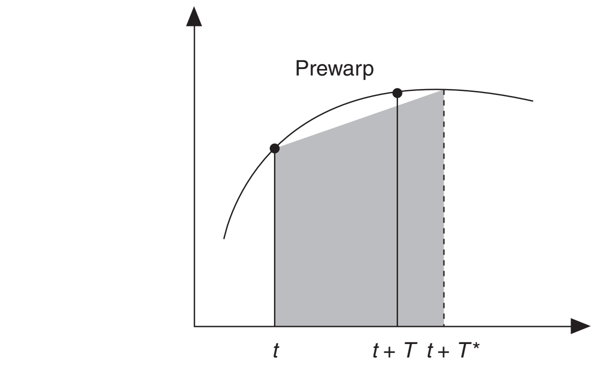

Figure 3-5. Prewarp Method

Figure 3-5 shows that, for this example, the Prewarp method compensates for the integration error by adjusting the sampling time to T^*. The area between $t + T$ and $t + T^*$ is roughly equal to the integration error, which is represented by the unshaded portion of the area under the curve.

Use a particular conversion method based on the model that you are converting and the requirements of the application for which you are designing a control system.

Zero-Order-Hold and First-Order-Hold Methods

The Zero-Order-Hold and First-Order-Hold methods assume properties of the continuous differential equation $\dot{y} = f(t)$. The Zero-Order-Hold method assumes that $f(t)$ consists of an input that you can hold constant during the integration period between sampling times t and $t + T$. The First-Order-Hold method assumes that you can increase this input over time during this same period. These methods also integrate the remaining terms of $f(t)$ not related to the input because these terms refer to the internal state dynamics.

You obtain the following equation after integrating a linear time-invariant system between sampling times t and $t + T$.

$$x(t+T) = e^{AT}x(t) + \int_t^{t+T} e^{A(t+T+\tau)}Bu(\tau)d\tau$$

$$y(t) = Cx(t) + Du(t)$$

In this equation, $u(t)$ is the input to the system and is not necessarily constant between sampling times t and $t + T$. The following equation shows the Zero-Order-Hold method approximating the input to a constant value $u(t)$ during the integration time.

$$x(t+T) = e^{AT}x(t) + \int_t^{t+T} e^{A(t+T+\tau)}Bd\tau\, u(t)$$

Conversely, the following equation shows the First-Order-Hold method ramping the input values with a constant slope $[u(t + T) - u(t)]/T$ during integration time.

$$x(t+T) = e^{AT}x(t) + \int_t^{t+T} e^{A(t+T+\tau)}B\left\{u(t) + [u(t+T) - u(t)]\frac{(\tau - t)}{T}\right\}d\tau$$

Refer to *Digital Control of Dynamic Systems*[1] for more information about the Zero-Order-Hold and First-Order-Hold methods.

Z Transform Method

The Z Transform method is defined such that the continuous and discrete impulse responses maintain major similarities. You calculate the impulse response of the discrete transfer function by multiplying the inverse Laplace transform of the continuous transfer function by the sampling time T.

Refer to *Discrete-Time Control Systems*[2] for more information about the Z Transform method.

[1] Franklin, Gene F., J. David Powell, and Michael L. Workman. *Digital Control of Dynamic Systems*, 3rd ed. Menlo Park, CA: Addison Wesley Longman, Inc., 1998.

[2] Ogata, Katsuhiko. *Discrete-Time Control Systems,* 2nd ed. Englewood Cliffs, N.J.: Prentice Hall, 1995.

Matched Pole Zero Method

The Matched Pole Zero method uses the following relationship between the continuous s and discrete z frequency domains.

$$z = e^{sT}$$

In this equation, T is the sampling time used for the discrete system. The Matched Pole Zero method maps continuous-time poles and finite zeroes to the z-plane using this relation. This method also maps zeroes at infinity to $z = 0$, so these zeroes do not affect the frequency response.

After the algorithm maps the poles and zeroes, the algorithm then attempts to make sure the system gains are equivalent at some critical frequency. If the systems have no poles or zeroes at $s = 0$ or $z = 1$, the Matched Pole Zero method selects a discrete-time gain such that the system gains match at these locations.

Alternatively, if the systems have no poles or zeroes at $s = p(i/T)$ or $z = -1$, where p is the location of a pole, this method equalizes the gains at that frequency. If the Matched Pole Zero method cannot match either of these gains, the algorithm does not choose a gain.

Refer to *Digital Control of Dynamic Systems*[1] for more information about the Matched Pole Zero method.

Converting Discrete Models to Continuous Models

Use the CD Convert Discrete to Continuous VI to convert a discrete model to a continuous model. This VI supports the following conversion methods: Forward Rectangular, Backward Rectangular, Tustin's, Prewarp, Z Transform, and Zero-Order-Hold. This VI does not support the First-Order-Hold or Matched Pole Zero methods. Refer to Table 3-1 for the equations for each mapping method.

The Z Transform method also is a reverse calculation to map a model in the z-plane to the s-plane. You calculate the impulse response of the continuous transfer function by dividing the inverse z-transform of the discrete transfer function by the sampling time T.

[1] Franklin, Gene F., J. David Powell, and Michael L. Workman. *Digital Control of Dynamic Systems*, 3rd ed. Menlo Park, CA: Addison Wesley Longman, Inc., 1998.

Resampling a Discrete Model

Use the CD Convert Discrete to Discrete VI to resample a discrete model. This VI converts the discrete model to a continuous model and then converts the continuous model back to a discrete model. The first conversion uses the initial sampling time T_1. The second conversion uses the final sampling time T_2.

The CD Convert Discrete to Discrete VI supports the following conversion methods: Forward Rectangular, Backward Rectangular, Tustin's, Prewarp, Zero-Order-Hold, and Z Transform. This VI does not support the First-Order-Hold or Matched Pole Zero methods.

4

Connecting Models

You typically create a dynamic system model by connecting many models, or subsystems, together. Connecting many models together makes developing a model of a complicated dynamic system less complicated because you can describe the dynamics of individual pieces.

You only can connect continuous models to other continuous models. To connect discrete models together, each model must have the same sampling time. Connected models might, however, be of any form. For example, you can connect a transfer function model to a state-space model or a state-space model to a zero-pole-gain model.

Furthermore, you can make connections between single-input single-output (SISO), single-input multiple-output (SIMO), multiple-input single-output (MISO), and multiple-input multiple-output (MIMO) systems.

This chapter provides information about using the LabVIEW Control Design Toolkit to connect models in the following four ways: in series, by appending, in parallel, and with feedback.

Connecting Models in Series

A series connection joins the outputs of the first model to the inputs of a second model. Use the CD Series VI to connect two models in series.

Note When connecting models of different forms, the **Series Model** output terminal returns a model based on the following hierarchy: state-space>transfer function> zero-pole-gain. For example, if you connect a zero-pole-gain model to a state-space model, **Series Model** returns a state-space model.

The following sections provide information about the kinds of connections you can make with the CD Series VI.

Connecting SISO Systems in Series

Consider a valve that controls the flow rate of water into a tank. Figure 4-1 represents this system.

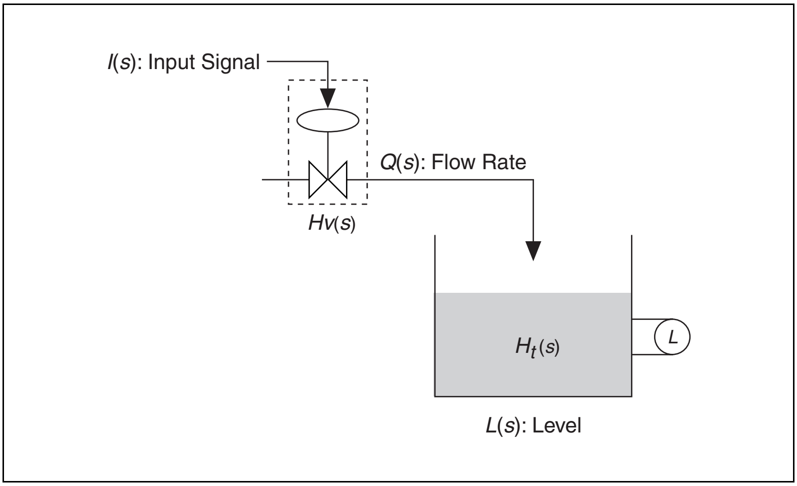

Figure 4-1. Flow of Water into a Tank

If you assume that the incoming water pressure to the valve is constant, only the valve input signal affects the level of the water in the tank. You can model the flow rate of water into the tank using the following transfer functions, where $H_v(s)$ is a model of the valve and $H_t(s)$ is a model of the tank.

$$H_v(s) \equiv \frac{Q(s)}{I(s)} = \frac{K_v}{\tau^2 s^2 + 2\zeta\tau s + 1} \qquad H_t(s) \equiv \frac{L(s)}{Q(s)} = \frac{K_t}{s}$$

$I(s)$, $Q(s)$, and $L(s)$ represent the Laplace transform of the input signal, the flow rate, and level of water in the tank, respectively. The constants K_v, τ, ζ, and K_t are parameters of the models that describe the valve and tank. To obtain the effect of the input signal on the water level, place the two systems in series and multiply their transfer functions.

$$H(s) \equiv \frac{L(s)}{I(s)} = H_v(s) \cdot H_t(s) = \frac{K_v}{\tau^2 s^2 + 2\zeta\tau s + 1} \cdot \frac{K_t}{s}$$

This equation represents the output of $H_v(s)$ connecting to the input of $H_t(s)$. Figure 4-2 illustrates this relationship.

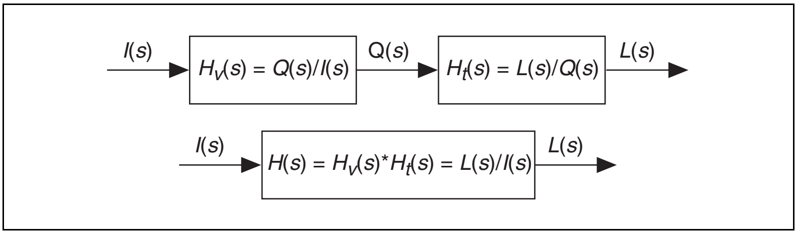

Figure 4-2. Valve Model and Tank Model in Series

The resulting SISO system $H(s)$ now represents the relationship between the input signal $I(s)$ and the level of water $L(s)$ in the tank.

Creating a SIMO System in Series

You can create a SIMO system by connecting two or more SISO systems with a SIMO subsystem. For example, adding another valve and tank to the example in the *Connecting SISO Systems in Series* section of this chapter results in a SIMO system that divides the flow rate between two different tanks. Figure 4-3 shows this system.

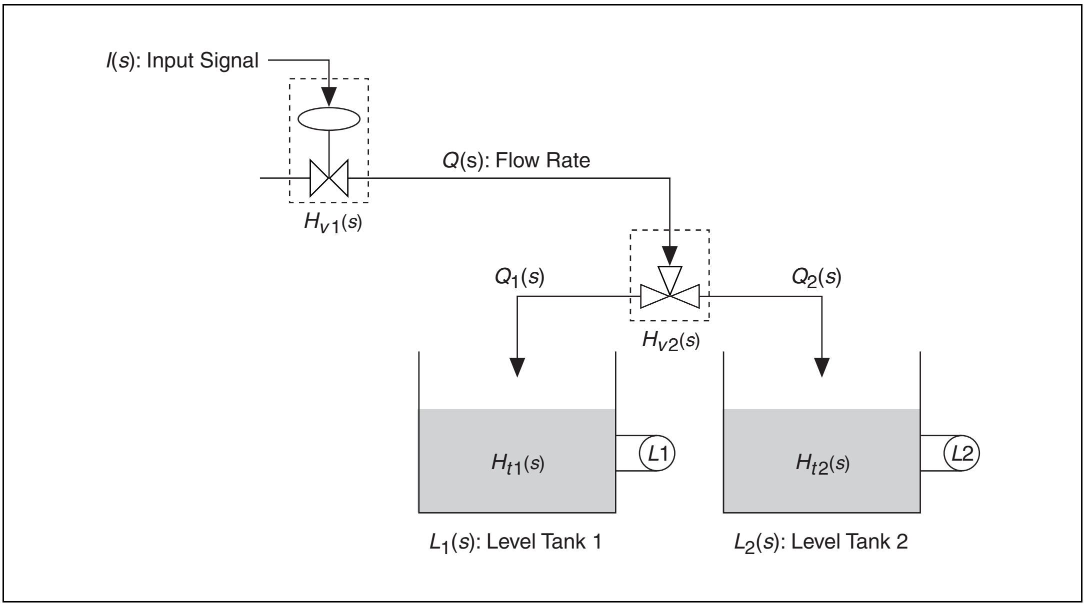

Figure 4-3. Dividing the Flow of Water between Two Tanks

$\boldsymbol{H}_{v2}(s)$ is a SIMO transfer function matrix that represents the relationship of the flow rates. By connecting $\boldsymbol{H}_{v2}(s)$ to H_{v1}(s) and Q(s), the entire system becomes SIMO. The total flow rate $Q(s)$ is equal to the sum of the parts $Q_1(s)$ and $Q_2(s)$.

$$Q(s) = Q_1(s) + Q_2(s) = \lambda Q(s) + (1 - \lambda)Q(s)$$

The constant λ represents the fraction of flow sent to the first tank, whereas (1 – λ) is the remaining fraction of flow sent to the second tank.

$$\boldsymbol{H}_{v2}(s) = \begin{bmatrix} \lambda \\ 1 - \lambda \end{bmatrix}$$

When you connect these models in series, the output of the first system $H_{v1}(s)$ connects to the input of the second system $\boldsymbol{H}_{v2}(s)$. Figure 4-4 illustrates this relationship.

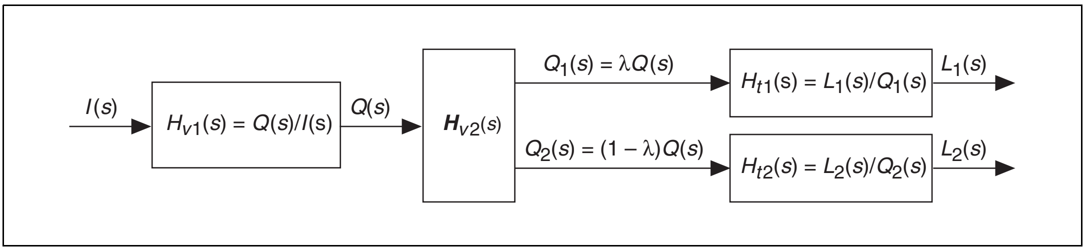

Figure 4-4. Two Valve Models and Two Tank Models in Series

This combined system, which is now a SIMO system, has one input $I(s)$ and two outputs $L_1(s)$ and $L_2(s)$. Figure 4-5 is a LabVIEW block diagram that illustrates this system.

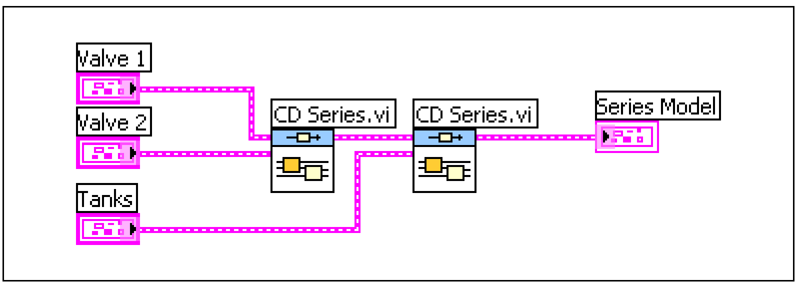

Figure 4-5. Block Diagram of the Two Valves and Tanks in Series

Connecting MIMO Systems in Series

When connecting MIMO systems, you can connect any output of the first model to any input(s) of the second model. Figure 4-6 shows an example of two MIMO system models connected in series.

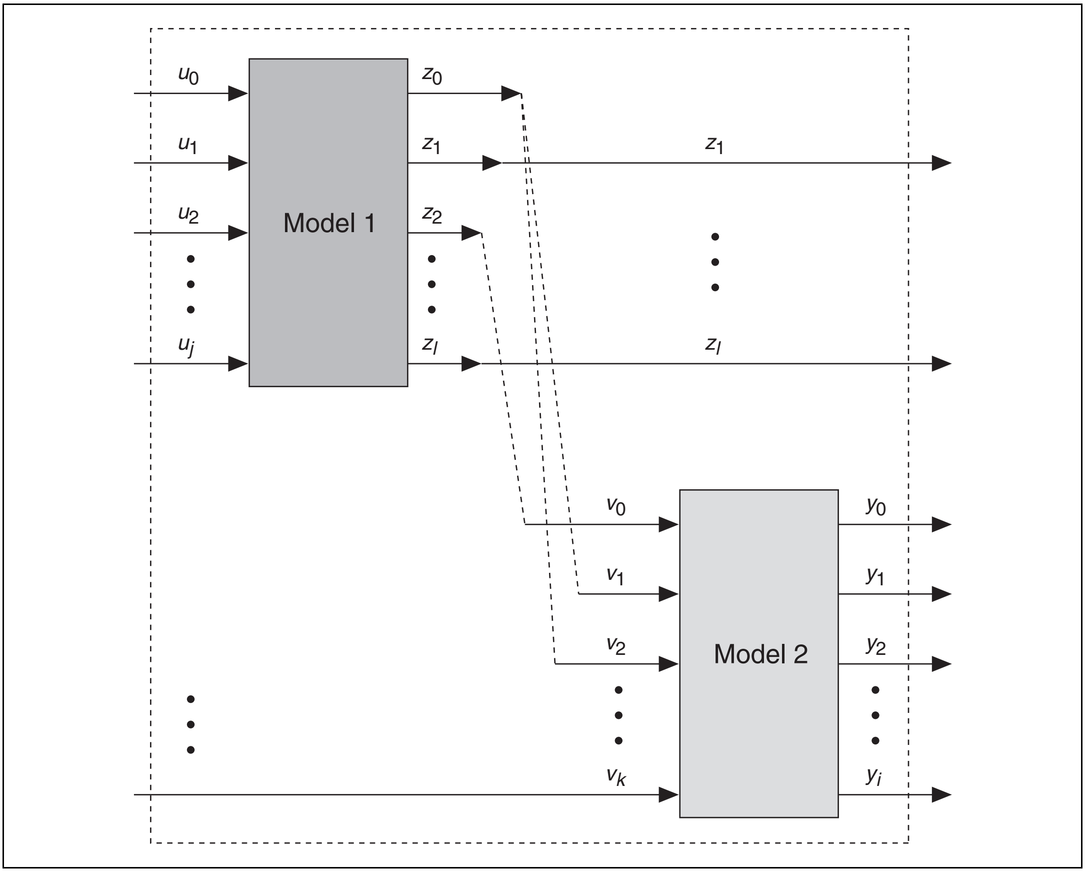

Figure 4-6. MIMO System Models in Series

Figure 4-6 shows how the outputs of Model 1 that are connected to the inputs of Model 2 do not appear as outputs of the resulting series model. For example, because z_0 connects to the Model 2 inputs v_1 and v_2, z_0 is no longer an output of the resulting series model. Similarly, because z_2 connects to v_0, z_2 is no longer an output of the resulting series model.

This same principle applies to the inputs of Model 2. Inputs of Model 2 that are connected to an output of Model 1 no longer appear as inputs of the resulting series model. Because the input v_0 of Model 2 is connected to the output of z_2 of Model 1, neither v_0 nor z_2 appear in the resulting series model.

You define the connections between two models using the **Connections** control of the CD Series VI. Figure 4-7 shows the settings this control used to connect the models in Figure 4-6.

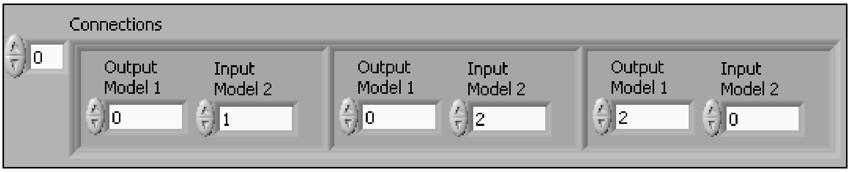

Figure 4-7. Connection Definitions for Models in Series

The control in Figure 4-7 indicates that the Model 1 output z_0 connects to the Model 2 inputs v_1 and v_2. You also can see how the Model 1 output z_2 connects to the Model 2 input v_0.

Appending Models

You can append models together to compare the time or frequency response of two models in the same plot. Use the CD Append VI to produce an augmented model from connections between two models. This augmented model contains all inputs and outputs of both models. With state-space models, states of the first model are combined with states of the second model.

Figure 4-8 shows two appended system models.

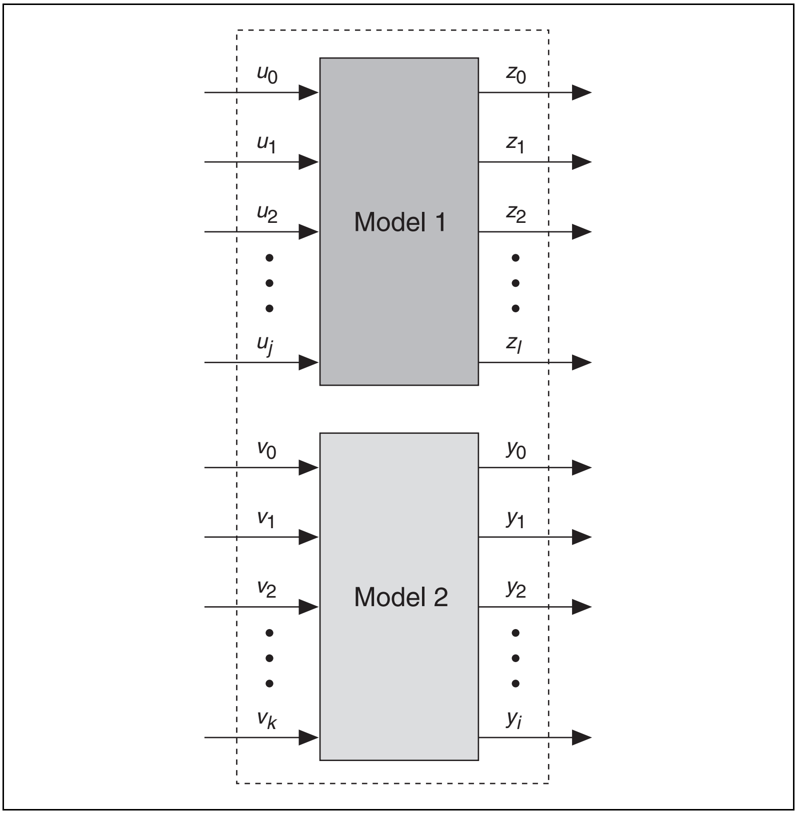

Figure 4-8. Appended Models

For example, consider the two tanks from the *Creating a SIMO System in Series* section of this chapter. The following equations define the transfer functions of the tanks.

$$H_{t1}(s) = \frac{K_1}{s} \qquad H_{t2}(s) = \frac{K_2}{s}$$

K_1 and K_2 are the gains of their respective transfer functions. Appending $H_{t1}(s)$ and $H_{t2}(s)$ results in the following appended matrix transfer function $\boldsymbol{H}_t$.

$$\boldsymbol{H}_t = \begin{bmatrix} H_{t1}(s) & 0 \\ 0 & H_{t2}(s) \end{bmatrix}$$

Figure 4-9 uses the block diagram from Figure 4-5 but replaces the **Tanks** input with $\boldsymbol{H_t}$. As in Figure 4-5, the two valves are connected in series with each other. In Figure 4-9, however, the two tanks now are appended to each other.

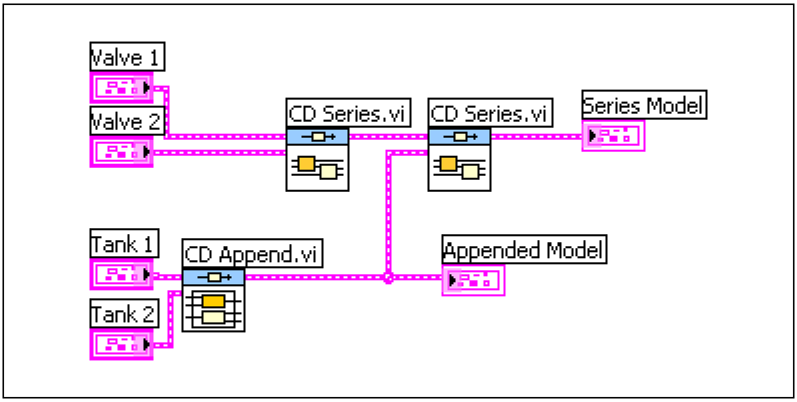

Figure 4-9. Appending the Two Tanks

Connecting Models in Parallel

A parallel connection creates a single model from two separate systems that share common inputs. You also can use a parallel connection to add or subtract outputs of two subsystems and represent them as a single output. Use the CD Parallel VI to connect systems in parallel.

For example, consider the circuit system in Figure 4-10.

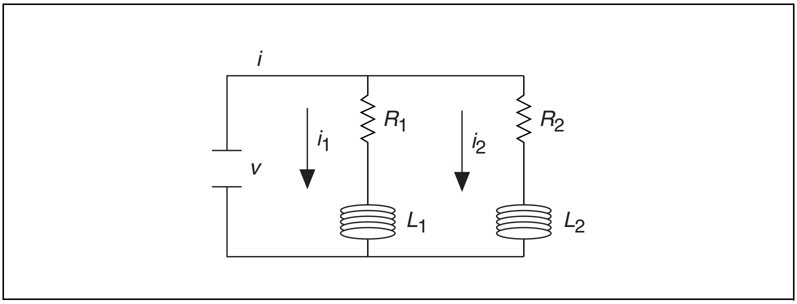

Figure 4-10. Circuit System

The input of this system is the voltage v. The output of this system is the total current i, which is the sum of currents i_1 and i_2. R_1 and R_2 are resistors, and L_1 and L_2 are inductors. The following equations describe the individual currents for the circuit system in Figure 4-10.

$$L_1\frac{di_1}{dt} + R_1 i_1 - v = 0$$

$$L_2\frac{di_2}{dt} + R_2 i_2 - v = 0$$

The following equations give the resulting transfer functions for each circuit loop.

$$H_1(s) = \frac{I_1(s)}{V(s)} = \frac{1}{L_1 s + R_1}$$

$$H_2(s) = \frac{I_2(s)}{V(s)} = \frac{1}{L_2 s + R_2}$$

In Figure 4-11, $H_1(s)$ and $H_2(s)$ represent the transfer functions defined in the previous equations, and $I_1(s)$ and $I_2(s)$ are the respective outputs of these transfer functions. $V(s)$ is the transfer function of the voltage input v that both circuit loops share.

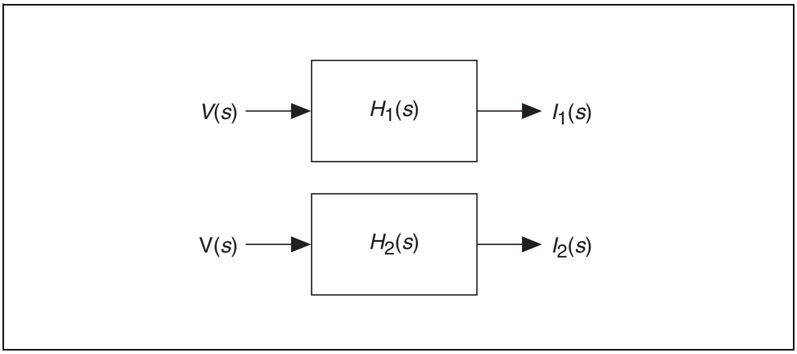

Figure 4-11. Each Circuit Loop in the Circuit System

Figure 4-12 illustrates the relationship between the voltage input *v* and total current *i* by placing both models together in one larger system model. When the two models are in parallel, both models share the same input $V(s)$ and provide a total output $I(s)$ as shown in Figure 4-12.

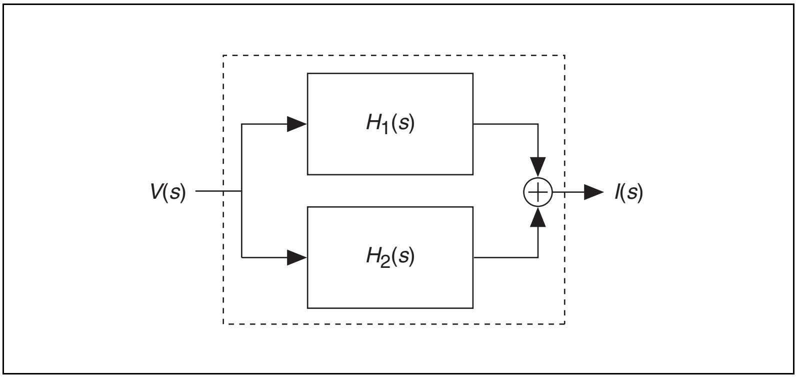

Figure 4-12. Entire Circuit System as a Parallel Model

The following equations describe the resulting transfer function as a second-order system.

$$I(s) = I_1(s) + I_2(s) = V(s)[H_1(s) + H_2(s)]$$

$$H(s) = \frac{I(s)}{V(s)} = H_1(s) + H_2(s)$$

Figure 4-13 illustrates how some inputs from Model 1 and Model 2 share the same inputs. The outputs of Model 1 are added to or subtracted from the outputs of Model 2 to provide one combined parallel model.

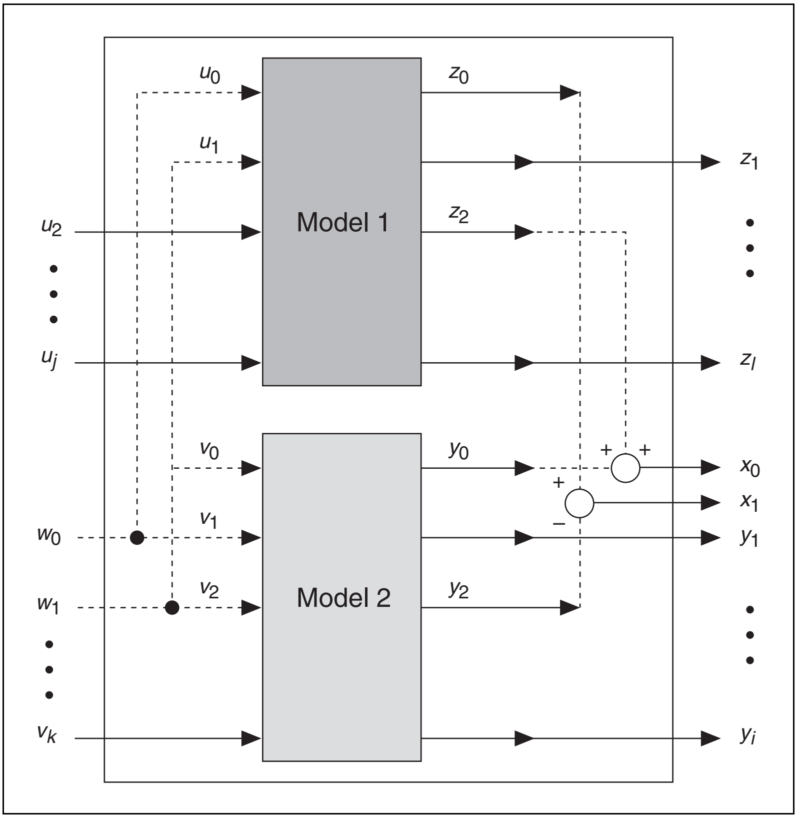

Figure 4-13. MIMO Models in Parallel

Use the CD Parallel VI to define the relationship between the inputs and outputs of the models. Figure 4-14 displays the **Input Connections** and

Output Connections controls that define the parallel interconnections shown in Figure 4-13.

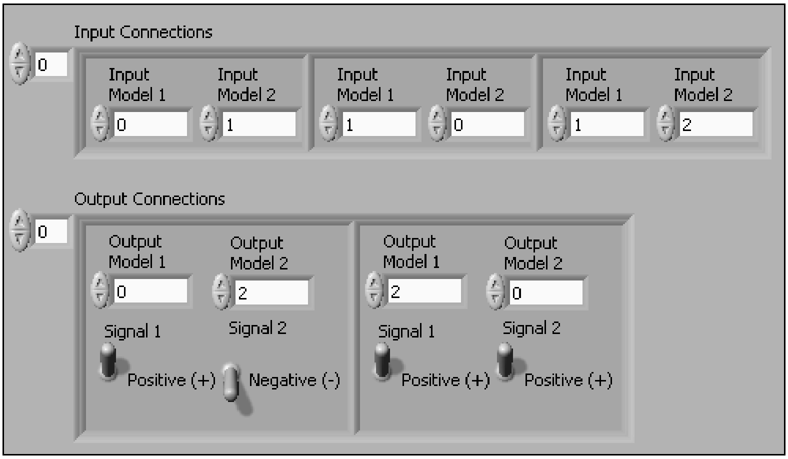

Figure 4-14. Connection Definitions for Models in Parallel

These controls indicate that the input for u_0 of Model 1 is the same as the input for v_1 of Model 2, the input for u_1 of Model 1 is the same as the input for v_0 of Model 2, and so on. You can see how the y_2 output of Model 2 is subtracted from the z_0 output of Model 1. You also can see how the z_2 output of Model 1 is added to the y_0 output of Model 2. You define addition and subtraction by specifying the output as a **Positive (+)** or **Negative (–)** connection.

In Figure 4-13, notice that any common inputs from the original models are replaced by a new input w_n in the resulting model. Likewise, any combined outputs of the original models are replaced by a new output x_n in the resulting model.

Placing Models in a Closed-Loop Configuration

Use the CD Feedback VI to place one or two models in a closed-loop configuration. The **Feedback Connections** and **Output Connections** parameters define the connections between the outputs of a model to the inputs of the same model or a second model. If the models have an unequal number of inputs and outputs, the CD Feedback VI establishes a number of connections equal to the smaller number of inputs or outputs. The remaining inputs or outputs remain unmodified.

For example, a model with m inputs and r outputs, where $m < r$, has m number of reference inputs. Similarly, a model with m inputs and r outputs, where $m > r$, has r number of reference inputs. All original y_r outputs remain in the resulting model.

The following sections provide information about how the CD Feedback VI configures the closed-loop feedback when you have one or two models in the closed-loop configuration. The following sections also describe the behavior of this VI when you leave connections undefined.

Single Model in a Closed-Loop Configuration

When you only have one model in a closed-loop configuration, the CD Feedback VI connects the outputs to the inputs of the same model. You define these connections using the **Feedback Connections** and the **Feedback Sign** parameters.

The following sections provide information about the configuration of the model when you define and do not define connections.

Feedback Connections Undefined

If you do not define **Feedback Connections**, all outputs from Model 1 are fed back to the inputs of Model 1. Additionally, the **Feedback Sign** input determines if these outputs are fed back negatively or positively. The resulting model, shown in Figure 4-15, contains new reference inputs r_0 and r_1 for each feedback connection you specify.

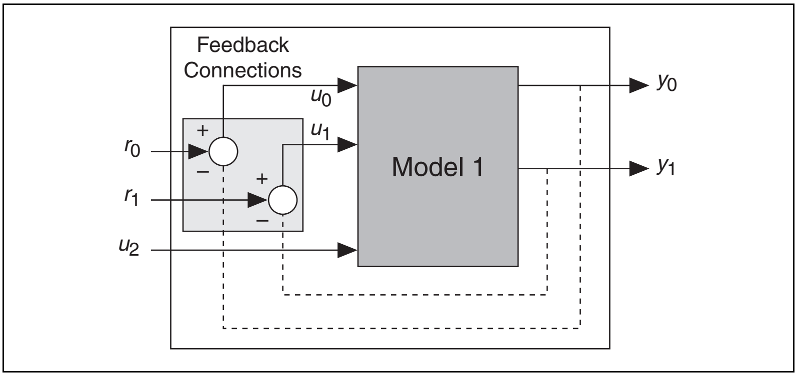

Figure 4-15. One Model with No Connections Defined

Feedback Connections Defined

If you define **Feedback Connections**, each specified output in Model 1 is fed back to each specified input of Model 1. You also define whether the connection is positive or negative. In this situation, the CD Feedback VI ignores the **Feedback Sign** input. The resulting model, shown in Figure 4-16, contains a new reference input r_0 for each feedback connection you specify.

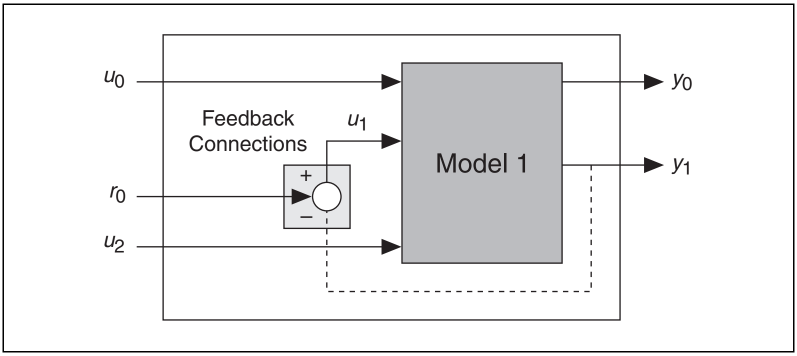

Figure 4-16. One Model with Connections Defined

Two Models in a Closed-Loop Configuration

When you have two models in a closed-loop configuration, the first model is always in the open-loop path, and the second model is always in the feedback path. You have the option to define feedback connections, output connections, both types of connections, or no types of connections.

Within the CD Feedback VI, **Feedback Connections** defines the connection between the outputs of Model 2 and the inputs of Model 1. **Output Connections** defines the connection between the outputs of Model 1 and the inputs of Model 2. By default, the CD Feedback VI connects the models with negative feedback.

The resulting model differs depending on the number of connections you define. The following sections provide information about the configuration of the models when you define or do not define connections.

Feedback and Output Connections Undefined

If you do not define **Feedback Connections** or **Output Connections**, the CD Feedback VI tries to connect all the outputs of Model 1 to the corresponding inputs of Model 2. The CD Feedback VI also tries to connect all the outputs of Model 2 to the corresponding inputs of Model 1. The **Feedback Sign** input determines if these outputs are fed back negatively or positively. By default, the CD Feedback VI connects the models with negative feedback.

The resulting model, shown in Figure 4-17, contains new reference inputs r_0 and r_1 for each feedback connection.

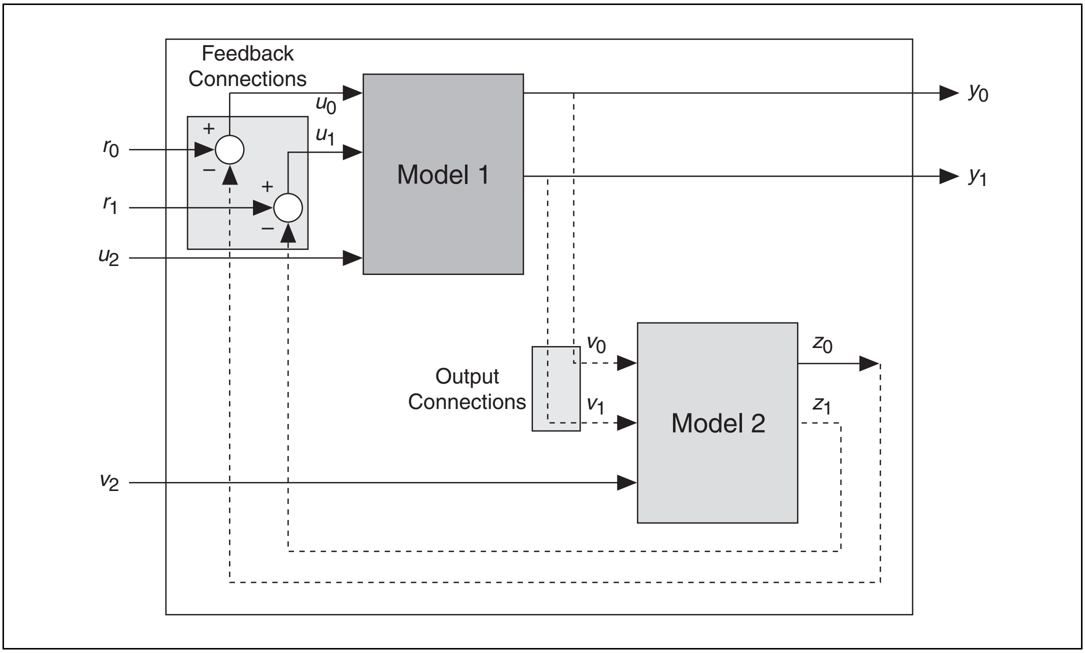

Figure 4-17. Two Models with No Connections Defined

Feedback Connections Undefined, Output Connections Defined

If you do not define **Feedback Connections** but define **Output Connections**, the CD Feedback VI connects the specified outputs for Model 1 to the specified inputs for Model 2. You define whether each connection is positive or negative. Because you have not defined **Feedback Connections**, the CD Feedback VI connects all outputs of Model 2 to the corresponding inputs in Model 1 based on the **Feedback Sign**.

Note All outputs of Model 1, whether they are connected to Model 2 outputs or not, remain as outputs in the resulting model. Conversely, Model 2 outputs do not remain in the resulting model when fed back to Model 1 inputs.

The resulting model, shown in Figure 4-18, contains new reference inputs r_0 and r_1 for each feedback connection.

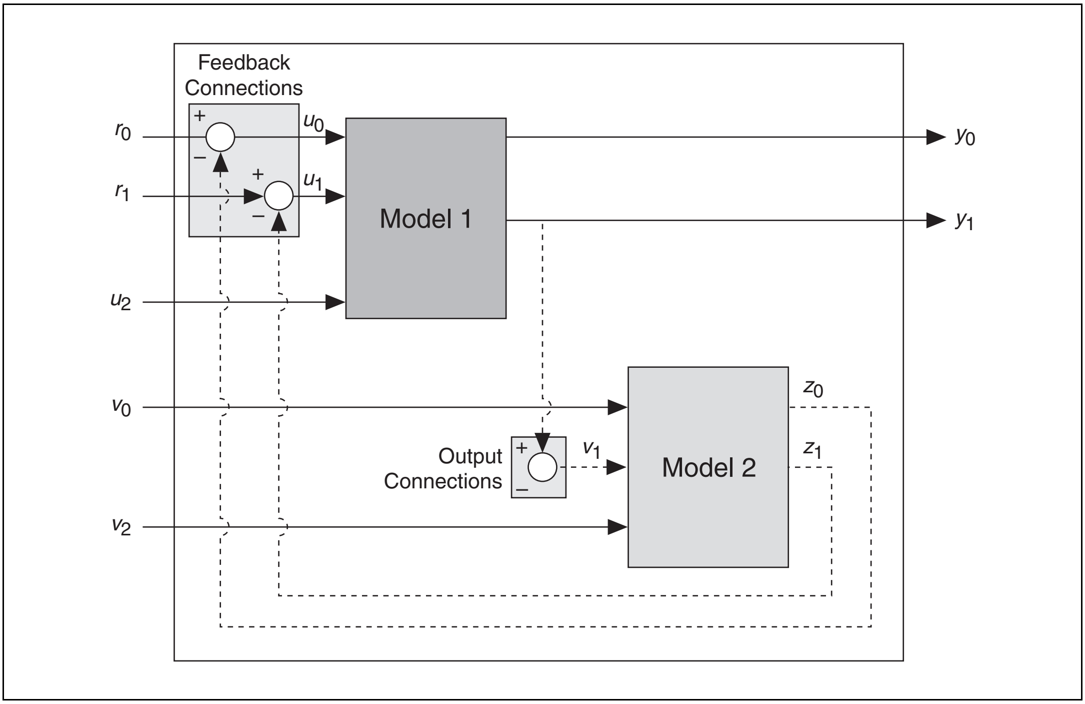

Figure 4-18. Two Models with Output Connections Defined

Feedback Connections Defined, Output Connections Undefined

If you define **Feedback Connections** but not **Output Connections**, the CD Feedback VI feeds the outputs specified for Model 2 back to the specified inputs for Model 1. You define whether the feedback connection is positive or negative. Because you have not defined **Output Connections**, the CD Feedback VI tries to connect all outputs of Model 1 positively to the inputs in Model 2.

The resulting model, shown in Figure 4-19, contains a new reference input r_0 for each feedback connection you have defined.

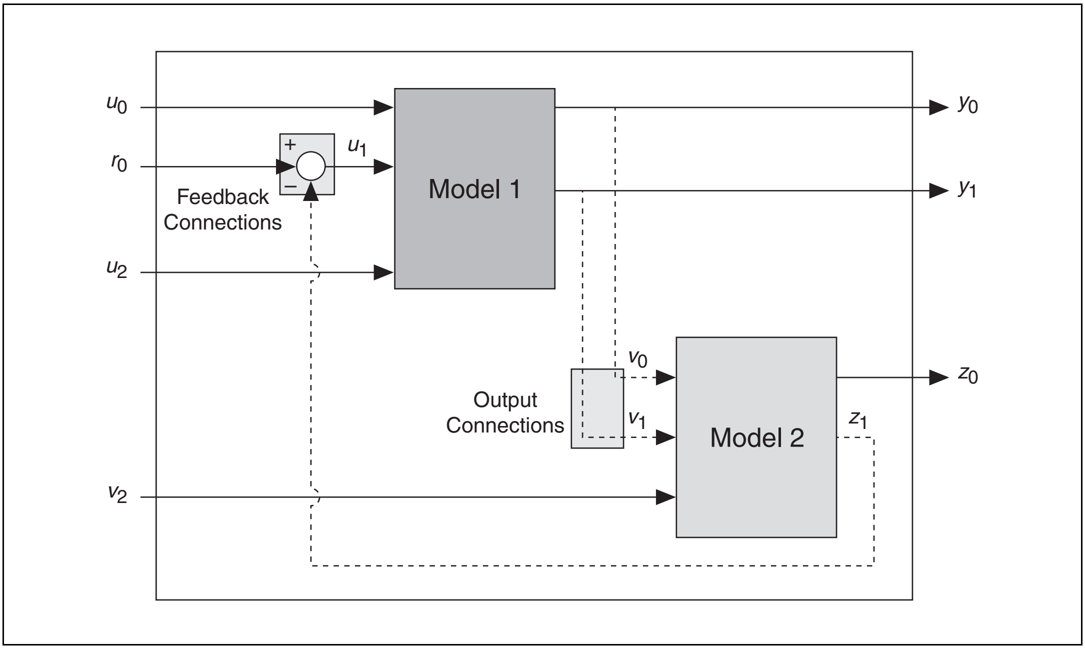

Figure 4-19. Two Models with Feedback Connections Defined

Both Feedback and Output Connections Defined

If you specify connections in both **Feedback Connections** and in **Output Connections**, you define all connections. Based on the connections you specified in **Output Connections**, the outputs specified for Model 1 are connected to the inputs specified for Model 2. You define whether the connection is positive or negative.

Based on the connections you specified in **Feedback Connections**, the outputs specified for Model 2 are fed back to the inputs specified for Model 1. You also define whether the feedback connection is positive or negative. Outputs of Model 2 not specified in **Feedback Connections** are removed from the resulting model. Again, because you specified connections using the **Feedback Connections**, the CD Feedback VI ignores the **Feedback Sign** input.

In the resulting model, shown in Figure 4-20, you can see how the CD Feedback VI creates a new reference input r_0 for each feedback connection you specified.

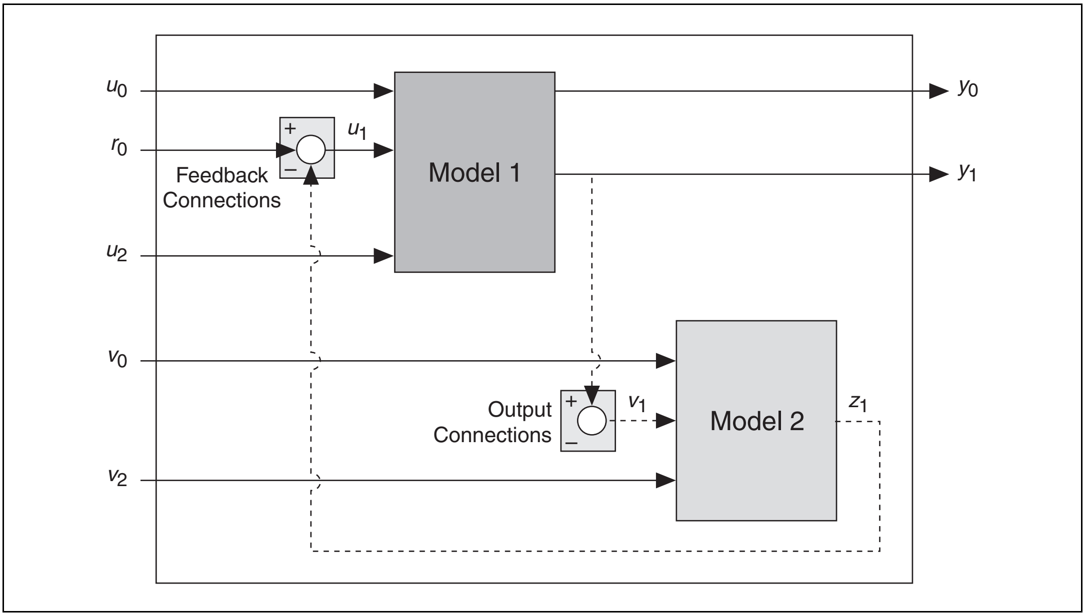

Figure 4-20. Two Models with Feedback and Output Connections Defined

5

Working with Delay Information

Delays in a system model account for the fact that the inputs and outputs of a system often do not respond immediately to excitation. For example, chemical plants transfer fluid and materials between the process equipment, the actuators, and the sensors. This transportation process can cause long delays in the output response of the system. To fully represent this system, a model must incorporate this delay. If a model of this system does not incorporate delay, you cannot predict how well a controller based on that model performs.

A system model can have the following three types of delay:

- Input delay—The time a past input takes to affect the current output
- Output delay—The time an output takes to respond to the current system input
- Transport delay—The time the dynamics of a system take to respond to a particular excitation

The total delay of a system model is the sum of all delays between each input-output pair. The total delay includes all input, output, and transport delays in the system model. Another type of delay, residual delay, results from certain operations. Refer to the *Residual Delay Information* section of this chapter for more information about residual delay.

Constructing a model in the LabVIEW Control Design Toolkit sets delay information but does not make that information part of the mathematical model. The Control Design Toolkit provides several VIs that you can use to transfer delay information from the model properties into the mathematical model. After you incorporate delay into a mathematical model, the model properties no longer contain delay information, and the delay information appears in any analysis you perform on the model.

This chapter provides information about using the Control Design Toolkit to account for delay information in a model and to manipulate delay information within a model.

Accounting for Delay Information

Accounting for delay information in a model involves the following two steps: setting delay in the properties of a model and transferring that delay from the model properties to the mathematical model. The following sections provide information about the Control Design VIs that you can use to accomplish these tasks.

Setting Delay Information

By default, when you construct a model in the Control Design Toolkit, the properties of that model have a delay of zero. Use the CD Set Delays to Model VI to define any non-zero delays in a model. You can use the **Input Delays**, **Output Delays**, and **Transport Delays** inputs of this VI to define the input, output, and transport delays in a model. The properties of the resulting output **Model Out** contain the original model with the delay information you defined.

You also can retrieve the delay information from the properties of a model with the CD Get Delays from Model VI. This VI returns the input, output, and transport delays of a model in the **Input Delays**, **Output Delays**, and **Transport Delays** outputs, respectively.

Incorporating Delay Information

After you define any delay information in a model, you then can make that delay a permanent part of the model. Incorporating delay information into a model works differently for continuous system models and discrete system models. In both cases, you represent a common delay factor and multiply the system model by this factor. The process by which you determine this factor, however, varies depending on the type of system model. With continuous system models, you apply the Laplace transformation to the system to represent the delay as an exponential factor. With discrete system models, you apply the shift operator to the system to represent the delay as a factor.

The delay factor for a continuous system is e^{-st_d}. The delay factor for a discrete system is z^{-n_d}. Refer to the *Delay Information in Continuous System Models* section and the *Delay Information in Discrete System Models* section of this chapter for information about these delay factors and incorporating them into system models.

Note These delay factors do not always have the same value in systems with more than one input-output pair. Single-input multiple-output (SIMO), multiple-input single-output (MISO), and multiple-input multiple-output (MIMO) system models have more than one input-output pair, and the delay might be different between each pair. Conversely, because single-input single-output (SISO) systems only have one input-output pair, the delay factor in a SISO system model always has the same value. Refer to the *Residual Delay Information* section of this chapter for more information about systems that do not have a common delay factor.

Use the CD Convert Delay with Pade Approximation VI to incorporate delay information into continuous models. Use the CD Convert Delay to Poles at Origin VI to incorporate delay information into discrete models. If you incorporate the delays in the model using one of these VIs, the Dynamic Characteristics VIs and the State Feedback Design VIs account for the delays in their results. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for more information about which VIs account for delays.

The following sections provide information about using the Control Design Toolkit to incorporate delay into continuous and discrete system models.

Delay Information in Continuous System Models

Mathematically, incorporating delay into a continuous system model involves evaluating that model at t_d units in the past, where t is the current time. For example, consider the continuous SISO system model $h(t)$. To represent this model at t_d units in the past, subtract t_d from t in the evaluation of the system model $h(t)$. The expression $h(t - t_d)$ represents this operation.

The first step in incorporating delay into a continuous system model is factoring a common delay out of the system model. Applying the Laplace transformation to the system model accomplishes this step. The following equation gives the Laplace transformation of $h(t - t_d)$.

$$\pounds[h(t-t_d)] \equiv \int_0^\infty h(t-t_d)e^{-st}dt = \int_0^\infty h(t-t_d)e^{-s(t-t_d)}d(t-t_d)e^{-st_d} = H(s)e^{-st_d}$$

This equation shows that the Laplace transform of a function delayed t_d units of time in the past is identical to the product of the Laplace transform of the original function and the factor e^{-st_d}, where s is the Laplace variable. Thus, you can incorporate delay into $h(t)$ by multiplying $H(s)$ by the delay factor e^{-st_d}.

For example, consider the continuous SISO transfer function $H(s)$ with output $Y(s)$ and input $U(s)$. Because e^{-st_d} represents the delay factor, $H(s)e^{-st_d}$ defines a system that has a transport delay.

$$H(s)e^{-st_d} = \frac{Y(s)}{U(s)}$$

You also can represent the delay as an input delay or output delay. Applying the delay factor e^{-st_d} to the input $U(s)$ results in an input delay as shown in the following equation:

$$H(s) = \frac{Y(s)}{e^{-st_d}U(s)}$$

Conversely, applying the delay factor to the output $Y(s)$ results in an output delay shown in the following equation:

$$H(s) = \frac{e^{st_d}Y(s)}{U(s)}$$

Figure 5-1 shows the mathematical representation of transport, input, and output delay factors for a continuous system.

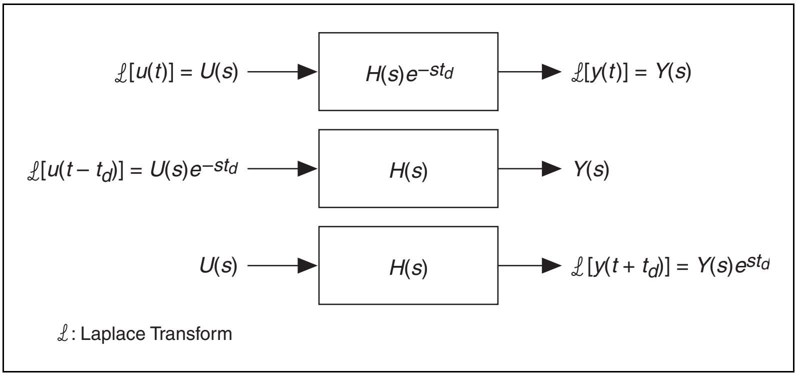

Figure 5-1. Mathematical Representation of Transport, Input, and Output Delay for a Continuous System

To accommodate the delay factor, you can convert e^{-st_d} from exponential form to a rational polynomial function. You can perform this conversion using the Padé approximation method. Use the CD Convert Delay with Pade Approximation VI to calculate a Padé approximation. This VI incorporates the delay information of the input model into the **Converted Model** output model. The delay becomes a part of the output

model and thus is not in the model properties. In the case of SIMO, MISO, and MIMO system models, the CD Convert Delay with Pade Approximation VI calculates the total delay in all the input-output pairs before incorporating the delay into the model.

This conversion process has several benefits. First, connecting models that contain all rational polynomial functions is less complicated than connecting models that contain a mixture of exponential factors and rational polynomial functions. Second, when you incorporate the delay into the polynomial function, the controller structure, analysis operations, and synthesis operations account for the delay.

Note The CD Convert Delay with Pade Approximation VI converts a state-space model to a transfer function model before incorporating the delay information. This VI then converts the resulting model back to a state-space model. As a result, the final states of the model might not directly correspond to the original states. Refer Chapter 3, *Converting Models*, for more information about converting between model forms.

For example, consider a continuous SISO system with an input delay of 25 seconds. The delay factor in this system is e^{-25s}, so the following equation represents the system:

$$H(s) = \frac{Y(s)}{e^{-25s}U(s)}$$

Figure 5-2 shows the step response of this system.

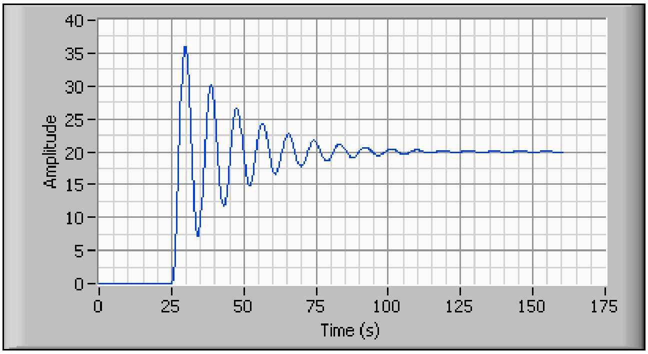

Figure 5-2. Step Response with a 25 Second Delay

You can see that incorporating e^{-25s} into the input of $H(s)$ delays the step response of $H(s)$ by 25 seconds. Refer to the *Analyzing a Step Response*

section of Chapter 6, *Time Response Analysis*, for information about a step response.

You can use the **Polynomial Order** input of the CD Convert Delay with Pade Approximation VI to affect the accuracy of the approximation. A larger **Polynomial Order** means a more accurate approximation but results in a higher-order system model. A large **Polynomial Order** can have the unintended side effect of making a model too complex to be useful.

Figure 5-3 shows the effects of polynomial orders on the accuracy of a Padé approximation of $H(s)$.

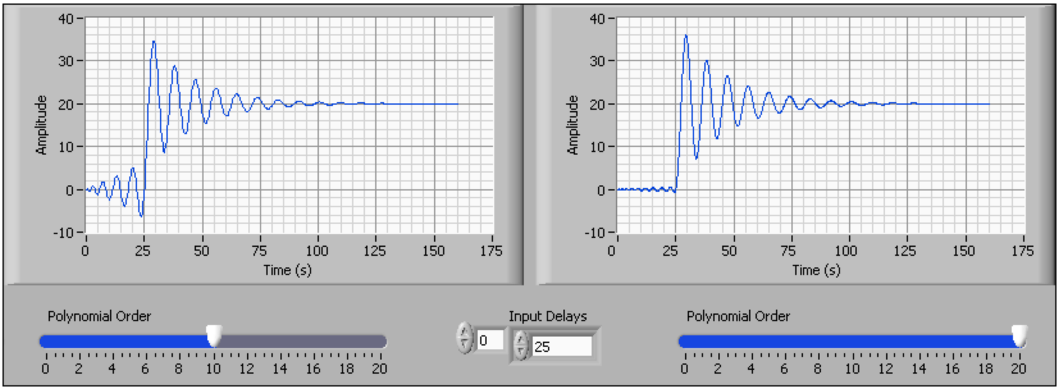

Figure 5-3. Effect of Polynomial Orders for a Padé Approximation

Delay Information in Discrete System Models

Mathematically, incorporating delay into a discrete system model involves evaluating that model at n_d units in the past. n_d equals the delay divided by the sampling time T of the system. For example, consider the discrete SISO system model $y(k)$. The equation $y(kT - n_dT)$ provides the output of $y(k)$ at n_d units in the past, where k represents the current sample. Removing the sampling time T from this equation provides the simplified equation $y(k - n_d)$. This simplified equation produces the same result as $y(kT - n_dT)$.

This equation shows the delay factor z^{-n_d} for a discrete system model, where z represents time in the discrete domain. You use z^{-n_d} to evaluate $y(k)$ at n_d samples in the past. The following equation shows this process, which also is known as applying the shift operator.

$$y(k - n_d) = y(k) \cdot z^{-n_d}$$

In transfer function models and zero-pole-gain models, incorporating delay information means adding poles at the origin. By applying z^{-n_d} to a transfer function or zero-pole-gain model, you increase the order of the denominator polynomial by adding n_d poles at the origin. In state-space models, incorporating delay information means creating n_d additional states.

Use the CD Convert Delay to Poles at Origin VI to incorporate delays into discrete models. This VI incorporates the delay information of the input model into the **Converted Model** output model. The delay becomes a part of the output model and thus is not in the model properties. In the case of SIMO, MISO, and MIMO system models, the CD Convert Delay to Poles at Origin VI totals the delay in all the input-output pairs before incorporating the delay into the model.

Figure 5-4 shows how you can create a transfer function model, define an input delay for the model properties, and then incorporate that delay directly into the model.

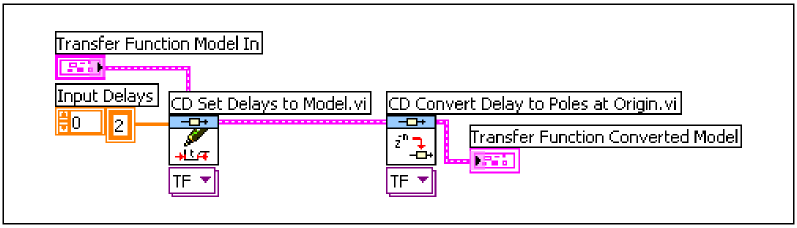

Figure 5-4. Adding Delay Information to a Discrete Transfer Function Model

Figure 5-5 shows the resulting transfer function model. The CD Convert Delay to Poles at Origin VI accounted for the input delay by increasing the number of poles at the origin in the model. Accordingly, the **Transfer Function Converted Model** has a larger order denominator than the **Transfer Function Model In**.

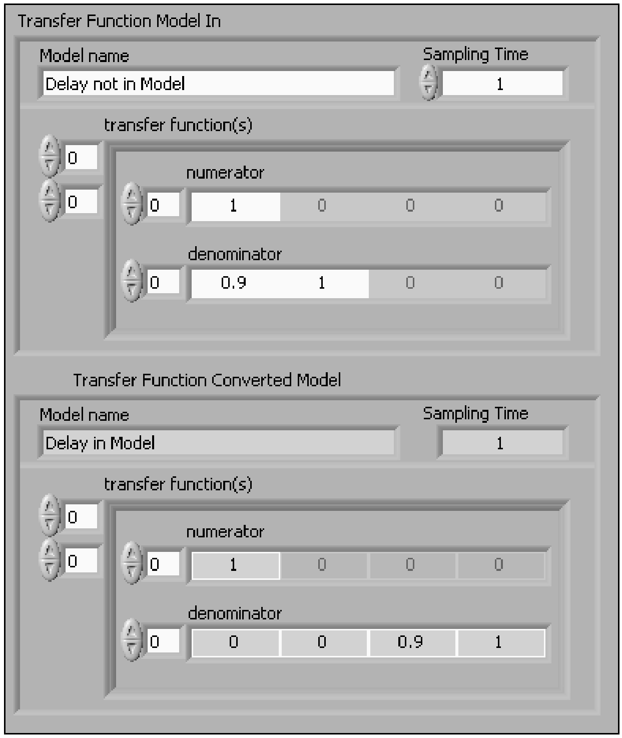

Figure 5-5. Additional Poles Accounting for the Input Delay

The **Transfer Function Converted Model** expresses the additional poles at the origin with two additional zeroes in the **denominator**.

Representing Delay Information

To illustrate how the Control Design Toolkit represents delay in a system model, consider the following MIMO transfer function equation, where ***U*** is the input transfer function matrix and ***Y*** is the output transfer function matrix.

$$\boldsymbol{H} = \frac{\boldsymbol{Y}}{\boldsymbol{U}}$$

The following equations define this MIMO transfer function:

$$\boldsymbol{H} = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \qquad \boldsymbol{U} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \qquad \boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

The following equations define the transport delay matrix $\boldsymbol{T}_d$, the input delay vector $\boldsymbol{I}_d$, and the output delay vector $\boldsymbol{O}_d$. Refer to the *Delay Information in Continuous System Models* section of this chapter for the definition of the continuous delay factor e^{-st_d}.

$$\boldsymbol{T}_d = \begin{bmatrix} e^{-st_{11}} & e^{-st_{12}} \\ e^{-st_{21}} & e^{-st_{22}} \end{bmatrix} \qquad \boldsymbol{I}_d = \begin{bmatrix} e^{-st_1} \\ e^{-st_2} \end{bmatrix} \qquad \boldsymbol{O}_d = \begin{bmatrix} e^{st_a} \\ e^{st_b} \end{bmatrix}$$

To incorporate this delay information into $\boldsymbol{H}$, compute the product of the transfer function, input, and output matrices with their respective delay matrices or vectors. $\boldsymbol{H}_d$, shown in the following equation, represents $\boldsymbol{H}$ with delay information included.

$$\boldsymbol{H}_d = \frac{\boldsymbol{Y}_d}{\boldsymbol{U}_d}$$

The following equations show the computation of these transfer functions to incorporate delay.

$$\boldsymbol{H}_d = \begin{bmatrix} H_{11}e^{-st_{11}} & H_{12}e^{-st_{12}} \\ H_{21}e^{-st_{21}} & H_{22}e^{-st_{22}} \end{bmatrix} = \boldsymbol{H} \cdot \boldsymbol{T}_d = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \cdot \begin{bmatrix} e^{-st_{11}} & e^{-st_{12}} \\ e^{-st_{21}} & e^{-st_{22}} \end{bmatrix}$$

$$\boldsymbol{U}_d \equiv \begin{bmatrix} U_1 e^{-st_1} \\ U_2 e^{-st_2} \end{bmatrix} = \boldsymbol{U} \cdot \boldsymbol{I}_d = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \cdot \begin{bmatrix} e^{-st_1} \\ e^{-st_2} \end{bmatrix}$$

$$\boldsymbol{Y}_d \equiv \begin{bmatrix} Y_1 e^{st_a} \\ Y_2 e^{st_b} \end{bmatrix} = \boldsymbol{Y} \cdot \boldsymbol{O}_d = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \cdot \begin{bmatrix} e^{st_a} \\ e^{st_b} \end{bmatrix}$$

To represent the delay of each element, you can use the following matrices:

$$\boldsymbol{T}_d = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \qquad \boldsymbol{I}_d = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \qquad \boldsymbol{O}_d = \begin{bmatrix} t_a \\ t_b \end{bmatrix}$$

Because the number of rows and columns of $\boldsymbol{T}_d$ are the same as the dimension of vectors $\boldsymbol{I}_d$ and $\boldsymbol{O}_d$, you can represent all the delay information of a model using the following structure:

$$\begin{matrix} \begin{bmatrix} t_1 & t_2 \end{bmatrix} & \\ \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} & \begin{bmatrix} t_a \\ t_b \end{bmatrix} \end{matrix}$$

In this delay matrix, the input delay vector $\boldsymbol{I}_d$ is on top. Each input uses one column. The output delay vector $\boldsymbol{O}_d$ is on the right-hand side. Each output uses one row.

Manipulating Delay Information

The Control Design Toolkit provides two VIs to help you manipulate the delay information of a system model. Use the CD Distribute Delay VI to minimize the transport delay of a system model by distributing the transport delay information to the inputs and outputs of a system model. Use the CD Total Delay VI to distribute the input and output delay of a model to the transport delay. The following sections provide information about using these VIs to manipulate delay information.

Accessing Total Delay Information

The CD Total Delay VI transfers delay information from the inputs and outputs of a system model to the transport delay of a system model by adding the input and output delays to the delay in the transport delay matrix. When you use the CD Total Delay VI, other Control Design VIs can access the total delay information of a system.

For example, consider a model with the following delay information. Refer to the *Representing Delay Information* section of this chapter for the derivation of this matrix and these vectors.

$$\begin{array}{c}\begin{bmatrix} t_1 & t_2 \end{bmatrix}\\ \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix}\end{array}\begin{bmatrix} t_a \\ t_b \end{bmatrix} = \begin{array}{c}\begin{bmatrix} 1 & 2 \end{bmatrix}\\ \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}\end{array}\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

The CD Total Delay VI first transfers the input delay information to the transport delay matrix. The following equations show this process:

$$\begin{array}{c}\begin{bmatrix} 1-1 & 2-2 \end{bmatrix}\\ \begin{bmatrix} 2+1 & 1+2 \\ 1+1 & 0+2 \end{bmatrix}\end{array}\begin{bmatrix} 1 \\ 2 \end{bmatrix} \approx \begin{array}{c}\begin{bmatrix} 0 & 0 \end{bmatrix}\\ \begin{bmatrix} 3 & 3 \\ 2 & 2 \end{bmatrix}\end{array}\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

The CD Total Delay VI then transfers the output delay information to the transport delay matrix. The following equations show this process:

$$\begin{array}{c}\begin{bmatrix} 0 & & 0 \end{bmatrix}\\ \begin{bmatrix} 3+1 & & 3+1 \\ 2+2 & & 2+2 \end{bmatrix}\end{array}\begin{bmatrix} 1-1 \\ 2-2 \end{bmatrix} \approx \begin{array}{c}\begin{bmatrix} 0 & 0 \end{bmatrix}\\ \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}\end{array}\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Figure 5-6 shows the output of the CD Total Delay VI.

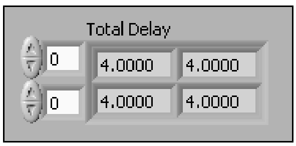

Figure 5-6. Resulting Total Delay

The input and output delay vectors are now $\begin{bmatrix} 0 & 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$, respectively.

Distributing Delay Information

The CD Distribute Delay VI calculates the total delay of a system model, then uses a common delay factor to distribute the total delay between the inputs and outputs. This operation minimizes the non-zero elements of the transport delay matrix. The CD Distribute Delay VI transfers delay information to the input delays before transferring delay information to the output delays.

Note Some Control Design VIs internally distribute the delay to preserve as much delay information as possible in the resulting model. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, to determine which VIs manipulate the transport delay matrix to preserve delay information.

For example, consider the system model described in the *Accessing Total Delay Information* section of this chapter. If you apply the CD Distribute Delay VI to this system model, you get the following equation:

$$\begin{matrix}\begin{bmatrix}0 & 0\end{bmatrix} & \\ \begin{bmatrix}4 & 4\\ 4 & 4\end{bmatrix} & \begin{bmatrix}0\\ 0\end{bmatrix}\end{matrix} \approx \begin{matrix}\begin{bmatrix}4 & 4\end{bmatrix} & \\ \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix} & \begin{bmatrix}0\\ 0\end{bmatrix}\end{matrix}$$

Because 4 is the common factor among the transport delay matrix, the CD Distribute Delay transferred a delay of 4 to the input delays.

Figure 5-7 shows the output of the CD Distribute Delay VI.

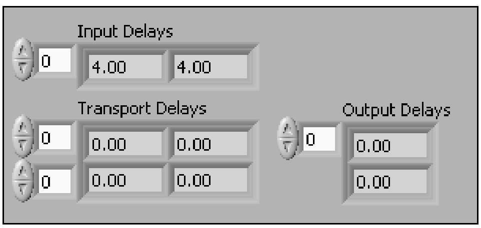

Figure 5-7. Resulting Delay Distribution

The input and output delay vectors are now $\begin{bmatrix}4 & 4\end{bmatrix}$ and $\begin{bmatrix}0\\ 0\end{bmatrix}$, respectively.

Figure 5-8 shows how you implement this example using the Control Design Toolkit.

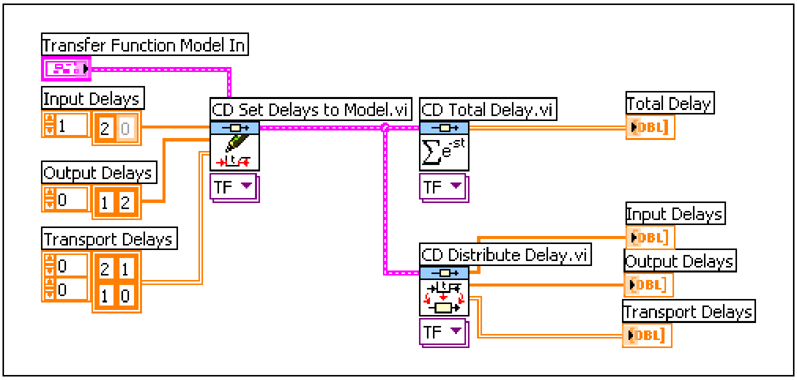

Figure 5-8. Totaling and Distributing the Delay Information in a Model

Residual Delay Information

Residual delay information is transport delay information that remains when the CD Distribute Delay VI cannot distribute all of the transport delay to the inputs or outputs. This situation most often occurs in SIMO, MISO, and MIMO system models because each input-output pair can have different delay information.

For example, consider a system model with the following delay information:

$$\begin{array}{cc} \begin{bmatrix} 0 & 0 \end{bmatrix} & \\ \begin{bmatrix} 5 & 3 \\ 4 & 4 \\ 3 & 5 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

The CD Distribute Delay VI first distributes the delay in the transport delay matrix to the input delay vector by subtracting the minimum value from each column in the transport delay matrix. In this case, the minimum value in both columns is 3. This VI then distributes the delay to the output delay vector by subtracting the minimum value from each row in the resulting

transport delay matrix. In this case, only the second row has a minimum value other than 0.

$$\begin{bmatrix} 0 & 0 \\ \hline 5 & 3 \\ 4 & 4 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 3 & 3 \\ \hline 2 & 0 \\ 1 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 3 & 3 \\ \hline 2 & 0 \\ 0 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Because the CD Distribute Delay VI cannot fully distribute all the delays, the transport delay matrix contains the residual delay information.

6

Time Response Analysis

The time response of a dynamic system provides information about how the system responds to certain inputs. You analyze the time response to determine the stability of the system and the performance of the controller.

Obtaining the time response of a system involves numerically integrating the system model in time. The LabVIEW Control Design Toolkit provides VIs to help you find these time-domain solutions. You can use these Time Response VIs to analyze the response of a system to step and impulse inputs. You can apply initial conditions to both of these responses. You also can use the Time Response VIs to simulate the response of the system to an arbitrary input.

This chapter provides information about using the Control Design Toolkit to measure and analyze the time response of a system. This chapter also provides information about solving the time-domain equations and simulating arbitrary inputs.

Calculating the Time-Domain Solution

The following equation represents the time-domain solution for a continuous state-space model.

$$\boldsymbol{x}(t) = e^{\boldsymbol{A}t}\boldsymbol{x}_0 + \int_0^t e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau$$

$\boldsymbol{x}_0$ represents any initial conditions of the states in the model. $e^{\boldsymbol{A}t}\boldsymbol{x}_0$ represents the solution of the model at the initial conditions. This solution is known as the free response.

$\int_0^t e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau$ represents the state response for stable systems over time as the inputs $\boldsymbol{u}(\tau)$ drive the dynamic system from time $t = t_0$ to t. This solution is the forced response.

The following equation represents the time-domain solution for a discrete state-space model.

$$\boldsymbol{x}(k) = \boldsymbol{A}^k\boldsymbol{x}(0) + \sum_{j=0}^{k-1}\boldsymbol{A}^{k-j-1}\boldsymbol{B}\boldsymbol{u}(j)$$

In this equation, $\boldsymbol{A}^k\boldsymbol{x}(0)$ denotes the discrete free response.

$\sum_{j=0}^{k-1}\boldsymbol{A}^{k-j-1}\boldsymbol{B}\boldsymbol{u}(j)$ denotes the discrete forced response.

Note The VIs discussed in this chapter automatically convert transfer function and zero-pole-gain models to state-space form before calculating the time-domain solution.

Spring-Mass Damper Example

To illustrate the different time responses you can obtain from a model, consider the following example of a spring-mass damper, shown in Figure 6-1.

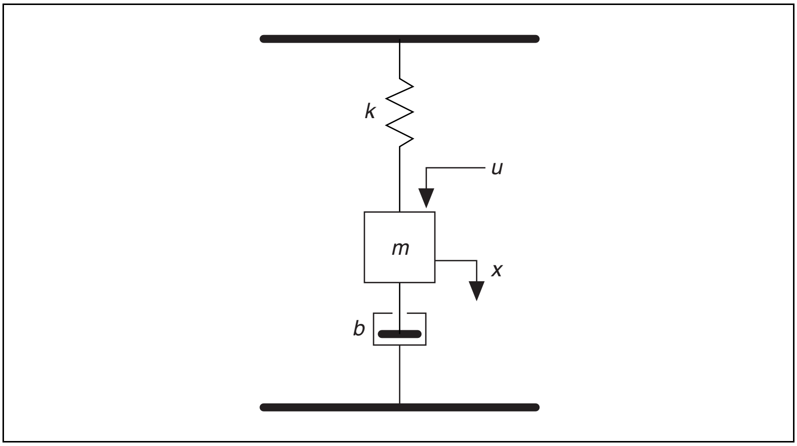

Figure 6-1. Spring-Mass Damper System

In this example, k is the spring constant, u is a force, m is the mass, and b is the damper coefficient. x is the displacement, which is the distance from the normal state of the spring to the current position of the spring. You can

represent this spring-mass damper system with the following state-space model:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}u$$

$$y = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}u = \begin{bmatrix} 1 & 0 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 0 \end{bmatrix}u = x$$

For this example, consider the following values:

$$k = 50\frac{kN}{cm} \qquad m = 100kg \qquad b = 10\frac{kN \cdot s}{cm}$$

The following equations define the state-space model.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ -0.5 & -0.1 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 0 \\ 0.01 \end{bmatrix}u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 0 \end{bmatrix}u = x$$

The following sections show how this system responds to different inputs.

Analyzing a Step Response

The step response of a dynamic system measures how the dynamic system responds to a step input signal. The following equations define a unit step input signal.

$$u(t) = 0 \; when \; t < 0$$
$$u(t) = 1 \; when \; t \geq 0$$

The Control Design Toolkit contains two VIs to help you measure the step response of a system and then analyze that response. The CD Step Response VI returns a graph of the step response. The CD Parametric Time

Response VI returns the following response data that helps you analyze the step response.

- Rise time (t_r)—The time required for the dynamic system response to rise from a lower threshold to an upper threshold. The default values are 10% for the lower threshold and 90% for the upper threshold.
- Maximum overshoot (M_p)—The dynamic system response value that most exceeds unity, expressed as a percent.
- Peak time (t_p)—The time required for the dynamic system response to reach the peak value of the first overshoot.
- Settling time (t_s)—The time required for the dynamic system response to reach and stay within a threshold of the final value. The default threshold is 1%.
- Steady state gain—The final value around which the dynamic system response settles to a step input.

Note You can modify the default values for the rise time thresholds and the settling time threshold using the **Rise Time Thresholds (%)** and **Settling Time Threshold (%)** parameters of the CD Parametric Time Response VI.

Figure 6-2 shows a sample step response graph and the locations of the parametric response data.

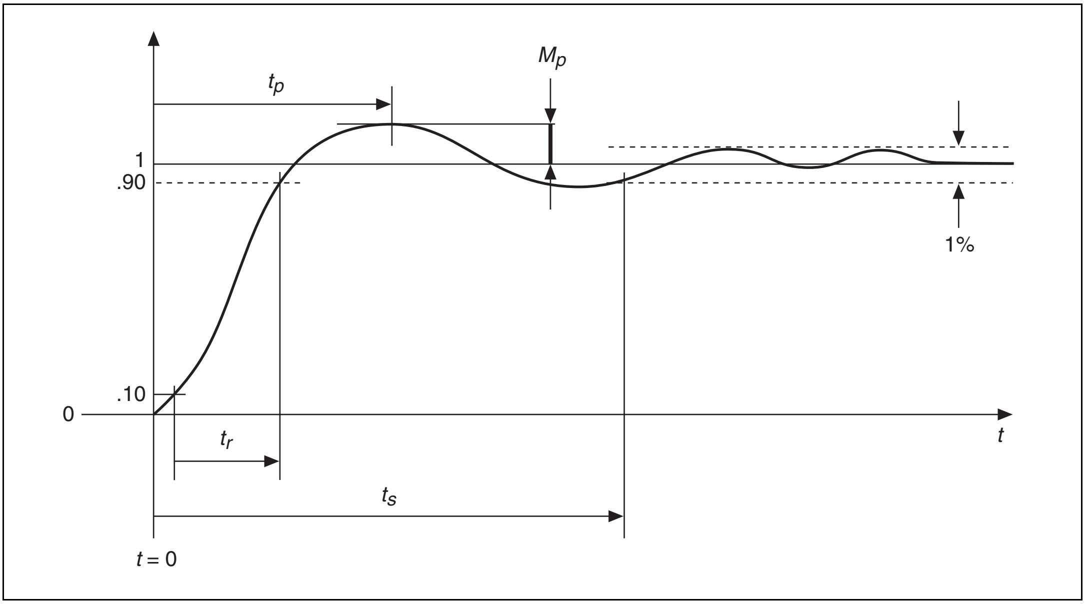

Figure 6-2. Step Response Graph and Associated Parametric Response Data

For example, consider the system described in the *Spring-Mass Damper Example* section of this chapter. Figure 6-3 shows how you determine the step response and associated parametric response data of this system.

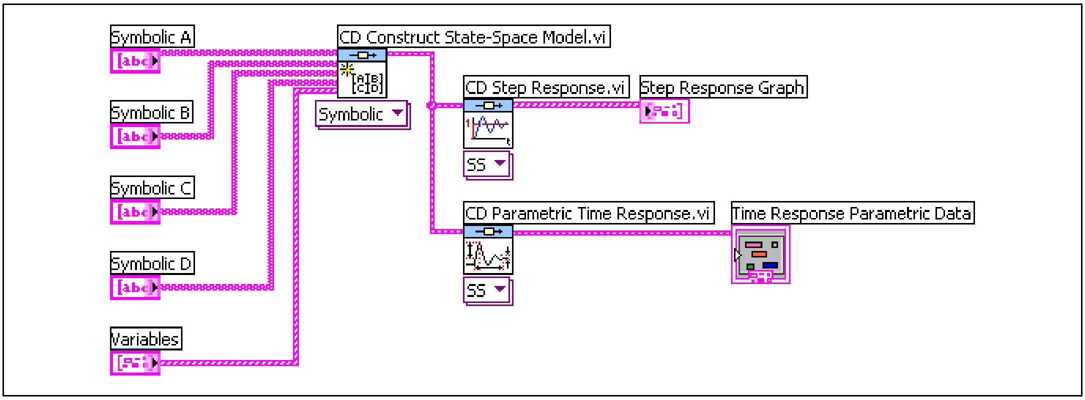

Figure 6-3. Step Response Block Diagram of the Spring-Mass Damper System

Figure 6-4 shows the **Step Response Graph** resulting from this block diagram.

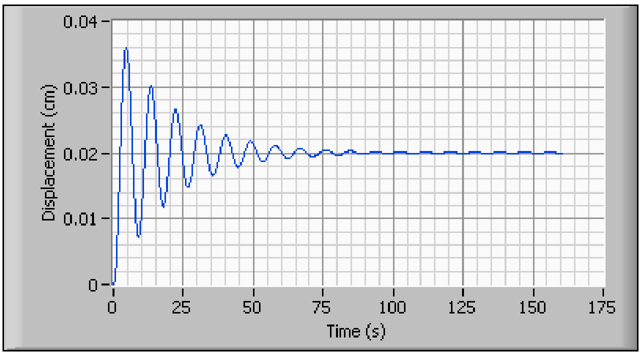

Figure 6-4. Step Response Graph of the Spring-Mass Damper System

You can see that the step input causes this system to settle at a steady-state value of 0.02 cm.

When you use the CD Parametric Time Response VI to analyze the step response of this system, you obtain the following response data:

- Rise time (t_r)—1.42 seconds
- Maximum overshoot (M_p)—79.90%
- Peak time (t_p)—4.54 seconds

- Settling time (t_s)—89.89 seconds
- Steady state gain—0.02 cm

Figure 6-5 shows the output of the CD Parametric Time Response VI.

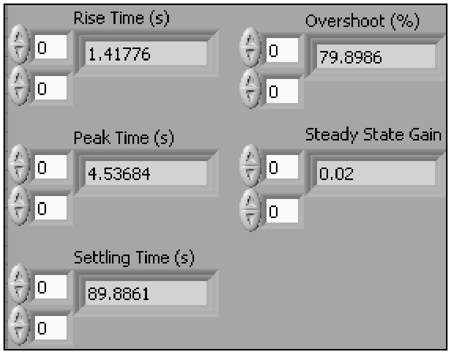

Figure 6-5. Parametric Data of the Spring-Mass Damper System

Analyzing an Impulse Response

The impulse response of a dynamic system measures how the system responds to an impulse input signal. You define an impulse input signal in the following manner:

- Continuous systems—Also known as the Dirac delta function, a continuous impulse input is a unit-area signal with an infinite amplitude and infinitely small duration occurring at a specified time. At all other times, the input signal value is zero.
- Discrete systems—Also known as the Kronecker delta function, a discrete impulse input is a physical pulse that has unit amplitude at the first sample period and zero amplitude for all other times.

Use the CD Impulse Response VI to calculate the impulse response of a dynamic system to a standard impulse input. Because the impulse signal excites all frequencies and the duration of this signal is infinitely small, the impulse response is the natural response of the system.

For example, consider the system described in the *Spring-Mass Damper Example* section of this chapter. Figure 6-6 shows how you determine the impulse response of this system.

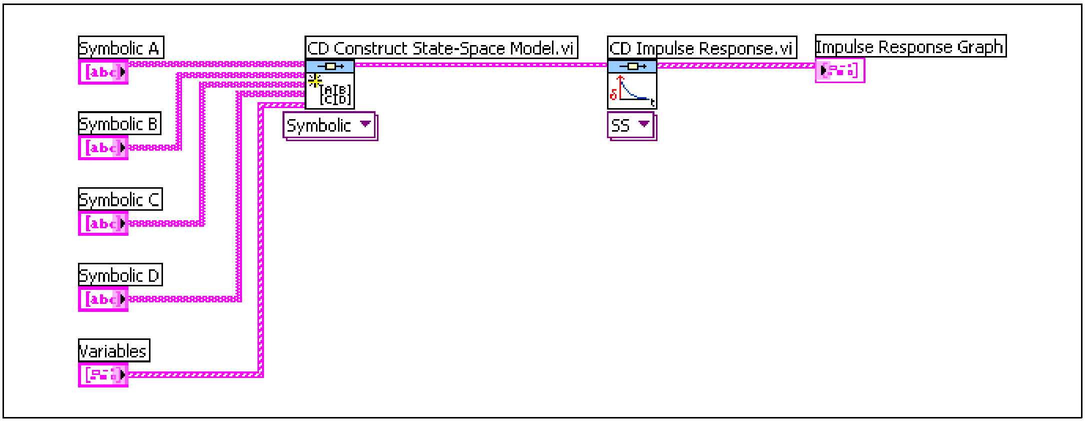

Figure 6-6. Impulse Response Block Diagram of the Spring-Mass Damper System

Figure 6-7 shows the **Impulse Response Graph** resulting from this block diagram.

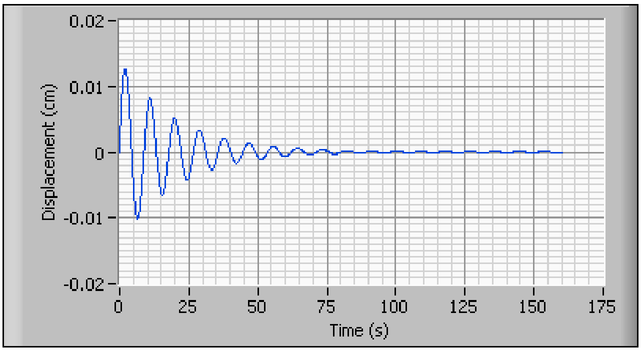

Figure 6-7. Impulse Response Graph of the Spring-Mass Damper System

Analyzing an Initial Response

The initial response of a dynamic system measures how the system responds to a set of non-zero initial conditions. Use the CD Initial Response VI to determine the initial response of a dynamic system.

Note The CD Step Response VI and the CD Impulse Response VI support initial conditions. Use the **Initial Conditions** parameter of these VIs to see how a set of initial conditions affects the step and/or impulse responses.

For example, consider the system described in the *Spring-Mass Damper Example* section of this chapter. Figure 6-8 shows how you determine the response of this system to an initial condition of 0.3 cm.

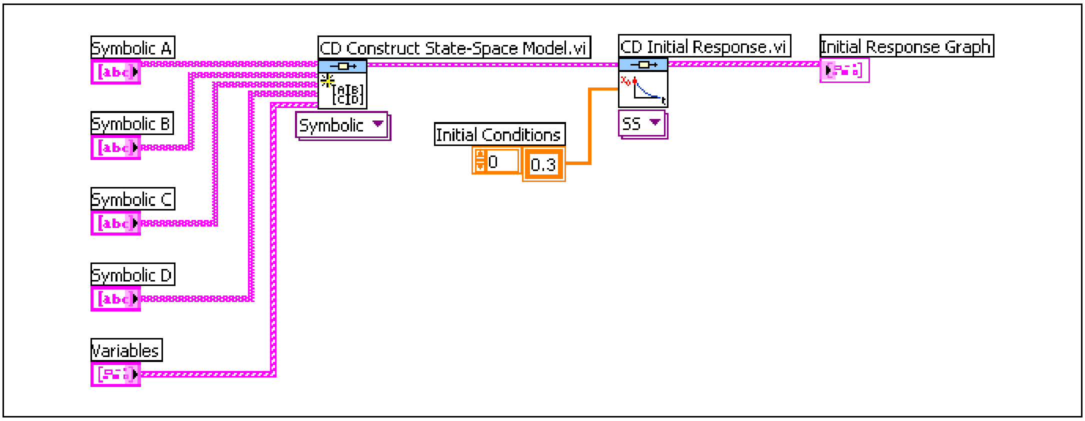

Figure 6-8. Initial Response Block Diagram of the Spring-Mass Damper System

Figure 6-9 shows the **Initial Response Graph** resulting from this block diagram.

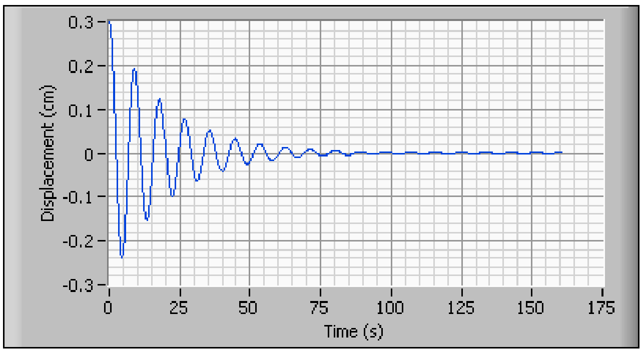

Figure 6-9. Initial Response Graph of the Spring-Mass Damper System

Notice that the displacement begins at the initial condition of 0.3 cm.

Analyzing a General Time-Domain Simulation

A general time-domain simulation of a system involves input signals that are more general than step, impulse, or initial input signals. Refer to the *Calculating the Time-Domain Solution* section of this chapter for equations representing the time response of continuous and discrete systems. Use the CD Linear Simulation VI to solve these equations in response to an arbitrary input signal ***u*** into a system. This VI determines the response by numerically integrating these equations at the specified time steps. You can define the time steps with the **Delta t** input.

The system model can be continuous or discrete, but the CD Linear Simulation VI converts continuous models to discrete models using either the exponential Zero-Order-Hold or the First-Order-Hold method. Refer to the *Converting Continuous Models to Discrete Models* section of Chapter 3, *Converting Models*, for more information about these methods.

If this conversion is necessary, you must specify **Delta t**, which becomes the sampling time. If no conversion is necessary, **Delta t** must be equal to the sampling time of the output data $\boldsymbol{u}(t)$.

Note For accurate results, use a sampling interval that is small enough to minimize the effects of converting a continuous system to a discrete one. Select this sampling time based on the location of the poles of the system. Refer to Chapter 8, *Analyzing Dynamic Characteristics*, for more information about locating the poles of a system. Also, verify that the sampling interval matches the sampling time of the output data $\boldsymbol{u}(t)$.

For example, consider the system described in the *Spring-Mass Damper Example* section of this chapter. Figure 6-10 shows how you simulate the response of this system to a square wave input.

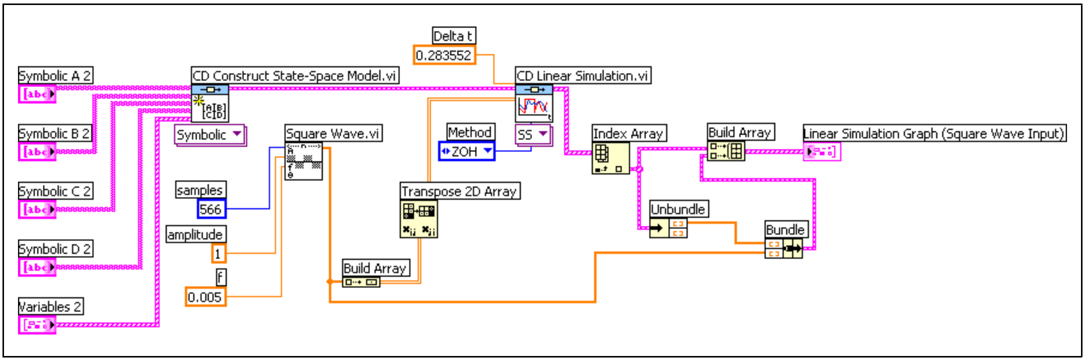

Figure 6-10. Linear Simulation Block Diagram of the Spring-Mass Damper System Using a Square Wave Input

Notice that the CD Linear Simulation VI converts the continuous state-space model to a discrete model using the Zero-Order-Hold method. This conversion uses a **Delta t** input of approximately 0.3. This block diagram bundles the state-space model and the square wave as the input to the **Linear Simulation Graph**.

Figure 6-11 shows the **Linear Simulation Graph** resulting from this block diagram.

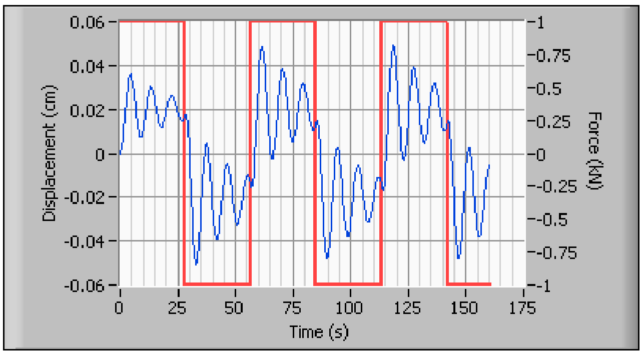

Figure 6-11. Linear Simulation Graph of the Spring-Mass Damper System Using a Square Wave Input

The scale for the square wave input is on the right-hand side of the graph, whereas the scale for the linear simulation output is on the left-hand side of the graph. You can specify any input and use the CD Linear Simulation VI to observe how the system responds to that input.

Obtaining Time Response Data

The Time Response VIs return time response data that contains information about the time response of all input-output pairs in the model. Use the CD Time Response VI to access this information for a specified input-output pair, a list of input-output pairs, or all input-output pairs of the system.

The CD Time Response VI uses the **Time Response Data** input, which contains the time response information for all the input-output pairs of a system model. If the system model is in state-space form, you can use the **Type of Response Data** parameter to obtain the time response of the input-state pair(s) as opposed to the input-output pair(s). Because transfer function and zero-pole-gain models do not have states, the time response data for an input-state pair of these forms is an empty array.

Refer to the *LabVIEW Help,* available by selecting **Help»VI, Function, & How-To Help**, for more information about the CD Get Time Response Data VI.

7

Frequency Response Analysis

The frequency response of a dynamic system is the output of a system given unit-amplitude, zero-phase sinusoidal inputs at varying frequencies. You can use the frequency response of a system to locate poles and zeroes of a system. Using this information, you then can design a controller to improve unwanted parts of the frequency response.

When applied to the system, a sinusoidal input with unit amplitude, zero phase, and frequency ω produces the following sinusoidal output.

$$H(i\omega) = A(\omega)e^{i\phi(w)}$$

A is the magnitude of the response as a function of ω, and ϕ is the phase. The magnitude and phase of the system output vary depending on the values of the system poles, zeroes, and gain.

This chapter provides information about using the LabVIEW Control Design Toolkit to perform Bode frequency analysis, Nichols frequency analysis, and Nyquist stability analysis.

Bode Frequency Analysis

Use Bode plots of system frequency responses to assess the relative stability of a closed-loop system given the frequency response of the open-loop system. By analyzing the frequency response, you can determine what the open- and closed-loop frequency responses of a system imply about the system behavior. Use the CD Bode VI to create a Bode plot.

Note Use the CD Evaluate at Frequency VI to determine the frequency at specified values.

For example, consider the following transfer function that represents a linear time-invariant system.

$$H(s) = \frac{Y(s)}{U(s)}$$

Applying the sinusoidal input $x(t) = \sin(\omega t)$ to this previous system produces the following equation:

$$y(t) = Y\sin(\omega t + \phi)$$

Using this equation, the following equation represents the complex frequency response.

$$H(i\omega) = A(\omega)e^{i\phi(w)}$$

You can separate the complex frequency response equation into two parts—the magnitude $A(\omega)$ and the phase $\phi(\omega)$. You obtain the magnitude from the absolute value of the response. You obtain the phase value from the four-quadrant arctangent of the response. The following equations illustrate these operations:

$$A(\omega) = |H(i\omega)|$$

$$\phi(\omega) = \angle H(i\omega) = \mathrm{atan}\left[\frac{Imaginary\ H(i\omega)}{Real\ H(i\omega)}\right]$$

These two equations represent the magnitude and the phase of the frequency response, respectively. Plotting these equations results in two subplots—the Bode magnitude plot and the Bode phase plot. The Bode magnitude plot shows the gain plotted against the frequency. The Bode phase plot shows the phase, in degrees, as a function of the frequency.

Use a linear scale when dealing with phase information. When using a linear scale, you can add the individual phase elements together to determine the phase angle.

Because you can add the magnitude and phase plots for systems in series, you can add Bode plots of an open-loop plant and potential compensators to determine the frequency response characteristics of the dynamic system. Bode plots also illustrate the system bandwidth as the frequency at which the output magnitude is reduced by three decibels or attenuated to approximately 70.7% of its original value. You also can use the CD Bandwidth VI to determine the system bandwidth.

You can measure how close a system is to instability by examining the value of the magnitude and phase at critical values. These values, gain margin and phase margins, are important because real-life models and controllers are prone to uncertainties. Low gain or phase margins indicate potential instability.

The following sections provide information about gain and phase margins.

Gain Margin

The gain margin indicates how much you can increase the gain before the closed-loop system becomes unstable. This critical gain value, which causes instability, indicates the location of the closed-loop poles of the system on the imaginary axis.

You often use this analysis on systems where $G(s)$ consists of a gain K and a dynamic model $H(s)$ in series. For cases where increasing the gain leads to system instability, the system is stable for a given value of K only if the magnitude of $KH(s)$ is less than 0 dB at any frequency where the phase of $KH(s)$ is –180°.

The Bode magnitude plot displays the gain margin as the number of decibels by which the gain exceeds zero when the phase equals –180°, as shown in Figure 7-1.

Phase Margin

The phase margin represents the amount of delay that you can add to a system before the system becomes unstable. Mathematically, the phase margin is the amount by which the phase exceeds –180° when the gain is equal to 0 dB. The phase margin also indicates how close a closed-loop system is to instability. A stable system must have a positive phase margin.

Figure 7-1 shows Bode plots with corresponding gain and phase margins.

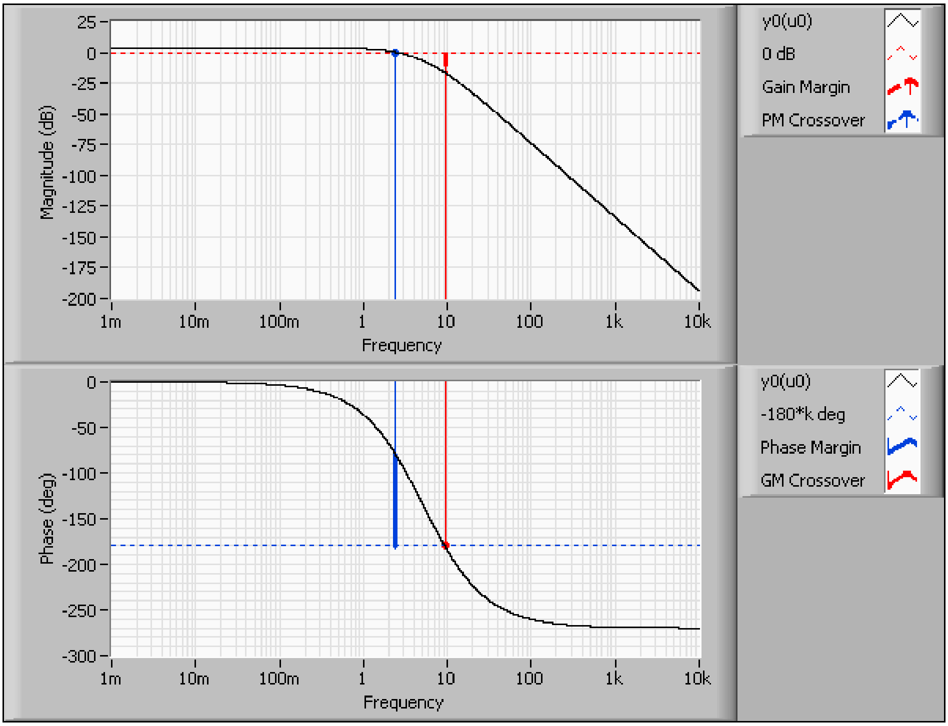

Figure 7-1. Gain and Phase Margins

Depending on the complexity of the system, a Bode plot might return multiple gain and/or phase margins.

Nichols Frequency Analysis

Use Nichols frequency analysis to obtain the closed-loop frequency response of a system from the open-loop response. Open-loop response curves, or loci, of constant magnitude and phase often provide reference points that help you analyze a Nichols plot. Each point on the open-loop response curve corresponds to the response of the system at a given frequency. You then can read the closed-loop magnitude response at that frequency from the Nichols plot by identifying the value of the magnitude locus at which the point on the curve intersects. Similarly, you can determine the closed-loop phase by identifying the phase locus at which the open-loop curve crosses.

Use the CD Nichols VI to create a Nichols plot and examine system performance in dynamic systems. The CD Nichols VI calculates and plots the open-loop frequency response against the gain and phase on the Nichols plot. Different points on the plot correspond to different values of the frequency ω. Examine the Nichols plot to determine the gain and phase margins, bandwidth, and the effect of gain variations on the closed-loop system behavior.

Nyquist Stability Analysis

Use Nyquist stability analysis to examine the system performance of dynamic systems. Nyquist plots consist of the real part of the frequency response plotted against the imaginary part of the response. Nyquist plots indicate the stability of a closed-loop system, given an open-loop system, which includes a gain of K. Use the CD Nyquist VI to create a Nyquist plot.

The Nyquist stability criterion relates the number of closed-loop poles of the system to the open-loop frequency response. On the Nyquist plot, the number of encirclements around (–1, 0) is equal to the number of unstable closed-loop poles minus the number of unstable open-loop poles.

You can use this criterion to determine how many encirclements the plant requires for closed-loop stability. For example, if the plant has all open-loop stable poles, there are no encirclements. If the plant has one

open-loop unstable pole, there is one negative, counter-clockwise encirclement. Figure 7-2 shows a system with one unstable pole.

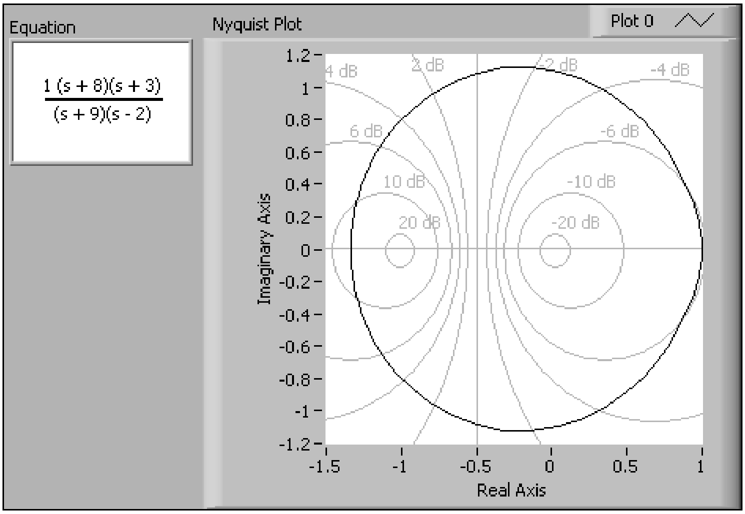

Figure 7-2. Nyquist Plot of One Unstable Pole

Often you want to determine a range of gain values for which the system is stable, rather than testing the stability of the system at a specific value of *K*. To determine the stability of a closed-loop system, you must determine how a range of gain values affect the stability of the system.

Consider the following closed-loop transfer function equation with output $Y(s)$ and input $U(s)$, where K is the gain.

$$\frac{Y(s)}{U(s)} = \frac{KH(s)}{1 + KH(s)}$$

The closed-loop poles are the roots of the equation $1 + KH(s) = 0$. The complex frequency response of $KH(s)$, evaluated for $s = i\omega$ in continuous systems and $e^{i\omega T}$ for discrete systems, encircles $(-1, 0)$ in the complex plane if $1 + KH(s)$ encircles $(0, 0)$. If you examine the Nyquist plot of $H(s)$, you can see that an encirclement of $(-1/K, 0)$ by $H(s)$ is the same as an encirclement of $(-1, 0)$ by $KH(s)$. Thus, you can use one Nyquist plot to determine the stability of a system for any and all values of K.

Obtaining Frequency Response Data

The Frequency Response VIs discussed in this chapter return frequency response data that contains information about the frequency response of all input-output pairs in the model. The frequency response information for the CD Bode VI returns information about the Bode magnitude and Bode phase. The frequency response information for the CD Nichols VI returns information about the real and imaginary parts of the frequency response. The frequency response information for the CD Nyquist VI returns information about the open-loop gain and open-loop phase. Use the CD Get Frequency Response Data VI to access this information for a specified input-output pair, a list of input-output pairs, or all input-output pairs of the system.

The CD Get Frequency Response Data VI uses the **Frequency Response Data** input, which contains the frequency response information for all the input-output pairs of a system model. For state-space models, the CD Get Frequency Response Data VI returns the frequency response of the input-state pair(s). Because transfer function and zero-pole-gain models do not have states, the frequency response data for an input-state pair of these forms is an empty array.

Refer to the *LabVIEW Help,* available by selecting **Help»VI, Function, & How-To Help**, for more information about using the CD Get Frequency Response Data VI.

8

Analyzing Dynamic Characteristics

Any given dynamic system has numerous dynamic characteristics such as stability, DC gain, damping ratio, natural frequency, and norm. You can use the LabVIEW Control Design Toolkit to analyze a system in terms of these characteristics.

This chapter provides information about using the Control Design Toolkit to analyze the stability of a dynamic system. This chapter also describes how to use the root locus method to analyze the stability of a system.

Determining Stability

The stability of a system depends on the locations of the poles and zeroes within the system. To design an effective controller, you must take these locations into account.

A continuous system is stable if all poles are on the left half of the complex plane. A discrete system is stable if all poles are within a unit circle centered at the origin of the complex plane. Additionally, both types of systems are stable if they do not contain any poles.

A continuous system is unstable if it contains at least one pole in the right half of the complex plane. A discrete system is unstable if at least one pole is outside of the unit circle in the complex plane. Additionally, both types of systems are unstable if they contain more than one pole at the origin.

In terms of the dynamic response associated with the poles and zeroes of a system, a pole is stable if the response of the pole decays over time. If the response becomes larger over time, the pole is unstable. If the response remains unchanged over time, the pole is marginally stable. To describe a system as stable, all the closed-loop poles of a system must be stable.

Continuous and discrete systems are marginally stable if they contain only one pole at the origin and no positive poles.

Use the CD Pole-Zero Map VI to obtain all the poles and zeroes of a system and plot their corresponding locations in the complex plane. Use the CD Stability VI to determine if a system is stable, unstable, or marginally stable.

Using the Root Locus Method

The root locus method provides the closed-loop pole positions for all possible changes in the loop gain K. Root locus plots provide an important indication of what gain ranges you can use to keep the closed-loop system stable. The root locus is a plot on the real-imaginary axis showing the values of s that correspond to pole locations for all gains, starting at the open-loop poles, $K = 0$ and ending at $K = \infty$.

You can rewrite the characteristic equation of a closed-loop system using the following equation, where $N(s)$ is the numerator and $D(s)$ is the denominator.

$$1 + KH(s) = D(s) + KN(s) = 0$$

This equation restates the fact that the open-loop system poles, which correspond to $K = 0$, are the roots of the transfer function denominator, $D(s)$. As K becomes larger, the roots of the previous characteristic equation approach either the roots of $N(s)$, the zeroes of the open-loop system, or infinity. For a closed-loop system with a non-zero, finite gain K, the solutions to the preceding equation are given by the values of s that satisfy both of the following conditions:

$$|KH(s)| = 1 \qquad \angle H(s) = \pm(2k+1)\pi \qquad (k = 0, 1, \ldots)$$

Use the CD Root Locus VI to compute and draw root locus plots for continuous and discrete SISO models of any form. You also can use this VI to synthesize a controller. Refer to the *The Root Locus Design Technique* section of Chapter 11, *Designing Classical Controllers*, for information about using the CD Root Locus VI to design a controller.

9

Analyzing State-Space Characteristics

State-space analysis involves analyzing the state variables of a system. State variables describe the relationship between the inputs and outputs of a system. These variables often have physical meaning and represent some internal state of the system under analysis. For example, consider a motor that has power as its input and speed as its output. If you represent this system as a state-space model, the state variables are speed and rotation angle.

To design an effective controller, you must perform a state-space analysis on the controller model. State-space analysis determines whether a system is stable, controllable, or observable. You can use state-space analysis to balance a system model. Balancing a system model is useful in both analyzing and synthesizing a controller. You also can use state-space analysis to define different representations of the same system.

Because you can choose a variety of state variables to represent a single system, the state-space form for a given linear time-invariant multiple-input multiple-output (MIMO) system is not unique. You must determine which state variables are best for the analysis and design of a state-space controller.

This chapter provides information about using the LabVIEW Control Design Toolkit to perform state-space analysis.

Determining Stability

In a state-space form, the time evolution of the states determines the stability of the system. If you have initial conditions and you eliminate all inputs to the system, only the state matrix $\boldsymbol{A}$ governs the response of the system. You then apply control theory to find the counterparts of poles, which you can use in transfer function and pole-zero analysis.

The counterparts of poles are the eigenvalues of the state matrix $\boldsymbol{A}$. The location of these eigenvalues determines the stability of the system.

A continuous system is stable if all eigenvalues of $\boldsymbol{A}$ have negative real parts. A discrete system is stable if these eigenvalues fall within the unit circle.

Determining Controllability

A system is controllable if all the states that describe the system respond to an input of the system, that is, you can influence the states of the system independently by adjusting the inputs. A system is not controllable if the system contains states that remain unaffected by any input.

If a system is controllable, there is an input that forces the system states, or linear combination of states, to go from any initial condition at $t = 0$ to zero at any time $t > 0$. If a system is open-loop unstable, you can adjust the input to affect the response of the states.

You can confirm the controllability of a system by verifying that the controllability matrix $\boldsymbol{Q}$, shown in the following equation, has full row rank or is nonsingular.

The state matrix $\boldsymbol{A}$ and the input matrix $\boldsymbol{B}$ determine the controllability properties of a state-space model. You use these matrices to calculate $\boldsymbol{Q}$, as shown in the following equation:

$$\boldsymbol{Q} = [\boldsymbol{B} \ \ \boldsymbol{AB} \ \ldots \ \boldsymbol{A}^{n-1}\boldsymbol{B}]$$

A system is controllable if $\boldsymbol{Q}$ has full row rank or is nonsingular.
For example, if $\boldsymbol{B}$ is an n-dimensional column vector that is colinear to an eigenvector of null eigenvalues of $\boldsymbol{A}$, you obtain the following matrix:

$$\boldsymbol{Q} = [\boldsymbol{B} \ \ 0 \ \ 0 \ \ldots \ 0]$$

This matrix is row rank deficient for $n > 1$. The null eigenvalue represents an uncontrollable mode of the system.

From the definition of a controllable system you can conclude that to place the system states at zero at any time $t > 0$ indicates that you can place all system poles anywhere to make the closed-loop response reach zero at time t as quickly as possible.

When you can adjust all system poles locations to a point you want, you can calculate a full state-feedback controller gain K to arbitrarily place the eigenvalues of the closed-loop system, $\boldsymbol{A}' = \boldsymbol{A} - \boldsymbol{B}K$. Conversely, the

eigenvalues associated with modes that are not controllable cannot be adjusted, regardless of the value you choose for K.

Use the CD Controllability Matrix VI to calculate the controllability matrix of the model and determine if the system is controllable. Use the CD Controllability Staircase VI to transform a state-space model into a model that you can use to identify controllable states in the system. You also can use the CD Controllability Staircase VI to inspect the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices of the transformed model to determine the controllable states.

Determining Observability

A system is observable if you can estimate each state of the system by looking only at the output response. If you can determine the states at time t_0 by observing the output from time t_0 to t_1, the system is observable.

Observability depends on the output matrix $\boldsymbol{C}$ and the state matrix $\boldsymbol{A}$ of the system. You can check observability by verifying that the observability matrix $\boldsymbol{O}$, defined in the following equation, is full column rank or is nonsingular for a SISO system.

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \vdots \\ \boldsymbol{CA}^{n-1} \end{bmatrix}$$

Use a state estimator to calculate the states of any observable system with a column-deficient matrix $\boldsymbol{C}$. Refer to Chapter 13, *Defining State Estimator Structures*, for information about state estimators.

Use the CD Observability Matrix VI to calculate the observability matrix of a model and determine if the system is observable. Use the CD Observability Staircase VI to transform a state-space model into a model that you can use to identify observable states in the system. Use the CD Observability Staircase VI to calculate the observability matrix of the transformed model. You also can use the CD Observability Staircase VI to inspect the $\boldsymbol{A}$ and $\boldsymbol{C}$ matrices of the transformed model to determine the observable states.

Analyzing Controllability and Observability Grammians

An alternative and numerically more stable approach to assessing controllability and observability is to compute the Grammians of the state-space matrices. The controllability Grammian is an $n \times n$ matrix that determines how dependent the state responses are on the different inputs of the system. Independent state responses indicate that there always is a set of inputs that can drive the states to zero at a certain time. In this case, the system is controllable.

Calculate the eigenvalues of the controllability Grammian to check the dependency of the state responses. If the controllability Grammian is positive-definite, meaning all eigenvalues are real and greater than zero, the chosen state-space form is controllable.

Similarly, the observability Grammian is an $n \times n$ matrix that determines how dependent the state effects are on the different outputs of the system. Independent state effects indicate that there always is a set of outputs that you can use to estimate the states at time $t = 0$. In this case, the system is observable.

Calculate the eigenvalues of the observability Grammian to check the dependency of the responses of the states. If the observability Grammian is positive-definite, meaning all eigenvalues are real and greater than zero, the chosen state-space form is observable.

Use the CD Grammians VI to calculate the controllability and observability Grammians of a state-space model for a stable system.

Balancing Systems

A system is balanced if the controllability and observability diagonal Grammians of that system are identical. A balanced model simplifies the analysis and use of model order reduction. Refer to Chapter 10, *Model Order Reduction*, for more information about model order reduction.

In model order reduction, balancing highlights the relative importance of the state to the input/output performance of the system. Balancing consists of finding a similarity transformation from the original model to generate a state-space representation. Use the CD Balance State-Space Model (Diagonal) VI and the CD Balance State-Space Model (Grammians) VI to balance a state-space system.

If you use the CD Balance State-Space Model (Grammians) VI, the **Balanced Model** output of this VI has equal and diagonal controllability and observability Grammians. To use this VI, the system must be stable, controllable, and observable.

If you use the CD Balance State-Space Model (Diagonal) VI, the balanced state-space model has an even eigenvalue spread for the state matrix $\boldsymbol{A}$ or the composite matrix, which contains the natural composition of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$.

10

Model Order Reduction

In most cases, different models of a dynamic system can represent the same input-output behavior of that system. For example, you can have two state-space models with different numbers of states that represent the same input-output behavior at varying degrees of accuracy. Often you can simplify, or reduce, these models to obtain a less complicated representation of the system.

How you reduce a model depends on the representation of the model. If the model is a state-space model, reducing the number of states reduces the order of the model. If the model is a transfer function or zero-pole-gain model, canceling matching poles and zeroes reduces the order of the model. Use the Model Reduction VIs to reduce the order of a model.

This chapter provides information about the minimal realization and model order reduction techniques you can use to simplify a model.

Obtaining the Minimal Realization of Models

The minimal realization of a system model involves cancelling all pairs of poles and zeroes at the same location. You refer to these pairs as pole-zero pairs. Use the CD Minimal Realization VI to calculate the minimal realization of a model.

For example, consider the following transfer function model $H(s)$.

$$H(s) = \frac{s^2 + 6s + 8}{s^3 - 8s^2 - 21s + 108} = \frac{(s+2)(s+4)}{(s+4)(s-3)(s-9)} = \left.\frac{(s+2)}{(s-3)(s-9)}\right\} \textit{Minimal Realization}$$

This model has a pole and zero in the same location, –4. Wire this model into the CD Minimal Realization VI to cancel this pole-zero pair. This VI returns the minimal realization of the model in the **Reduced Model** output. This VI also returns the number of pole-zero locations removed. For state-space models, this VI returns the number of states removed.

Minimal realizations are minimal because the only modes represented in the model are those modes that you can infer by observing the inputs and outputs of the system. The modes that you eliminate to obtain a minimal transfer function or zero-pole-gain model still exist in the system, but you cannot infer their existence by simply observing the input and outputs of the model. For this reason, you do not want to cancel unstable pole-zero pairs.

For example, consider the following transfer function model $G(s)$.

$$G(s) = \frac{s^2 - 2s - 8}{s^3 - 16s^2 + 75s - 108} = \frac{(s+2)(s-4)}{(s-4)(s-3)(s-9)} = \left.\frac{(s+2)}{(s-3)(s-9)}\right\} \textit{Minimal Realization}$$

$G(s)$ has the same minimal realization as $H(s)$, but $G(s)$ contains an unstable pole-zero pair at 4. If you cancel this pole-zero pair, you no longer can observe any effects the pair has on the stability of the system.

A minimal realization for a state-space model is a state-space representation in which you remove all states that are not observable or controllable. Use the CD Minimal State Realization VI to determine the minimal realization for a state-space model. Refer to Chapter 9, *Analyzing State-Space Characteristics*, for information about controllability and observability.

Reducing the Order of Models

In certain situations, you might want to work with a lower-order model of the system. The goal of model order reduction is to remove stable states that have the smallest impact on the input-output model representation. You might want to reduce a model order when the real part of stable system poles differ significantly. From an input-output standpoint, you usually ignore fast dynamic modes, which are modes that correspond to stable eigenvalues far from the imaginary axis, because you only see the effects of these modes over a short initial period of time. Use the CD Model Order Reduction VI to reduce high-order models.

Note Model order reduction applies only to a state-space model of a system.

You can reduce the order of the model by decreasing the order of the stable modes. Reducing stable modes of the model does not affect the unstable modes of the model.

You have several options for reducing the order of a model. You can match the DC gain between the reduced order model and the original model. You also can delete the states directly.

Balancing the original state-space model can make the model order reduction process easier. When you balance the state-space model, the Grammian matrices are diagonal and you avoid computing the eigenvalues.

Given a state-space model, complete the following steps to reduce the model order:

1. Balance the state-space model.
2. Compute the Grammians.
3. Remove stable states corresponding to small eigenvalues, in proportion to the other eigenvalues, of the Grammian matrix.
4. Repeat steps 1 through 3 until the model is of the order you want.

Refer to the *Analyzing Controllability and Observability Grammians* section and the *Balancing Systems* section of Chapter 9, *Analyzing State-Space Characteristics*, for more information about computing controllability and observability Grammians and balancing a model.

Selecting and Removing an Input, Output, or State

Manipulating the system representation involves ignoring certain inputs and outputs of a model, such as those connected by a unit gain. In a state-space model, manipulating the system representation involves removing unwanted states from the description. Use the CD Select IO from Model VI and the CD Remove IO from Model VI to reduce a model by directly removing inputs, outputs, or states.

Manipulating a model is useful for building new models from old ones and for quickly removing zero states from a large state-space model representation. Zero states are states for which the state matrix $\boldsymbol{A}$ has zeroes in an input row and the corresponding output column. Use the CD Minimal State Realization VI to perform this operation. Figure 10-1 shows an example of a zero-state.

$$
\underbrace{\begin{bmatrix} & 0 & & & \\ 0 & 0 & 0 & \cdots & 0 \\ & \vdots & & & \\ & 0 & & & \\ & 0 & & & \end{bmatrix}}_{= \boldsymbol{A}}
\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ \\ \end{bmatrix}
$$

Figure 10-1. A Zero State in ***A***

If the matrix has no zero rows or columns, consider using another method to reduce the model order.

Note When you work with transfer function and zero-pole-gain models, you generally do not select and remove specific inputs and outputs to reduce the model order. You mainly use this method with state-space models.

11

Designing Classical Controllers

Classical control design involves creating controllers based on the input-output behavior of a system. In classical control design, you select one or more specific gain values to achieve one or more control objectives. The first step in designing a controller is identifying a control objective. For example, you might focus on the rise time, overshoot, and damping ratio of a controller model. Based on this objective, you specify the location of the poles of the system. You then select an appropriate set of parameters, such as the gain, to satisfy the stated objectives. You use these parameters to design a controller.

This chapter provides information about using the LabVIEW Control Design Toolkit to implement the root locus design technique. This chapter also describes the proportional integral derivative (PID) controller.

The Root Locus Design Technique

Root locus is a technique that shows how the roots of a system vary with respect to the gain K. Taking into account a control objective, you decide on the locations of the roots of the system. From the locations of these roots, you infer the optimal value of K. You then can use the gain K to design a controller for a single-input single-output (SISO) system. Use the CD Root Locus VI to apply the root locus technique to a system.

You can use the root locus technique to design SISO systems by analyzing the variation of closed-loop pole positions for all possible changes in a controller variable. The closed-loop zeroes of a system, between any two points in the control system, are a subset of the open-loop zeroes and poles of the feedback element. The root locus plot depicts the path that the roots follow as you vary the gain. You use this relationship to analyze the closed-loop behavior in terms of the value of a variable in the feedback transfer function.

For example, consider a system with the following open-loop transfer function:

$$H(s) = \frac{1}{(s+1)(s+2)(s+3)}$$

If a simple proportional feedback controller controls this system, the following equation describes the characteristic equation.

$$1 + H(s)K = 1 + \frac{K}{(s+1)(s+2)(s+3)} = 0$$

Figure 11-1 illustrates the root locus plot of this system.

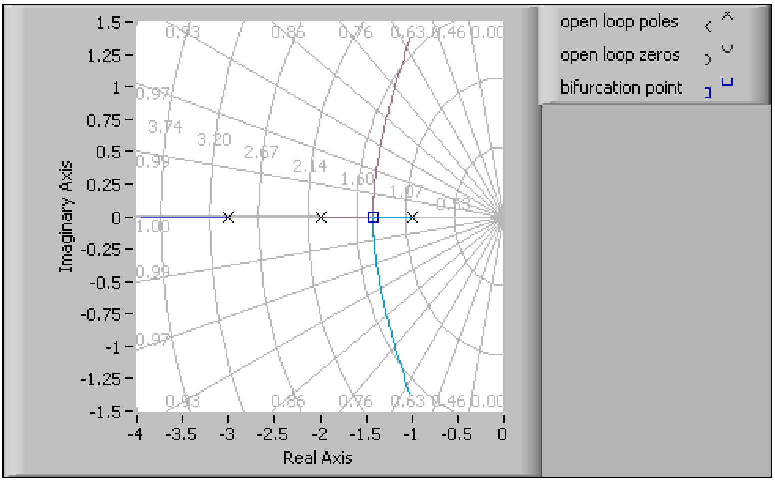

Figure 11-1. Root Locus

This graph shows the locations of the closed-loop poles. The pole locations are –1, –2, and –3.

You can use root locus design to synthesize a variety of different controller configurations, including the following types:

- Lead compensator—Lowers the rise time and decreases the transient overshoot.
- Lag compensator—Improves the steady-state accuracy of the system.

- Notch compensator—Achieves stability in system with lightly damped flexible modes. This compensator adds a zero near the resonance point of the flexible mode.
- Proportional Integral Derivative (PID) controller—Forms a controller using the most common architecture. Refer to the *The Proportional Integral Derivative Controller Architecture* section of this chapter for more information about PID controllers.

The difference in these controller configurations is the form of the transfer function equations you use to synthesize the controller. Different transfer function models result in different dynamic characteristics of the controlled system.

For example, consider a controller transfer function model $D(s)$ defined by the form of the following equation:

$$D(s) = K\frac{s+z}{s+p}$$

If $z < p$, this transfer function results in a lead compensator. If $z > p$, this transfer function results in a lag compensator. You typically place this lead compensator in series with the plant $H(s)$ in the feed-forward path.

Refer to the CDEx Interactive Root Locus VI in the `labview\examples\Control Design\Getting Started.llb` for an example that demonstrates root locus analysis.

You also can use other frequency domain tools, such as Bode, Nyquist, and Nichols plots, to design a system. These plots show the specific locations and shape of key points. You examine these locations to iteratively modify the controller parameters to meet these specifications. The number and nature of the controller parameters depends on the topology of the controller. Refer to *Feedback Control of Dynamic Systems*[1] and *Modern Control Engineering*[2] for more information about the using the root locus technique to design controllers.

1 Franklin, Gene F., J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2002.

2 Ogata, Katsuhiko. *Modern Control Engineering*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2001.

The Proportional Integral Derivative Controller Architecture

The PID controller, also known as the three-term controller, is the most widely-used controller architecture. PID controllers compare the output against the reference input and initiate the appropriate corrective action. PID controllers combine proportional P, integral I, and derivative D compensation. Use the CD Construct Special Model VI to construct a PID controller.

The following equation defines control action for a general PID controller.

$$u(t) = K_c\left[e(t) + \frac{1}{\tau_I}\int_0^t e(t^*)dt^* + \tau_d\frac{de(t)}{dt}\right]$$

In this equation, K_c is the gain, τ_d is the derivative time constant, and τ_I is the integral time constant. The following equation defines the error.

$$e(t) = R(t) - B(t)$$

In this equation, $R(t)$ is the reference input and $B(t)$ is the output.

Because the control action is a function of the error, the following equation defines the transfer function for the PID controller.

$$\frac{U(s)}{E(s)} = K_c\left(1 + \frac{1}{\tau_I s} + \tau_d s\right)$$

This transfer function is improper, which means the transfer function has more zeroes than poles. You cannot physically realize an improper transfer function. You can place a pole at $-1/\alpha\tau_d$ to make the transfer function proper. α is a small number, typically between 0.05 and 0.2, such that the pole has negligible effect on the system dynamics.

The Control Design Toolkit supports the PID controller in the following three forms: PID Academic, PID Parallel, and PID Serial. Table 11-1 shows the equations for each of these forms.

Table 11-1. PID Controller Forms in the Control Design Toolkit

PID Controller Form	Equation
PID Academic	$\frac{U(s)}{E(s)} = K_c\left(1 + \frac{1}{T_i s} + \frac{T_d s}{\alpha T_d s + 1}\right)$
PID Parallel	$\frac{U(s)}{E(s)} = K_c + \frac{K_i}{s} + \frac{\tau K_d s}{\alpha K_d s + 1}$
PID Series	$\frac{U(s)}{E(s)} = K_c\left(1 + \frac{1}{T_i s}\right)\left(\frac{T_d s + 1}{\alpha T_d s + 1}\right)$

Each PID form produces the same result but incorporates information in a different manner. For example, you can adjust each term independently using the PID Parallel form. The PID form you use depends on the design decisions you make, such as how you need to manipulate the output of the controller. Use the polymorphic VI selector of the CD Construct Special Model VI to implement a PID controller using one of these three PID forms.

Note In some applications, you specify the gain in the PID Academic transfer function in terms of a proportional band (PB).

$$PB = \frac{1}{K_c} \times 100\%$$

A proportional band, defined by the previous equation, is the percentage of the input range of the controller that causes a change equal to the maximum range of the output.

You can use the root locus and Bode design methods to determine appropriate gain values for the PID controller. Refer to *PID Controllers: Theory, Design, and Tuning*[1] for more information about these techniques. Refer to the *LabVIEW PID Control Toolset User Manual* for more information about experimentally determining controller gain parameters.

[1] Astrom, K. and T. Hagglund. *PID Controllers: Theory, Design, and Tuning*, 2nd ed. ISA, 1995.

12 Designing State-Space Controllers

State-space control design involves creating controllers based on the relationship between the inputs, states, and outputs of a system. State-space control techniques use state-space models to synthesize and analyze controllers.

Because all states are not directly measurable, you sometimes need to use an estimator. An estimator infers the states with which you are working, based on measurements of the outputs and known states.

Note Estimator, unless otherwise noted, also refers to an observer.

Similar to classical control design, the process of designing a controller begins with one or more control objectives. Typical objectives include minimizing a cost function and placing the poles and zeroes of a system in specific locations. You use this process to achieve a specific dynamic response. You then select the architecture of the controller, such as whether the feedback is based only on outputs or on all the states of the system. With this information, you can synthesize a controller by selecting an appropriate set of parameters to satisfy the stated objectives.

This chapter provides information about using the Control Design Toolkit to determine estimator and controller gain matrix values. This chapter also describes the difference between measured outputs, known inputs, and adjustable inputs.

Calculating Estimator and Controller Gain Matrices

Before you can implement an estimator or a controller, you need to calculate their respective gain matrices. These gain matrices define the structure of the estimator or the controller. The Control Design VIs help you calculate the gain matrix for an estimator or controller.

The following sections provide information about using the Control Design Toolkit to perform the pole placement technique and design a linear quadratic regulator. The following sections also describe how to use the Kalman gain function and how to construct a linear quadratic Gaussian controller.

Pole Placement Technique

Pole placement is a technique in which you specify the locations of the closed-loop poles of a system and calculate the gain matrix based on these locations. You can use the pole placement technique to calculate either the observer gain matrix $\boldsymbol{L}$ or the controller gain matrix $\boldsymbol{K}$.

Use the CD Ackermann VI to apply this technique in the following situations:

- A single-input single-output (SISO) system
- A single-input multiple-output (SIMO) system if you are defining the controller gain matrix $\boldsymbol{K}$
- A multiple-input single-output (MISO) system if you are defining the observer gain matrix $\boldsymbol{L}$

Use the CD Pole Placement VI in all other situations, for example, a multiple-input multiple-output (MIMO) system. The computation of the gain for these systems is more complex and based on a Sylvester matrix equation. Refer to the *LabVIEW Control Design Toolkit Algorithm Reference* manual for information about the Sylvester matrix equation.

Use the **Gain Type** parameter of the CD Ackermann VI and the CD Pole Placement VI to determine which kind of gain matrix these VIs return. This section uses the controller gain matrix $\boldsymbol{K}$ as an example.

Note The Control Design Toolkit refers to the pole placement technique as an observer, because this technique does not estimate measurements given random noise. This distinction does not affect the interaction between the CD Ackermann or CD Pole Placement VIs and other VIs.

Consider the following SISO state-space system with $u = -\boldsymbol{Kx}$ as the control action.

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{B}u$$
$$y = \boldsymbol{Cx} + \boldsymbol{D}u$$

Figure 12-1 shows how you apply the gain matrix $\boldsymbol{K}$ to a controller.

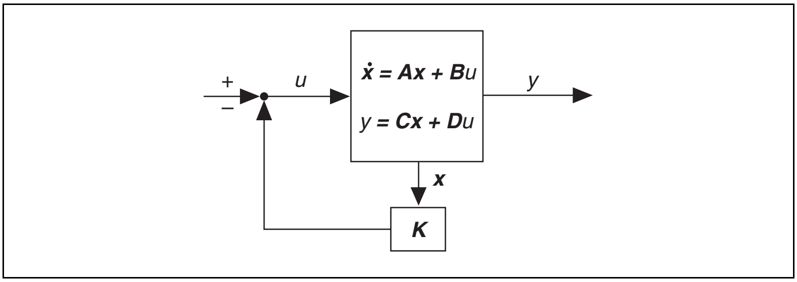

Figure 12-1. Using ***K*** to Regulate the Input of a State-Feedback System

Given a specification of the closed-loop pole locations, $\lambda_1, \lambda_2, \ldots \lambda_n$, you can calculate the controller gain matrix $\boldsymbol{K}$ that achieves this goal. The system in question must be controllable.

For example, consider a closed-loop continuous system that has the following form:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$$
$$\tilde{\boldsymbol{A}} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}$$

Because $\tilde{\boldsymbol{A}}$ satisfies the characteristic polynomial equation that the specified closed-loop pole locations $\lambda_1, \lambda_2, \ldots \lambda_n$ define, you can state the following relationships:

$$s^n + \alpha_1 s^{n-1} + \ldots + \alpha_{n-1} s + \alpha_n \boldsymbol{I} \equiv (s - \lambda_1)(s - \lambda_2)\ldots(s - \lambda_n)$$

$$\phi(\tilde{\boldsymbol{A}}) = \tilde{\boldsymbol{A}}^n + \alpha_1 \tilde{\boldsymbol{A}}^{n-1} + \ldots + \alpha_{n-1}\tilde{\boldsymbol{A}} + \alpha_n \boldsymbol{I} = 0$$

The locations of α_n are based on the locations of λ_n. s is the Laplace variable. You can use these equations to calculate Ackermann's formula, defined by the following equation:

$$\boldsymbol{K} = \begin{bmatrix} 0 & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \ldots & \boldsymbol{A}^{n-1}\boldsymbol{B} \end{bmatrix}^{-1} \phi(\tilde{\boldsymbol{A}})$$

Combine the controller gain matrix $\boldsymbol{K}$ with the CD State-Space Controller VI to define a controller structure for the system. Refer to Chapter 14, *Defining State-Space Controller Structures*, for more information about defining a controller structure. If you use the pole placement technique to calculate the estimator gain matrix $\boldsymbol{L}$, combine $\boldsymbol{L}$ with the CD State

Estimator VI to define an estimator structure for the system. Refer to Chapter 13, *Defining State Estimator Structures*, for more information about defining an estimator structure.

Linear Quadratic Regulator Technique

The linear quadratic regulator (LQR) technique minimizes a quadratic control objective and calculates the controller gain matrix ***K***. Unlike the pole placement technique, you cannot use the LQR technique to calculate a estimator gain matrix ***L***. Use the CD Linear Quadratic Regulator VI to apply the LQR technique to a SISO, SIMO, MISO, or MIMO system.

The design process for LQR requires that you specify matrices ***Q*** and ***R***, which specify the weights on the states and inputs, respectively. You must also specify a matrix ***N***, which penalizes the cross product between the inputs and states. Typically, the selection of these gain matrices is an iterative process. You then use the LQR technique to calculate the gain matrix ***K*** required to control the system.

For a continuous state-space model, the LQR technique minimizes a control objective of the following form:

$$J(u) = \int_0^{\infty} [\boldsymbol{x}^T(t)\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}\boldsymbol{u}(t) + 2\boldsymbol{x}^T(t)\boldsymbol{N}\boldsymbol{u}(t)]dt$$

Q is a symmetric, positive, semi-definite matrix that penalizes the state vector ***x*** in the control objective. ***R*** is a positive definite matrix, usually symmetric, that penalizes the input vector ***u*** in the control objective. ***N*** is a symmetric, positive, semi-definite matrix that penalizes the cross product between input and state vectors.

Combine the controller gain matrix ***K*** with the CD State-Space Controller VI to define a controller structure for the system. Refer to Chapter 14, *Defining State-Space Controller Structures*, for more information about defining a controller structure.

Kalman Gain Function

The Kalman gain function calculates the estimator gain matrix ***L***. Unlike the pole placement technique, you cannot use the Kalman gain function to calculate a controller gain matrix ***K***. Use the CD Kalman Gain VI to apply this function to a given system.

Note The Control Design Toolkit refers to the Kalman gain function as an estimator, because this function estimates measurements given random noise. This distinction does not affect the interaction between the CD Kalman Gain VI and other VIs.

Consider the following continuous state-space system where w is the process noise, $\boldsymbol{v}$ is the measurement noise, and $\boldsymbol{u}$ is the known input.

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu} + \boldsymbol{Gw}$$
$$\boldsymbol{y} = \boldsymbol{Cx} + \boldsymbol{Du} + \boldsymbol{Hw} + \boldsymbol{v}$$

The following matrices define the covariance of these signals, or how long the error of the estimator takes to reach zero. The function E is the expected output you receive when you perform the included operations.

$$E(\boldsymbol{ww}^T) = \boldsymbol{Q}$$
$$E(\boldsymbol{vv}^T) = \boldsymbol{R}$$
$$E(\boldsymbol{wv}^T) = \boldsymbol{N}$$

If you use $\hat{\boldsymbol{x}}$ to denote the estimated values of the state, the estimator updates the state-space equation as follows:

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{Bu} + \boldsymbol{L}(\boldsymbol{y} - \hat{\boldsymbol{y}})$$

Combine the estimator gain matrix $\boldsymbol{L}$ with the CD State Estimator VI to define an estimator structure for the system. Refer to Chapter 13, *Defining State Estimator Structures*, for more information about defining an estimator structure.

Linear Quadratic Gaussian Controller

A linear quadratic Gaussian (LQG) controller utilizes the LQR technique to build the controller and the Kalman gain technique to filter out any system noise. Use the CD Linear Quadratic Regulator VI and the CD Kalman Gain VI together with the CD State-Space Controller VI to synthesize a LQG controller.

Using an arbitrary estimator with a design such as LQR might not result in the most optimal design of the controller. If the estimator starts with the same initial condition as the unmeasured states, $\hat{\boldsymbol{x}}(0) \equiv \boldsymbol{x}(0)$, and if the system satisfies a number of controllability and observability conditions, the closed-loop system with the observer based controller has the same response as the LQR design. This form of state feedback controller, when

combined with a estimator defined with the Kalman gain function, is called the LQG controller.

Certainty equivalence is the property that enables this combined usage of optimal estimator and controller. Certainty equivalence is important because you can synthesize a controller gain matrix ***K*** and estimator gain matrix ***L*** independently. You can build a controller assuming all states are measurable and then estimate unmeasured states using an optimal estimator. The resulting design is optimal for the specified problem.

Note Because a LQG controller uses an estimator, the robustness properties of a LQG controller are not the same as that of a LQR controller. You have no guarantee that robustness properties can be established for an estimated state feedback controller. You only can guarantee robustness by changing the way you measure the states of the system to remove the need for an estimator.

13 Defining State Estimator Structures

State estimators reconstruct unmeasurable state information. To define the structure of a state estimator, you need a model of the system and an estimator gain matrix $\boldsymbol{L}$. You can calculate $\boldsymbol{L}$ using the CD Pole Placement VI, the CD Ackermann VI, or the CD Kalman Gain VI. Refer to Chapter 12, *Designing State-Space Controllers*, for more information about these VIs.

You use $\boldsymbol{L}$ to define the structure of an estimator. You can design an estimator structure to take various factors, such as input noise or input disturbances, into consideration.

This chapter provides information about using the LabVIEW Control Design Toolkit to define the structure of a state estimator. This chapter also discusses known inputs and measurable outputs.

Measuring and Adjusting Inputs and Outputs

The estimator gain $\boldsymbol{L}$ considers all inputs $\boldsymbol{u}$ and outputs $\boldsymbol{y}$, which are known and measured. Also, some inputs and outputs might be unavailable. So you can divide the system into adjustable inputs, measured outputs, unknown inputs, and unmeasured outputs. You base this division on diagonal matrices, such as Λ_u and Λ_y.

Diagonal matrices incorporate the effect of known, unknown, measured, and unmeasured inputs and outputs into the equation. A diagonal element in these matrices equals unity for the known and measured inputs and outputs, and zero for the unknown and unmeasured inputs and outputs or states. The following equation describes how you incorporate the diagonal elements for the inputs and outputs in the controller model.

$$\begin{aligned} \hat{\boldsymbol{x}} &= \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}^*\boldsymbol{u} + \boldsymbol{L}^*(\boldsymbol{y} - \hat{\boldsymbol{y}}) \\ \hat{\boldsymbol{y}} &= \boldsymbol{C}\hat{\boldsymbol{x}} + \boldsymbol{D}^*\boldsymbol{u} \end{aligned}$$

In this equation, $\boldsymbol{B}^* = \boldsymbol{B}\Lambda_u$, $\boldsymbol{D}^* = \boldsymbol{D}\Lambda_u$, and $\boldsymbol{L}^* = \boldsymbol{L}\Lambda_y$. These substitutions apply to both estimators and controllers. Controllers have an additional substitution when inputs are not adjustable. For a controller, the controller gain $\boldsymbol{K}^*$ is given by $\boldsymbol{K}^* = \boldsymbol{K}\Lambda_z$, where Λ_z is a diagonal matrix with the same characteristics as Λ_u and Λ_y. Therefore, a diagonal element in Λ_z equals unity for the adjustable input, and zero for the nonadjustable or system disturbances.

By default, matrices Λ_u and Λ_z are identity matrices whose size equals the number of inputs. Λ_y is an identity matrix whose size equals the number of outputs.

Adding a State Estimator to a General System Configuration

Use the CD State Estimator VI to define an estimator structure. This VI integrates $\boldsymbol{L}$ into a dynamic system so you can analyze and simulate the estimator performance.

Note To simplify the equations in the rest of this chapter, assume that all inputs are known and all outputs are measurable. This assumption means $\boldsymbol{B}^* = \boldsymbol{B}$, $\boldsymbol{L}^* = \boldsymbol{L}$, and $\boldsymbol{D}^* = \boldsymbol{D}$.

Consider the following equations that represent a continuous state-space system.

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} + \boldsymbol{r}_y \end{aligned}$$

Assume that $\boldsymbol{L}$ is based on this system, some estimator performance specifications, and the output noise $\boldsymbol{r}_y$ covariance. You then can calculate the estimated states $\hat{\boldsymbol{x}}$ using the following equations for dynamic models:

$$\begin{aligned} \dot{\hat{\boldsymbol{x}}} &= \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \\ \hat{\boldsymbol{y}} &= \boldsymbol{C}\hat{\boldsymbol{x}} + \boldsymbol{D}\boldsymbol{u} \end{aligned}$$

The state-space system and dynamic model equations share the same system matrices and input $\boldsymbol{u}$. The states $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are different because the initial conditions of the system might differ from the model and because of the noise input $\boldsymbol{r}_y$. Without a noise input, however, the model states track the system states, making the difference $\boldsymbol{x} - \hat{\boldsymbol{x}}$ converge asymptotically to

zero. The following equation shows how the estimator gain $\boldsymbol{L}$ enhances the convergence of the error $\dot{\boldsymbol{e}}$ to zero.

$$\dot{\boldsymbol{e}}_x \equiv \dot{\hat{\boldsymbol{x}}} - \dot{\boldsymbol{x}} = \boldsymbol{A}(\hat{\boldsymbol{x}} - \boldsymbol{x}) + \boldsymbol{L}(\boldsymbol{y} - \hat{\boldsymbol{y}}) = (\boldsymbol{A} - \boldsymbol{L}\boldsymbol{C})\boldsymbol{e}_x + \boldsymbol{L}\boldsymbol{r}_y$$

Without the noise input, the following equation defines the error convergence.

$$\dot{\boldsymbol{e}}_x = (\boldsymbol{A} - \boldsymbol{L}\boldsymbol{C})\boldsymbol{e}_x$$

$\boldsymbol{L}$ is designed to place the poles of the matrix $\boldsymbol{A} - \boldsymbol{LC}$ in the specified complex-plane location.

To include the estimator in the composed system model, you append the original model states $\boldsymbol{x}$ to the estimated model states $\hat{\boldsymbol{x}}$. The following equations show this process:

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & 0 \\ 0 & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} & \boldsymbol{L} \\ \boldsymbol{B} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} - \hat{\boldsymbol{y}} \end{bmatrix}$$

$$\begin{bmatrix} \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} & 0 \\ 0 & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{D} & 0 \\ \boldsymbol{D} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} - \hat{\boldsymbol{y}} \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{r}_y \end{bmatrix}$$

Given this general system configuration, the following sections provide information about deriving the possible configurations of a state estimator.

Configuring State Estimators

Use the **Configuration** parameter of the CD State Estimator VI to define the structure of an estimator using one of the following three configurations:

- System Included—Appends the actual states of the system to the estimated states.
- System Included with Noise—Incorporates noise $\boldsymbol{r}_y$ into the system included configuration.
- Standalone—Defines a structure of the estimator that analyzes a system-model mismatch.

Table 13-1 summarizes the different state estimator configurations and their corresponding states, inputs, and outputs.

Table 13-1. State Estimator Configurations

Configuration Type	States	Inputs	Outputs
System Included	$\begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix}$	$\boldsymbol{u}$	$\begin{bmatrix} \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$
System Included with Noise	$\begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{r}_y \end{bmatrix}$	$\begin{bmatrix} \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$
Standalone	$\hat{\boldsymbol{x}}$	$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix}$	$\hat{\boldsymbol{y}}$

The following sections discuss each of these configuration types in detail.

System Included Configuration

You can use the system included configuration to analyze and simulate the estimated states and the original states at the same time. For example, the following equation defines the output estimator error in a system included configuration.

$$\boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{C}(\boldsymbol{x} - \hat{\boldsymbol{x}})$$

By substituting the output estimator error in the general system configuration and removing the sensor noise $\boldsymbol{r}_y$, you obtain the following equations that describe the system included configuration.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{LC} & \boldsymbol{LC} \\ 0 & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{B} \end{bmatrix} \boldsymbol{u}$$

$$\begin{bmatrix} \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} & 0 \\ 0 & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{D} \end{bmatrix} \boldsymbol{u}$$

Figure 13-1 represents the dynamic system that these equations describe.

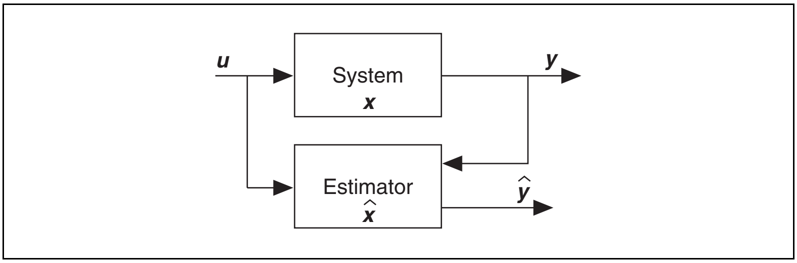

Figure 13-1. System Included State Estimator

The states, inputs, and outputs of the estimator are $\begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix}$, $\boldsymbol{u}$, and $\begin{bmatrix} \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$, respectively.

System Included Configuration with Noise

The system included configuration with noise incorporates noise $\boldsymbol{r}_y$ into the system included configuration. The following equation defines the output estimator error.

$$\boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{C}(\boldsymbol{x} - \hat{\boldsymbol{x}}) + \boldsymbol{r}_y$$

By substituting the output estimator error in the general system configuration, you obtain the following equations that describe the system included configuration with noise.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{LC} & \boldsymbol{LC} \\ 0 & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{x} \\ x \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} & \boldsymbol{L} \\ \boldsymbol{B} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{r}_y \end{bmatrix}$$

$$\begin{bmatrix} \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} & 0 \\ 0 & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{D} & 0 \\ \boldsymbol{D} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{r}_y \end{bmatrix}$$

Figure 13-2 represents the dynamic system that these equations describe.

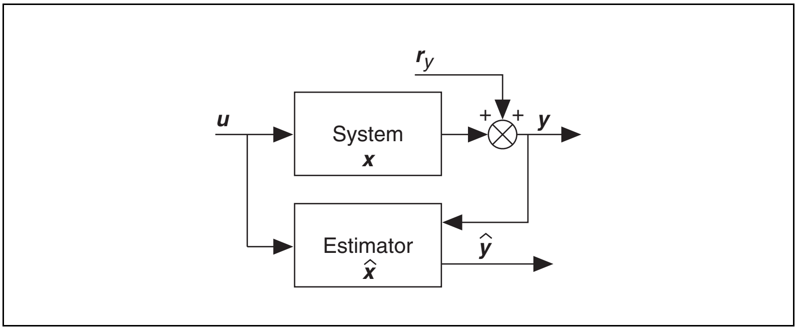

Figure 13-2. System Included State Estimator with Noise

The states, inputs, and outputs of the estimator are $\begin{bmatrix}\hat{x}\\x\end{bmatrix}$, $\begin{bmatrix}u\\r_y\end{bmatrix}$, and $\begin{bmatrix}\hat{y}\\y\end{bmatrix}$, respectively.

Standalone Configuration

In the standalone configuration, the system model detaches from the estimator. The system outputs $\boldsymbol{y}$ become inputs to the estimator. Unlike the system included and system included with noise configurations, the standalone configuration does not account for output noise $\boldsymbol{r}_y$.

The primary purpose of the standalone configuration is to implement the estimator on a real-time (RT) target. A secondary purpose of the standalone configuration is to perform offline simulation and analysis of the estimator. Offline simulation and analysis are useful for testing the estimator with mismatched models and systems. Mismatched models and systems have a calculated estimator gain that applies to a model with uncertainties.

The following equations describe the standalone configuration.

$$\hat{x} = (A - LC)\hat{x} + \begin{bmatrix}B - LD & L\end{bmatrix}\begin{bmatrix}u\\y\end{bmatrix}$$

$$\hat{y} = C\hat{x} + \begin{bmatrix}D & 0\end{bmatrix}\begin{bmatrix}u\\y\end{bmatrix}$$

This configuration does not include the original system. This configuration does not generate the system output internally but considers the output as another input to the estimator. Figure 13-3 represents the dynamic system that these equations describe.

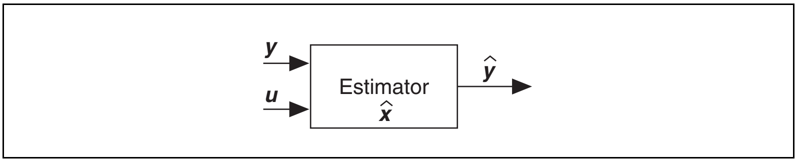

Figure 13-3. Standalone State Estimator

The states, inputs, and outputs of the estimator are $\hat{\boldsymbol{x}}$, $\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix}$, and $\hat{\boldsymbol{y}}$, respectively.

Example System Configurations

The following equations define an example second-order SISO state-space model with poles at –0.2 and –0.1.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -0.2 & 0.5 \\ 0 & -0.1 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u}$$

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \end{bmatrix} \boldsymbol{u}$$

You can implement a full state estimator for this system because this system is observable. To implement a state estimator for this system, you must calculate the estimator gain matrix $\boldsymbol{L}$ for the model of the system. Use the CD Ackermann VI to calculate $\boldsymbol{L}$ by placing the poles of the matrix $\boldsymbol{A} - \boldsymbol{LC}$ at [–1, –1]. This location is to the left of the original pole location in the complex plane. You can use this estimator gain matrix $\boldsymbol{L}$, along with the CD State Estimator VI, to study the performance of the estimator.

Note Use the CD Observability Matrix VI to verify that this system is observable. Use the CD Pole-Zero Map VI to determine the initial location of the system poles.

The following sections use this example system model to illustrate the different state estimator configurations. The examples in these sections use the CD Ackermann VI to calculate the estimator gain matrix $\boldsymbol{L}$. You also can calculate $\boldsymbol{L}$ using the CD Pole Placement VI or the CD Kalman Gain VI.

Example System Included State Estimator

Figure 13-4, shown below, uses the CD Ackermann VI to determine the estimator gain matrix ***L*** of the second-order SISO **State-Space Model**. You then use ***L*** with the CD State Estimator VI to create the state estimator, represented by the **Estimator Model**, for the system.

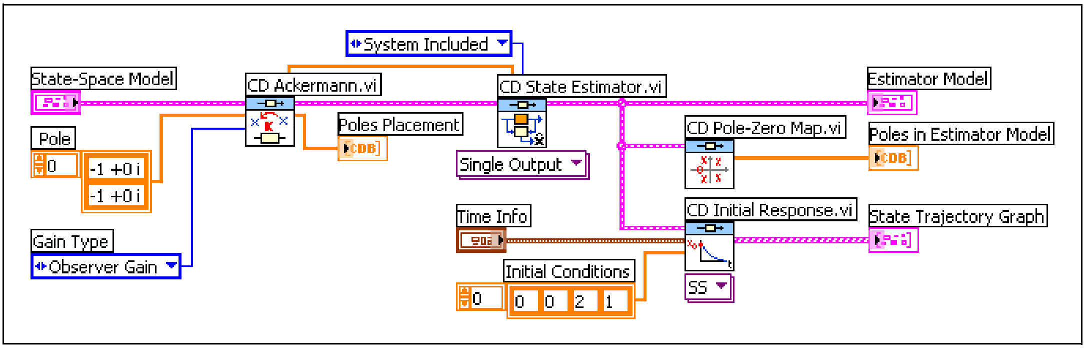

Figure 13-4. System Included State Estimator

Note You can study the performance of the state estimator with the CD Initial Response VI.

This configuration creates an **Estimator Model** that represents the original, or actual, states of the system and the estimated states in the same model. The **Estimator Model** consists of four states because this configuration appends the original second-order SISO state-space model to the state estimator, as shown in the following expression:

$$\begin{bmatrix}\dot{\hat{\boldsymbol{x}}}\\ \dot{\boldsymbol{x}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{A}-\boldsymbol{LC} & \boldsymbol{LC}\\ 0 & \boldsymbol{A}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{x}}\\ \boldsymbol{x}\end{bmatrix} + \begin{bmatrix}\boldsymbol{B}\\ \boldsymbol{B}\end{bmatrix}\boldsymbol{u}$$

$$\begin{bmatrix}\hat{\boldsymbol{y}}\\ \boldsymbol{y}\end{bmatrix} = \begin{bmatrix}\boldsymbol{C} & 0\\ 0 & \boldsymbol{C}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{x}}\\ \boldsymbol{x}\end{bmatrix}$$

Note The direct transmission matrix ***D*** is not part of the expression because it is null in this example.

The system included configuration monitors the response of the actual states of the system to a set of initial conditions. The CD Initial Response VI uses [0, 0, 2, 1] as the initial conditions. These initial conditions mean that the initial conditions of the actual states are [2, 1], whereas the initial conditions of the estimated states are [0, 0]. Therefore, the **Initial Conditions** vector of the **Estimator Model** is [0, 0, 2, 1].

The **State Trajectory Graph**, as shown in Figure 13-5, displays the response of the system and state estimator to the initial conditions [0, 0, 2, 1].

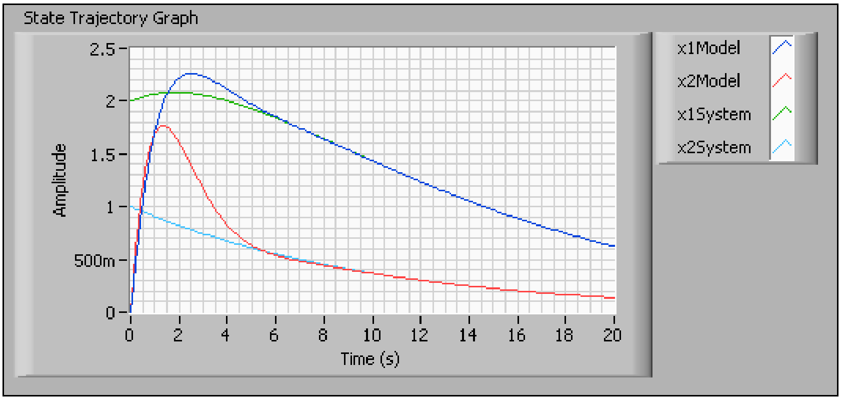

Figure 13-5. State Trajectory of System Included State Estimator

The initial conditions of the actual states are [2, 1]. The response of the actual states, therefore, starts at 2 and 1. The initial conditions of the estimated states are [0, 0]. The response of the estimated states, therefore, start at the origin. The estimated states promptly begin to track the actual states as the response of the actual system settles to steady state. This state estimator takes approximately six seconds to track the response of the system.

Example System Included State Estimator with Noise

In theory, you can place the poles of the state estimator as far left of the complex plane as necessary. This placement leads to very aggressive state estimators. Noise and system uncertainties, however, prevent you from configuring such aggressive estimators. To account for noise and system uncertainties, you can implement a state estimator using the system included with noise configuration. Consider the following system included with noise configuration.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{LC} & \boldsymbol{LC} \\ 0 & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{B} \end{bmatrix} \boldsymbol{u}$$

$$\begin{bmatrix} \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} & 0 \\ 0 & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{r}_y$$

The configuration of this system is essentially the same as the system in the *Example System Configurations* section of this chapter. The only addition is the measurement noise $\boldsymbol{r}_y$. Assume that the measurement noise in this example is a Gaussian noise in the system. The output noise influences the estimated model dynamics through the estimator gain matrix $\boldsymbol{L}$.
Figure 13-6 shows how to account for a Gaussian noise of 0.1 standard deviation in the **Estimator Model**.

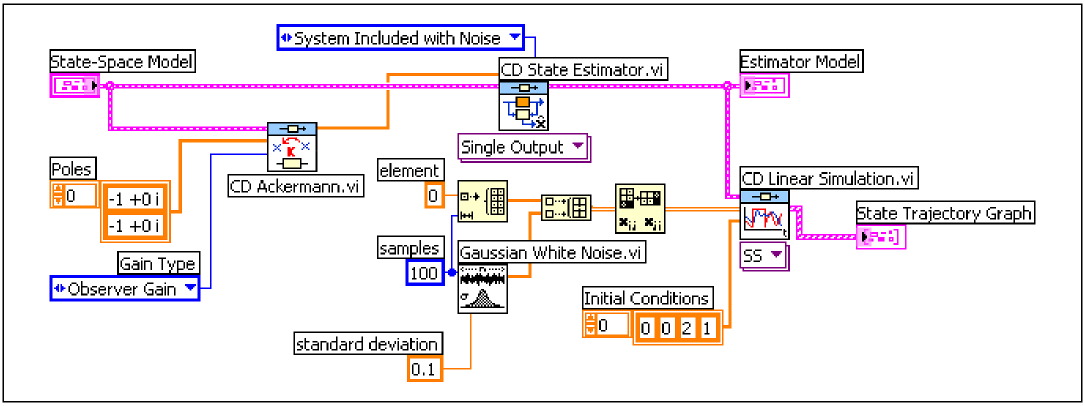

Figure 13-6. System Included State Estimator with Noise

The example in Figure 13-6 uses the state-space model and the CD Ackermann VI to determine the estimator gain matrix $\boldsymbol{L}$. The CD State Estimator VI then uses the system included with noise configuration to implement the state estimator, represented by the **Estimator Model**. Use

the Gaussian White Noise VI to view the effects of Gaussian noise on the system and the state estimator.

Note The CD Linear Simulation VI provides the response to a Gaussian noise with the same initial conditions as in Figure 13-4.

The **State Trajectory Graph**, as shown in Figure 13-7, displays the response of the system and state estimator to the same initial conditions [0, 0, 2, 1] used in the *Example System Included State Estimator* section of this chapter.

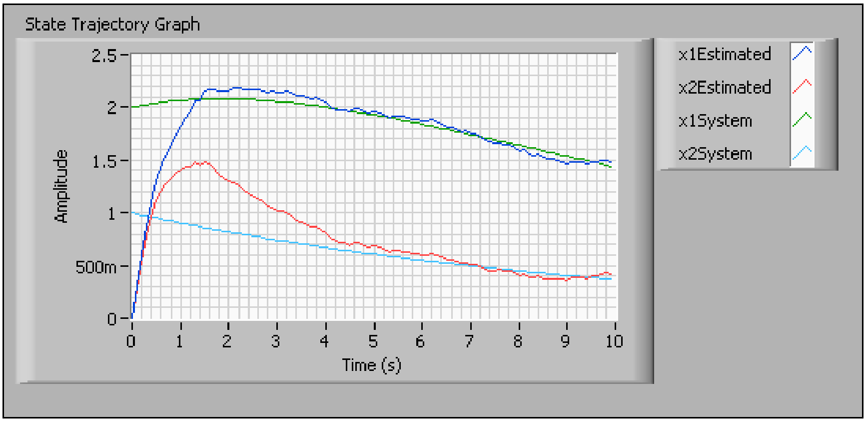

Figure 13-7. State Trajectory of System Included State Estimator with Noise

Similar to the graph in the *Example System Included State Estimator* section of this chapter, this **State Trajectory Graph** shows the response of the actual states starting at 2 and 1. The graph also shows the response of the estimated states starting at the origin. Notice the effect of the output noise $\boldsymbol{r}_y$ on the state estimation. Without noise, the state estimator took approximately six seconds to begin tracking the actual system. With noise, the state estimator takes much longer to track the actual system and the state estimator cannot track the actual system perfectly.

You can place the estimator poles closer to the origin to reduce the effect of the noise. However, when you move the estimator poles closer to the origin on the left side of the complex plane, you diminish the performance of the estimator in tracking the actual states.

One solution is to use the Kalman gain function to obtain an estimator gain matrix that effectively tracks the system states with an acceptable level of noise rejection. Refer to the *Kalman Gain Function* section of Chapter 12,

Designing State-Space Controllers, for information about using the Kalman gain function to find an optimal solution to this state estimator problem.

Example Standalone State Estimator

Most systems are complex and have many parameters and uncertainties. You often do not know all the parameters of a system when you create a model of that system, or you cannot create a model that encompasses all the uncertainties of the system. Thus, the actual system and the model of the system do not match.

When you build a state estimator based on a model that does not match the actual system, the result is a system-model mismatch. In this situation, you need to use the standalone configuration. This configuration detaches the system from the model so you can determine the effect of the system-model mismatch. Consider the following state-space model:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -0.1 & 0.5 \\ 0 & -0.1 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u}$$

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \end{bmatrix} \boldsymbol{u}$$

This model is similar to the model in the *Example System Configurations* section of this chapter. For this example, however, assume that the actual system contains uncertainties that cause this state-space model to be an inaccurate representation of the system. The difference is in the first entry of the system matrix $\boldsymbol{A}$, –0.1.

Figure 13-8 shows how the CD State Estimator VI uses the mismatched model, **State-Space Model**, to create the standalone estimator. This configuration connects the actual system, **System**, and the mismatched model, **State-Space Model**, in series so the actual system can provide the output $\boldsymbol{y}$ to the standalone state estimator.

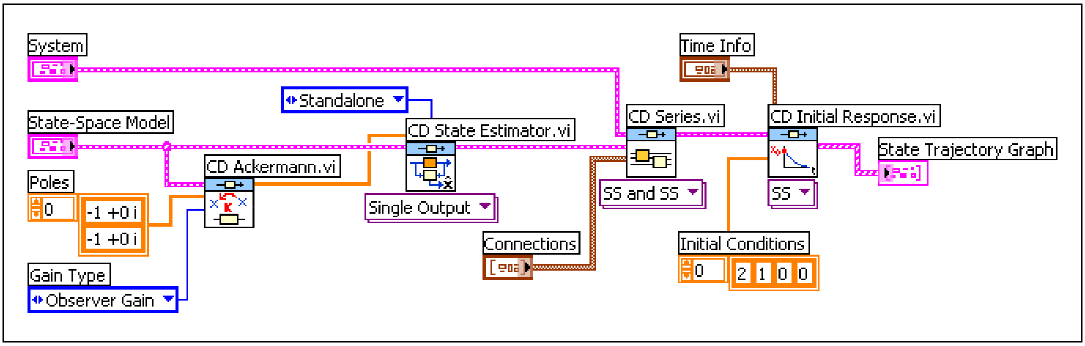

Figure 13-8. Standalone State Estimator

The example uses the CD Initial Response VI to evaluate the effectiveness of the state estimator. The **State Trajectory Graph** in Figure 13-9 shows the response of the actual and estimated states to the same set of initial conditions as in the *Example System Included State Estimator* section of this chapter.

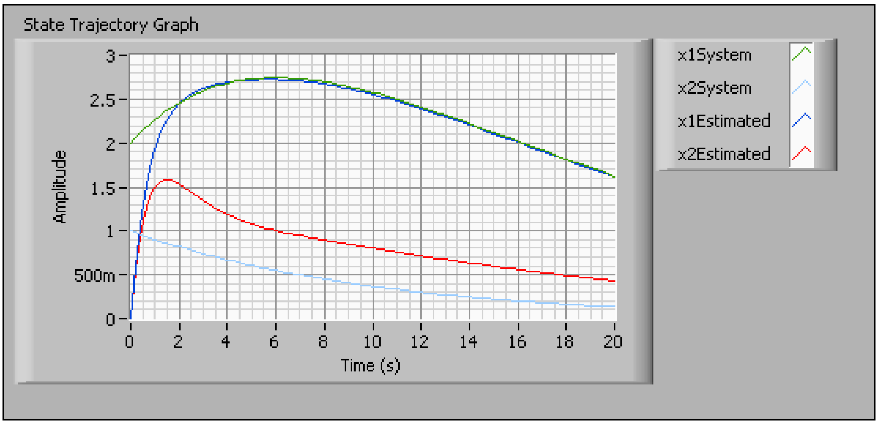

Figure 13-9. State Trajectory of Standalone State Estimator

Notice that a mismatch in the actual system and the model of the system greatly impacts the estimation of the second state. After 20 seconds, the state estimator still cannot track the actual state. Therefore, you must study the system and model mismatch to determine the effect of the mismatch on the state estimation.

14

Defining State-Space Controller Structures

State controllers use state information to calculate the control action. To define the structure of a state controller, you need a model of the system and a controller gain matrix $\boldsymbol{K}$. You can calculate $\boldsymbol{K}$ using the CD Pole Placement VI, the CD Ackermann VI, or the CD Linear Quadratic Regulator VI. Refer to Chapter 12, *Designing State-Space Controllers*, for information about these VIs.

You use $\boldsymbol{K}$ to define the structure of a controller. You can design a controller structure to take various factors, such as input noise or input disturbances, into consideration.

The following sections provide information about using the LabVIEW Control Design Toolkit to incorporate the gain matrix $\boldsymbol{K}$ into the control system. The controllers in the following sections assume that all inputs are known and all outputs are measurable. Refer to the *Measuring and Adjusting Inputs and Outputs* section of Chapter 13, *Defining State Estimator Structures*, for information about measuring inputs and outputs.

Configuring State Controllers

Use the CD State-Space Controller VI to define a controller structure. This VI integrates $\boldsymbol{K}$ into a dynamic system for analyzing and simulating the controller performance. Use the polymorphic VI selector to define one of the following three controller types:

- Compensator—Places a reference on the state. Defines the control action using $\boldsymbol{u} = \boldsymbol{K}(\boldsymbol{r}_x - \boldsymbol{x})$, where $\boldsymbol{r}_x$ is a state reference. If you estimate any states, $\boldsymbol{u} = \boldsymbol{K}(\boldsymbol{r}_x - \hat{\boldsymbol{x}})$ defines the state compensator control action.
- Regulator—Places a reference on the input. Defines the control action using $\boldsymbol{u} = \boldsymbol{r}_u - \boldsymbol{K}\boldsymbol{x}$, where $\boldsymbol{r}_u$ is an input reference. If you estimate any states, $\boldsymbol{u} = \boldsymbol{r}_u - \boldsymbol{K}\hat{\boldsymbol{x}}$ defines the state regulator control action.

- Regulator with Integral Action—Uses the following equation to define the control action.

$$\boldsymbol{u} = -\begin{bmatrix} \boldsymbol{K} & \boldsymbol{K_I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \int (\boldsymbol{y}_{ref} - \boldsymbol{y}) \end{bmatrix}$$

  In this equation, $\boldsymbol{y}_{ref}$ is the output reference, or setpoint.

The difference in these controllers is in how you calculate the control action $\boldsymbol{u}$.

You can implement any of these controller types using one of four different configurations. Use the **Configuration** parameter of the CD State-Space Controller VI to define a controller structure using one of the following four configurations:

- System Included—Appends the actual states of the system to the estimated states. This configuration is useful for analyzing and simulating the original and estimated states at the same time.
- System Included with Noise—Incorporates noise $\boldsymbol{r}_y$ into the system included configuration.
- Standalone with Estimator—Defines an estimator structure with the controller target. This configuration is useful for performing offline simulations and analyses of the controller. You can use offline simulations and analyses to test the controller with mismatched models and systems. Mismatched models and systems have a calculated estimator and controller gain that applies to the mismatched model, or the model with uncertainties. To select this configuration, choose a standalone configuration and then wire an estimator with output $\boldsymbol{L}$ to the **Estimator Gain (L)** input of the CD State-Space Controller VI.
- Standalone without Estimator—Bases the control action $\boldsymbol{u}$ on the actual states $\boldsymbol{x}$ instead of using an estimator to reconstruct the states. This configuration is useful for analyzing a closed-loop system. To select this configuration, choose a standalone configuration, but do not wire anything to the **Estimator Gain (L)** input of the CD State-Space Controller VI.

Note Both the system included and system included with noise configurations automatically include an estimator.

The following sections show the implementation of all four configurations for all three controller types.

State Compensator

A general system configuration appends the original model states $\boldsymbol{x}$ to the estimation model states $\hat{\boldsymbol{x}}$ to represent the compensator with an estimator. The following equations show this process:

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{BK} & 0 \\ -\boldsymbol{BK} & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{BK} & \boldsymbol{L} \\ \boldsymbol{BK} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_x \\ \boldsymbol{y} - \hat{\boldsymbol{y}} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K} & 0 \\ \boldsymbol{C} - \boldsymbol{DK} & 0 \\ -\boldsymbol{DK}^* & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{K} & 0 \\ \boldsymbol{DK} & 0 \\ \boldsymbol{DK} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_x \\ \boldsymbol{y} - \hat{\boldsymbol{y}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{r}_y \end{bmatrix}$$

Table 14-1 summarizes the different state compensator configurations and their corresponding states, inputs, and outputs.

Table 14-1. State Compensator Configurations

Configuration Type	States	Inputs	Outputs
System Included	$\begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix}$	$\boldsymbol{r}_x$	$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$
System Included with Noise	$\begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{r}_x \\ \boldsymbol{r}_y \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$
Standalone with Estimator	$\hat{\boldsymbol{x}}$	$\begin{bmatrix} \boldsymbol{r}_x \\ \boldsymbol{y} \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \end{bmatrix}$
Standalone without Estimator	$\boldsymbol{x}$	$\boldsymbol{r}_x$	$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix}$

The following sections show how to define each configuration of a state compensator.

System Included Configuration

In the system included configuration, the following equation defines the output error.

$$\boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{C}(\boldsymbol{x} - \hat{\boldsymbol{x}})$$

By substituting the output error in the general system configuration and removing the sensor noise $\boldsymbol{r}_y$ from the system, you obtain the following equations that describe the system included configuration.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}-\boldsymbol{BK}-\boldsymbol{LC} & \boldsymbol{LC} \\ -\boldsymbol{BK} & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{BK} \\ \boldsymbol{BK} \end{bmatrix} \boldsymbol{r}_x$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K} & 0 \\ \boldsymbol{C}-\boldsymbol{DK} & 0 \\ -\boldsymbol{DK} & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{DK} \\ \boldsymbol{DK} \end{bmatrix} \boldsymbol{r}_x$$

The reference vector $\boldsymbol{r}_x$ has as many elements as the number of states. Also, this configuration calculates the control action $\boldsymbol{u}$ internally and then gives $\boldsymbol{u}$ as an output of the state compensator.

Figure 14-1 represents the dynamic system that these equations describe.

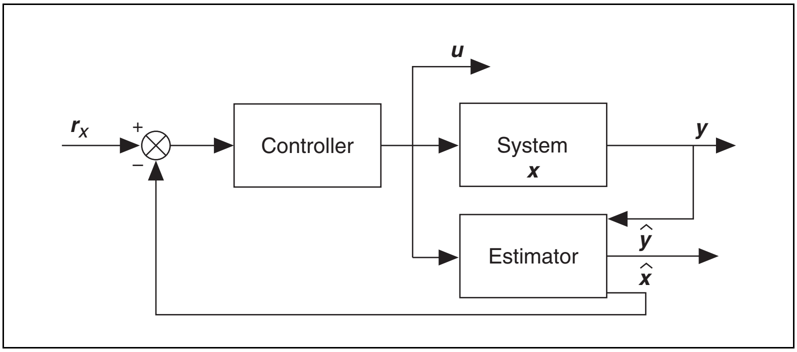

Figure 14-1. System Included State Compensator

The states, inputs, and outputs of the state compensator are $\begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix}$, $\boldsymbol{r}_x$, and $\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$, respectively.

System Included Configuration with Noise

The system included configuration with noise incorporates noise $\boldsymbol{r}_y$ into the system included configuration. The following equation defines the output error.

$$\boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{C}(\boldsymbol{x} - \hat{\boldsymbol{x}}) + \boldsymbol{r}_y$$

By substituting the output error in the general system configuration, you obtain the following equations that describe the system included configuration with noise.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{BK} - \boldsymbol{LC} & \boldsymbol{LC} \\ -\boldsymbol{BK} & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{BK} & \boldsymbol{L} \\ \boldsymbol{BK} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_x \\ \boldsymbol{r}_y \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K} & 0 \\ \boldsymbol{C} - \boldsymbol{DK} & 0 \\ -\boldsymbol{DK} & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{K} & 0 \\ \boldsymbol{DK} & 0 \\ \boldsymbol{DK} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_x \\ \boldsymbol{r}_y \end{bmatrix}$$

The reference vector $\boldsymbol{r}_x$ has as many elements as the number of states. Also, this configuration calculates the control action $\boldsymbol{u}$ internally and then gives $\boldsymbol{u}$ as an output of the compensator.

Figure 14-2 represents the dynamic system that these equations describe.

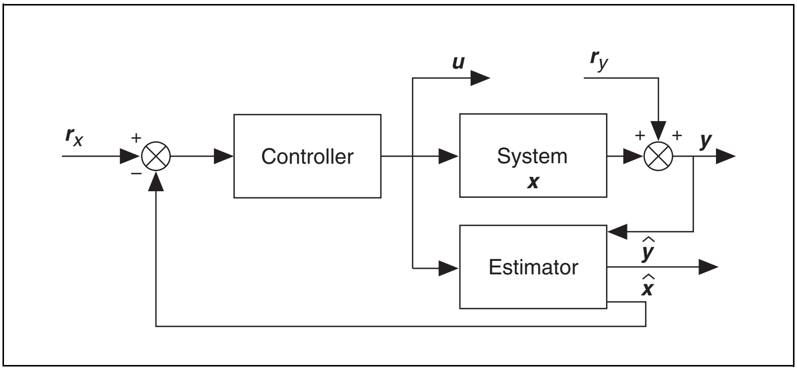

Figure 14-2. System Included State Compensator with Noise

The states, inputs, and outputs of the state compensator are $\begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix}$, $\begin{bmatrix} \boldsymbol{r}_x \\ \boldsymbol{r}_y \end{bmatrix}$, and $\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$, respectively.

Standalone Configuration with Estimator

In the standalone configuration with estimator, the system model detaches from the controller. The system outputs ***y*** become inputs to the estimator. Unlike the system included and system included with noise configurations, the standalone configuration with estimator does not account for output error. You must wire a value to the **Estimator Gain (L)** input of the CD State-Space Controller VI to include the estimator in the standalone state compensator.

The following equations describe the standalone configuration.

$$\dot{\hat{x}} = [A - BK - L(C - DK)]\hat{x} + [BK - LDK \ \ L]\begin{bmatrix} r_x \\ y \end{bmatrix}$$

$$\begin{bmatrix} u \\ \hat{y} \end{bmatrix} = \begin{bmatrix} -K \\ C - DK \end{bmatrix}\hat{x} + \begin{bmatrix} K & 0 \\ DK & 0 \end{bmatrix}\begin{bmatrix} r_x \\ y \end{bmatrix}$$

This configuration does not include the original system. This configuration considers the system output ***y*** as another input to the estimator.

Figure 14-3 represents the dynamic system that these equations describe.

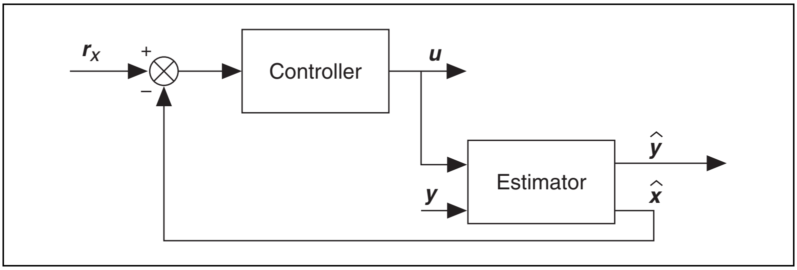

Figure 14-3. Standalone State Compensator with Estimator

The states, inputs, and outputs of state compensator are $\hat{x}$, $\begin{bmatrix} r_x \\ y \end{bmatrix}$, and $\begin{bmatrix} u \\ \hat{y} \end{bmatrix}$, respectively.

Standalone Configuration without Estimator

In the standalone configuration without estimator, you calculate the control action ***u*** using the states. As such, you do not need an estimator. In the CD State-Space Controller VI, do not wire a value to the **Estimator Gain (L)** input to exclude the estimator in the standalone state compensator.

The following equations describe the standalone configuration.

$$\dot{x} = (A - BK)x + BKr_x$$

$$\begin{bmatrix} u \\ y \end{bmatrix} = \begin{bmatrix} -K \\ C - DK \end{bmatrix} x + \begin{bmatrix} K \\ DK \end{bmatrix} r_x$$

The states and outputs of the standalone compensator without estimator correspond to the states and outputs of the actual system.

Figure 14-4 represents the dynamic system that these equations describe.

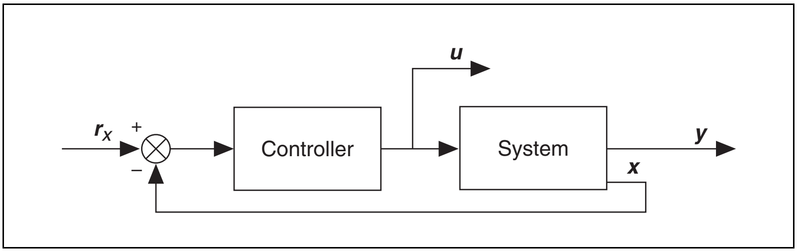

Figure 14-4. Standalone State Compensator without Estimator

The states, inputs, and outputs of the state compensator are x, r_x, and $\begin{bmatrix} u \\ y \end{bmatrix}$, respectively.

State Regulator

A general system configuration appends the original model states x to the estimation model states $\hat{x}$ to represent the state regulator with an estimator. The following equations show this process:

$$\begin{bmatrix} \dot{\hat{x}} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} A - BK & 0 \\ -BK & A \end{bmatrix} \begin{bmatrix} \hat{x} \\ x \end{bmatrix} + \begin{bmatrix} B & L \\ B & 0 \end{bmatrix} \begin{bmatrix} r_u \\ y - \hat{y} \end{bmatrix}$$

$$\begin{bmatrix} u \\ \hat{y} \\ y \end{bmatrix} = \begin{bmatrix} -K & 0 \\ C - DK & 0 \\ -DK & C \end{bmatrix} \begin{bmatrix} \hat{x} \\ x \end{bmatrix} + \begin{bmatrix} I & 0 \\ D & 0 \\ D & 0 \end{bmatrix} \begin{bmatrix} r_u \\ y - \hat{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ r_y \end{bmatrix}$$

Table 14-2 summarizes the different state regulator configurations and their corresponding states, inputs, and outputs.

Table 14-2. State Regulator Configuration Types

Configuration Type	States	Inputs	Outputs
System Included	$\begin{bmatrix}\hat{\boldsymbol{x}}\\ \boldsymbol{x}\end{bmatrix}$	$\boldsymbol{r}_u$	$\begin{bmatrix}\boldsymbol{u}\\ \hat{\boldsymbol{y}}\\ \boldsymbol{y}\end{bmatrix}$
System Included with Noise	$\begin{bmatrix}\hat{\boldsymbol{x}}\\ \boldsymbol{x}\end{bmatrix}$	$\begin{bmatrix}\boldsymbol{r}_u\\ \boldsymbol{r}_y\end{bmatrix}$	$\begin{bmatrix}\boldsymbol{u}\\ \hat{\boldsymbol{y}}\\ \boldsymbol{y}\end{bmatrix}$
Standalone with Estimator	$\hat{\boldsymbol{x}}$	$\begin{bmatrix}\boldsymbol{r}_u\\ \boldsymbol{y}\end{bmatrix}$	$\begin{bmatrix}\boldsymbol{u}\\ \hat{\boldsymbol{y}}\end{bmatrix}$
Standalone without Estimator	$\boldsymbol{x}$	$[\boldsymbol{r}_u]$	$\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{y}\end{bmatrix}$

The following sections show how to define each configuration.

System Included Configuration

In the system included configuration, the following equation defines the output error.

$$\boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{C}(\boldsymbol{x} - \hat{\boldsymbol{x}})$$

By substituting the output error in the general system configuration and removing the sensor noise $\boldsymbol{r}_y$ from the system, you obtain the following equations that describe the system included configuration.

$$\begin{bmatrix}\dot{\hat{\boldsymbol{x}}}\\ \dot{\boldsymbol{x}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{A}-\boldsymbol{BK}-\boldsymbol{LC} & \boldsymbol{LC}\\ -\boldsymbol{BK} & \boldsymbol{A}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{x}}\\ \boldsymbol{x}\end{bmatrix} + \begin{bmatrix}\boldsymbol{B}\\ \boldsymbol{B}\end{bmatrix}\boldsymbol{r}_u$$

$$\begin{bmatrix}\boldsymbol{u}\\ \hat{\boldsymbol{y}}\\ \boldsymbol{y}\end{bmatrix} = \begin{bmatrix}-\boldsymbol{K} & 0\\ \boldsymbol{C}-\boldsymbol{DK} & 0\\ -\boldsymbol{DK} & \boldsymbol{C}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{x}}\\ \boldsymbol{x}\end{bmatrix} + \begin{bmatrix}\boldsymbol{I}\\ \boldsymbol{D}\\ \boldsymbol{D}\end{bmatrix}\boldsymbol{r}_u$$

The reference vector, or actuator noise, $\boldsymbol{r}_u$ has as many elements as the number of inputs. Also, this configuration calculates the control action $\boldsymbol{u}$ internally and then gives $\boldsymbol{u}$ as an output of the state regulator.

Figure 14-5 represents the dynamic system that these equations describe.

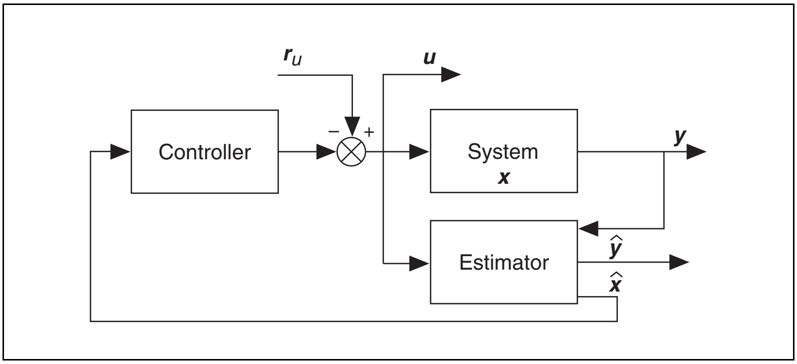

Figure 14-5. System Included State Regulator

The states, inputs, and outputs of the state regulator are $\begin{bmatrix} \hat{x} \\ x \end{bmatrix}$, r_u, and $\begin{bmatrix} u \\ \hat{y} \\ y \end{bmatrix}$, respectively.

System Included Configuration with Noise

The system included configuration with noise incorporates noise $\boldsymbol{r}_y$ into the system included configuration. The following equation defines the output error.

$$\boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{C}(\boldsymbol{x} - \hat{\boldsymbol{x}}) + \boldsymbol{r}_y$$

By substituting the output error in the general system configuration, you obtain the following equations that describe the system included configuration with noise.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{BK} - \boldsymbol{LC} & \boldsymbol{LC} \\ -\boldsymbol{BK} & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} & \boldsymbol{L} \\ \boldsymbol{B} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_u \\ \boldsymbol{r}_y \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K} & 0 \\ \boldsymbol{C} - \boldsymbol{DK} & 0 \\ -\boldsymbol{DK} & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{I} & 0 \\ \boldsymbol{D} & 0 \\ \boldsymbol{D} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_u \\ \boldsymbol{r}_y \end{bmatrix}$$

The reference vector, or actuator noise, $\boldsymbol{r}_u$ has as many elements as the number of inputs. Also, this configuration calculates the control action $\boldsymbol{u}$ internally and then gives $\boldsymbol{u}$ as an output of the state regulator.

Figure 14-6 represents the dynamic system that these equations describe.

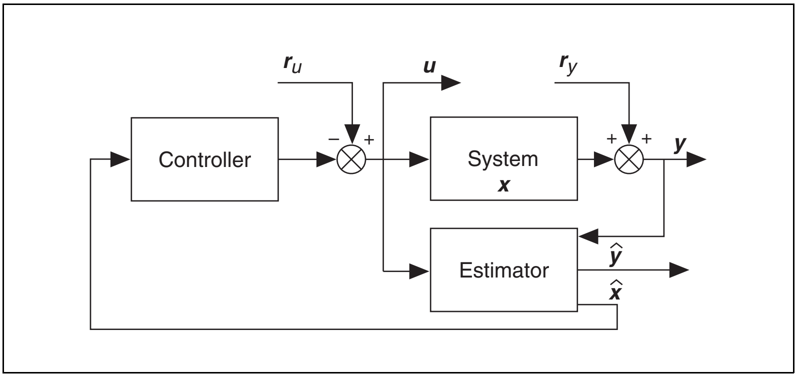

Figure 14-6. System Included State Regulator with Noise

The states, inputs, and outputs of the state regulator are $\begin{bmatrix} \hat{x} \\ x \end{bmatrix}$, $\begin{bmatrix} r_u \\ r_y \end{bmatrix}$, and $\begin{bmatrix} u \\ \hat{y} \\ y \end{bmatrix}$, respectively.

Standalone Configuration with Estimator

In the standalone configuration with estimator, the system model detaches from the controller. The system outputs $\boldsymbol{y}$ become inputs to the estimator. Unlike the system included and system included with noise configurations, the standalone configuration with estimator does not account for output error. You must wire a value to the **Estimator Gain (L)** input of the CD State-Space Controller VI to include the estimator in the standalone state compensator.

The following equations describe the standalone configuration with estimator.

$$\hat{\boldsymbol{x}} = [\boldsymbol{A} - \boldsymbol{BK} - \hat{\boldsymbol{L}}(\boldsymbol{C} - \boldsymbol{DK})]\hat{\boldsymbol{x}} + \begin{bmatrix} \boldsymbol{B} - \boldsymbol{LD} & \boldsymbol{L} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_u \\ \boldsymbol{y} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K} \\ \boldsymbol{C} - \boldsymbol{DK} \end{bmatrix} \hat{\boldsymbol{x}} + \begin{bmatrix} \boldsymbol{I} & 0 \\ \boldsymbol{D} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_u \\ \boldsymbol{y} \end{bmatrix}$$

This configuration does not include the original system. This configuration considers the system output $\boldsymbol{y}$ as another input to the estimator.

Figure 14-7 represents the dynamic system that these equations describe.

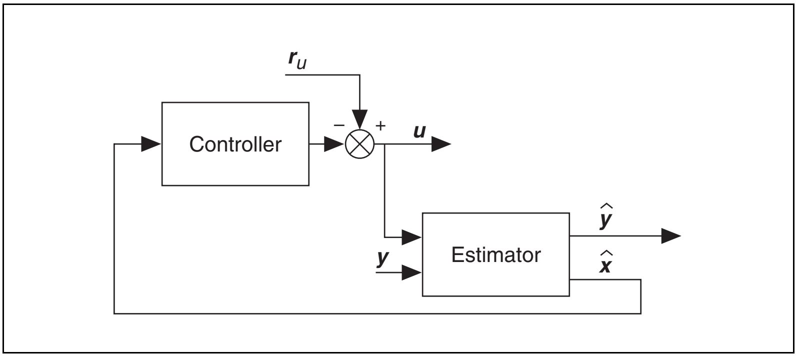

Figure 14-7. Standalone State Regulator with Estimator

The states, inputs, and outputs of the state regulator are $\hat{x}$, $\begin{bmatrix} r_u \\ y \end{bmatrix}$, and $\begin{bmatrix} u \\ \hat{y} \end{bmatrix}$, respectively.

Standalone Configuration without Estimator

The standalone configuration without estimator uses states to calculate of the control action u. As such, you do not need an estimator. In the CD State-Space Controller VI, do not wire a value to the **Estimator Gain (L)** input to exclude the estimator in the standalone state regulator.

The following equations describe the standalone configuration.

$$\hat{x} = (A - BK)x + Br_u$$
$$y = (C - DK)x + Dr_u$$

The states and outputs of the standalone state regulator without estimator correspond to the states and outputs of the actual system.

Figure 14-8 represents the dynamic system that these equations describe.

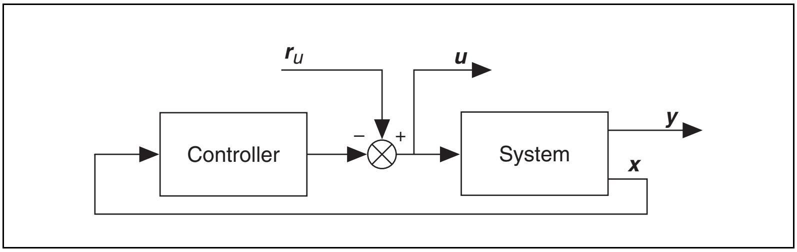

Figure 14-8. Standalone State Regulator without Estimator

The states, inputs, and outputs of the state regulator are $\boldsymbol{x}$, $\boldsymbol{r}_u$, and $\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix}$, respectively.

State Regulator with Integral Action

A general system configuration appends the output error integrator z to the estimation model states $\hat{\boldsymbol{x}}$. A general system configuration also augments the resulting vector $(\hat{\boldsymbol{x}}, z)$ with the original model states $\boldsymbol{x}$ to represent the state regulator with integral action and an estimator. The following equations show this process:

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{z} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & 0 & 0 \\ 0 & \Gamma & 0 \\ 0 & 0 & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} & 0 & \boldsymbol{L} \\ 0 & \boldsymbol{I} & 0 \\ \boldsymbol{B} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y}_{ref} - \boldsymbol{y} \\ \boldsymbol{y} - \hat{\boldsymbol{y}} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K_x} & -\boldsymbol{K_i} & 0 \\ \boldsymbol{C} & 0 & 0 \\ 0 & 0 & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \boldsymbol{D} & 0 & 0 \\ \boldsymbol{D} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y}_{ref} - \boldsymbol{y} \\ \boldsymbol{y} - \hat{\boldsymbol{y}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{r}_y \end{bmatrix}$$

In these equations, $\boldsymbol{K}_x$ is the gain, $\boldsymbol{K}_i$ is the integral action, $\boldsymbol{y}_{ref}$ is the reference variable that you are tracking, and $\boldsymbol{y}$ is the output variable that you use to track $\boldsymbol{y}_{ref}$. In these equations, Γ varies depending on whether the model describes a continuous or discrete system. If the system is continuous, $\Gamma = 0$. If the system is discrete, $\Gamma = \boldsymbol{I}$.

When you define the control action for a state regulator with integral action using the output error integrator z, you obtain the following control action equation.

$$\boldsymbol{u} = -\begin{bmatrix} \boldsymbol{K} & \boldsymbol{K_I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ z \end{bmatrix}$$

Substituting the control action into state dynamics of the general system configuration defined in the previous equation, you obtain the following equation that also defines the general system configuration.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{z} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{BK_x} & 0 - \boldsymbol{BK_i} & 0 \\ 0 & 0 & 0 \\ -\boldsymbol{BK_x} & -\boldsymbol{BK_i} & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} 0 & \boldsymbol{L} \\ \boldsymbol{I} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} - \boldsymbol{y} \\ \boldsymbol{y} - \hat{\boldsymbol{y}} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K_x} & -\boldsymbol{K_i} & 0 \\ \boldsymbol{C} - \boldsymbol{DK_x} & -\boldsymbol{DK_i} & 0 \\ -\boldsymbol{DK_x} & -\boldsymbol{DK_i} & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} - \boldsymbol{y} \\ \boldsymbol{y} - \hat{\boldsymbol{y}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{r}_y \end{bmatrix}$$

Table 14-3 summarizes the different state regulator with integral action configurations and their corresponding states, inputs, and outputs.

Table 14-3. State Regulator with Integral Action Configuration Types

Configuration Type	States	Inputs	Outputs
System Included	$\begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix}$	$\boldsymbol{y}_{ref}$	$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$
System Included with Noise	$\begin{bmatrix} \hat{\boldsymbol{x}} \\ z \\ \boldsymbol{x} \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{y}_{ref} \\ \boldsymbol{r}_y \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$
Standalone with Estimator	$\hat{\boldsymbol{x}}$	$\begin{bmatrix} \boldsymbol{y}_{ref} \\ \boldsymbol{y} \end{bmatrix}$	$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \end{bmatrix}$
Standalone without Estimator	$\boldsymbol{x}$	$\boldsymbol{y}_{ref}$	$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix}$

The following sections show how to derive each configuration.

System Included Configuration

In the system included configuration, the following equations define the output error and system output.

$$y - \hat{y} = C(x - \hat{x})$$
$$y = -D(K_x\hat{x} + K_i z) + Cx$$

By substituting the output error and system output in the general system configuration and removing the sensor noise r_y from the system, you obtain the following equations that describe the system included configuration.

$$\begin{bmatrix} \dot{\hat{x}} \\ \dot{z} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} A - BK_x - LC & -BK_i & LC \\ DK_x & \Gamma + DK_i & -C \\ -BK_x & -BK_i & A \end{bmatrix} \begin{bmatrix} \hat{x} \\ z \\ x \end{bmatrix} + \begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix} y_{ref}$$

$$\begin{bmatrix} u \\ \hat{y} \\ y \end{bmatrix} = \begin{bmatrix} -K_x & -K_i & 0 \\ C - DK_x & -DK_i & 0 \\ -DK_x & -DK_i & C \end{bmatrix} \begin{bmatrix} \hat{x} \\ z \\ x \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} y_{ref}$$

The reference vector y_{ref} has as many elements as the number of outputs. Also, this configuration calculates the control action u internally and then gives u as an output of the state regulator with integral action.

Figure 14-9 represents the dynamic system that these equations describe.

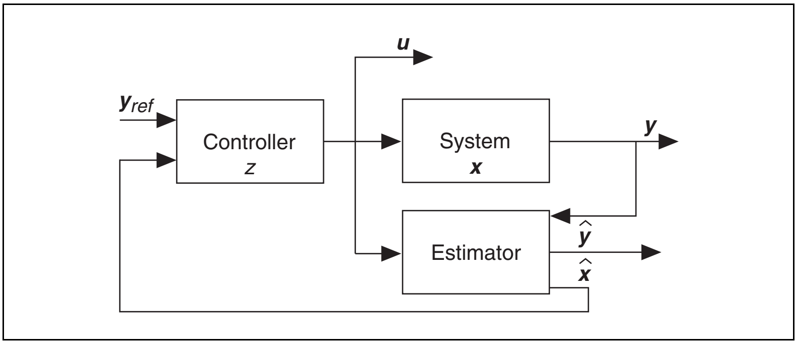

Figure 14-9. System Included Regulator with Integral Action

The states, inputs, and outputs of the state regulator with integral action are $\begin{bmatrix} \hat{\boldsymbol{x}} \\ z \\ \boldsymbol{x} \end{bmatrix}$, $\boldsymbol{y}_{ref}$, and $\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$, respectively.

System Included Configuration with Noise

The system included configuration with noise incorporates noise $\boldsymbol{r}_y$ into the system included configuration. The following equations define the output error and system output.

$$\boldsymbol{y} - \hat{\boldsymbol{y}} = -\boldsymbol{C}\hat{\boldsymbol{x}} + \boldsymbol{C}\boldsymbol{x} + \boldsymbol{r}_y$$

$$\boldsymbol{y} = -\boldsymbol{D}(\boldsymbol{K}_x\hat{\boldsymbol{x}} + \boldsymbol{K}_i z) + \boldsymbol{C}z + \boldsymbol{r}_y$$

By substituting the output error and system output in the general system configuration, you obtain the following equations that describe the system included configuration with noise.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{z} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}_x - \boldsymbol{L}\boldsymbol{C} & -\boldsymbol{B}\boldsymbol{K}_i & \boldsymbol{L}\boldsymbol{C} \\ \boldsymbol{D}\boldsymbol{K}_x & \Gamma + \boldsymbol{D}\boldsymbol{K}_i & -\boldsymbol{C} \\ -\boldsymbol{B}\boldsymbol{K}_x & -\boldsymbol{B}\boldsymbol{K}_i & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} 0 & \boldsymbol{L} \\ \boldsymbol{I} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} \\ \boldsymbol{r}_y \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K}_x & -\boldsymbol{K}_i & 0 \\ \boldsymbol{C} - \boldsymbol{D}\boldsymbol{K}_x & -\boldsymbol{D}\boldsymbol{K}_i & 0 \\ -\boldsymbol{D}\boldsymbol{K}_x & -\boldsymbol{D}\boldsymbol{K}_i & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \\ \boldsymbol{x} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} \\ \boldsymbol{r}_y \end{bmatrix}$$

The reference vector $\boldsymbol{y}_{ref}$ has as many elements as the number of outputs. Also, this configuration calculates the control action $\boldsymbol{u}$ internally and then gives $\boldsymbol{u}$ as an output of the state regulator with integral action.

Figure 14-10 represents the dynamic system described by these equations.

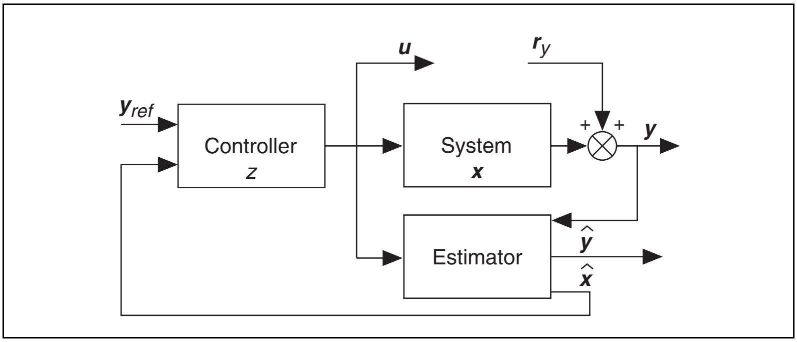

Figure 14-10. System Included State Regulator with Integral Action, with Noise

The states, inputs, and outputs of the state regulator with integral action are $\begin{bmatrix} \hat{\boldsymbol{x}} \\ z \\ \boldsymbol{x} \end{bmatrix}$, $\begin{bmatrix} \boldsymbol{y}_{ref} \\ \boldsymbol{r}_y \end{bmatrix}$, and $\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \\ \boldsymbol{y} \end{bmatrix}$, respectively.

Standalone Configuration with Estimator

In the standalone configuration with estimator, the system model detaches from the controller. The system outputs ***y*** become inputs to the estimator. Unlike the system included and system included with noise configurations, the standalone configuration with estimator does not account for output error. You must wire a value to the **Estimator Gain (L)** input of the CD State-Space Controller VI to include the estimator in the standalone state regulator with integral action.

The following equations describe the standalone configuration.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - (\boldsymbol{B} - \boldsymbol{LD})\boldsymbol{K}_x - \boldsymbol{LC} & (\boldsymbol{D} - \boldsymbol{B})\boldsymbol{K}_i \\ 0 & \Gamma \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \end{bmatrix} + \begin{bmatrix} 0 & \boldsymbol{L} \\ \boldsymbol{I} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} - \boldsymbol{y} \\ \boldsymbol{y} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K}_x & -\boldsymbol{K}_i \\ \boldsymbol{C} - \boldsymbol{DK}_x & -\boldsymbol{DK}_i \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} - \boldsymbol{y} \\ \boldsymbol{y} \end{bmatrix}$$

Use the following substitution to make the input independent.

$$\begin{bmatrix} 0 & \boldsymbol{L} \\ \boldsymbol{I} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} - \boldsymbol{y} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Ly} \\ \boldsymbol{y}_{ref} - \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{L} \\ \boldsymbol{I} & -\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} \\ \boldsymbol{y} \end{bmatrix}$$

This process results in the following equations that describe the standalone configuration.

$$\begin{bmatrix} \dot{\hat{\boldsymbol{x}}} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} - (\boldsymbol{B} - \boldsymbol{LD}) & \boldsymbol{K}_x - \boldsymbol{LC}(\boldsymbol{D} - \boldsymbol{B})\boldsymbol{K}_i \\ 0 & \Gamma \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \end{bmatrix} + \begin{bmatrix} 0 & \boldsymbol{L} \\ \boldsymbol{I} & -\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} \\ \boldsymbol{y} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{K}_x & -\boldsymbol{K}_i \\ \boldsymbol{C} - \boldsymbol{DK}_x & -\boldsymbol{DK}_i \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ z \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{ref} \\ \boldsymbol{y} \end{bmatrix}$$

This configuration does not include the original system. This configuration considers the system output $\boldsymbol{y}$ as another input to the estimator.

Figure 14-11 represents the dynamic system that these equations describe.

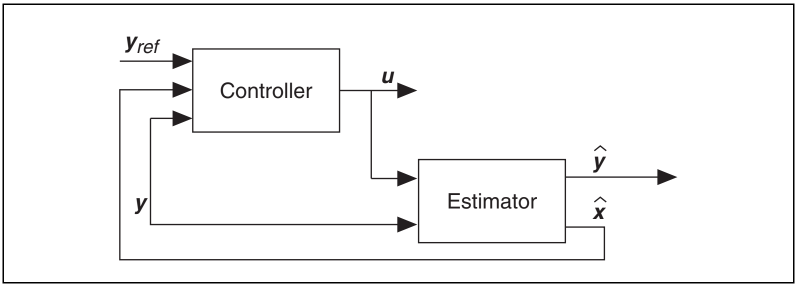

Figure 14-11. Standalone State Regulator with Integral Action, with Estimator

The states, inputs, and outputs of the state regulator with integral action are $\hat{\boldsymbol{x}}$, $\begin{bmatrix} \boldsymbol{y}_{ref} \\ \boldsymbol{y} \end{bmatrix}$, and $\begin{bmatrix} \boldsymbol{u} \\ \hat{\boldsymbol{y}} \end{bmatrix}$, respectively.

Standalone Configuration without Estimator

The standalone configuration without estimator uses states to calculate of the control action $\boldsymbol{u}$. As such, you do not need an estimator. In the CD State-Space Controller VI, do not wire a value to the **Estimator Gain (L)** input to exclude the estimator in the standalone state regulator with integral action.

The following equations describe the standalone configuration.

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} A - BK_x & -BK_i \\ 0 & \Gamma \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} (y_{ref} - y)$$

$$\begin{bmatrix} u \\ \hat{y} \end{bmatrix} = \begin{bmatrix} -K_x & -K_i \\ C - DK_x & -DK_i \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} (y_{ref} - y)$$

Use the following substitution to make the inputs independent.

$$\begin{bmatrix} 0 \\ I \end{bmatrix} (y_{ref} - y) = \begin{bmatrix} 0 \\ y_{ref} - y \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ I & -I \end{bmatrix} \begin{bmatrix} y_{ref} \\ y \end{bmatrix}$$

This process results in the following equations that describe the standalone configuration without estimator.

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} A - BK_x & -BK_i \\ 0 & \Gamma \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ I & -I \end{bmatrix} \begin{bmatrix} y_{ref} \\ y \end{bmatrix}$$

$$\begin{bmatrix} u \\ y \end{bmatrix} = \begin{bmatrix} -K_x & -K_i \\ C - DK_x & -DK_i \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_{ref} \\ y \end{bmatrix}$$

Using this configuration, the states and outputs of the standalone state regulator with integral action correspond to the states and outputs of the actual system.

Figure 14-12 represents the system that these equations describe.

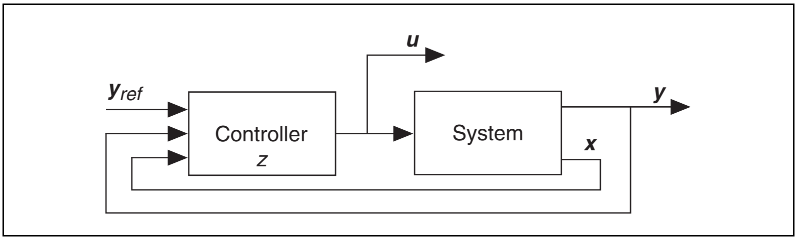

Figure 14-12. Standalone State Regulator with Integral Action, without Estimator

The states, inputs, and outputs of the state regulator with integral action are $\begin{bmatrix} x \\ z \end{bmatrix}$, $\begin{bmatrix} y_{ref} \\ y \end{bmatrix}$, and $\begin{bmatrix} u \\ y \end{bmatrix}$, respectively.

Example System Configurations

The following equations define an example second-order SISO state-space model with poles at –0.2 and –0.1.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -0.2 & 0.5 \\ 0.1 & -0.1 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u}$$

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \end{bmatrix} \boldsymbol{u}$$

You can implement a full state controller for this system because this system is controllable. To implement a state controller for this system, you must calculate the controller gain matrix ***K*** for the model of the system. Use the CD Ackermann VI to calculate ***K*** by placing the poles of the matrix ***A*** – ***BK*** at [–1, –1]. This location is to the left of the original pole location in the complex plane. You can use this controller gain matrix ***K***, along with the CD State-Space Controller VI, to study the performance of the compensator.

Note Use the CD Controllability Matrix VI to verify that this system is observable. Use the CD Pole-Zero Map VI to determine the initial location of the system poles.

Figure 14-13 shows the response of the example system to initial conditions of [2, 1]. This system is unstable because the response increases exponentially and does not settle at a steady-state value.

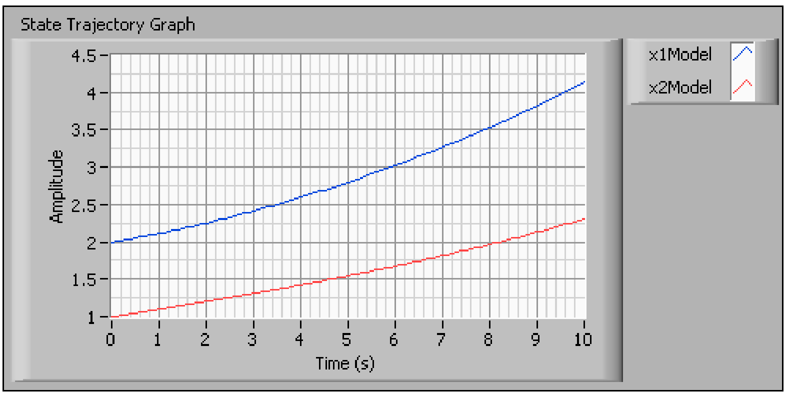

Figure 14-13. Unstable Open-Loop System

Even though this system is unstable, the system is still controllable. Because the system is controllable, you can use a state compensator to place the closed-loop poles in the left-hand side of the complex plane to make the response stable. You can calculate the controller gain matrix $\boldsymbol{K}$ by using the CD Ackermann VI to place the poles of the matrix $\boldsymbol{A} - \boldsymbol{BK}$ at [–1, –1]. You can use $\boldsymbol{K}$ to study the performance of the compensator by selecting the **Compensator** instance of the CD State-Space Controller VI.

The following sections use this example system model to illustrate the different state controller configurations. These examples are state compensators. You can define a state regulator or state regulator with integral action by selecting the **Regulator** or **Regulator with Integral Action** instance of the CD State-Space Controller VI, respectively.

The examples in these sections use the CD Ackermann VI to calculate the controller gain matrix $\boldsymbol{K}$. You also can calculate $\boldsymbol{K}$ using the CD Pole Placement VI or the CD Linear Quadratic Regulator VI.

Example System Included State Compensator

In theory, you cannot always directly measure the system states for control purposes. Therefore, you must synthesize a controller using the system outputs. To calculate the control action based on the estimated states, the estimator needs to approach the actual states faster than the controller. Therefore, you can calculate an estimator gain matrix such that $\boldsymbol{A} - \boldsymbol{LC}$ has eigenvalues at [–5, –5], which is farther to the left of the origin than the poles of the controller located at [–1, –1].

The system included configuration takes both the estimator gain matrix $\boldsymbol{L}$ and the controller gain matrix $\boldsymbol{K}$ and uses them to synthesize a state compensator. Figure 14-14 shows the implementation of a state compensator using the system included configuration.

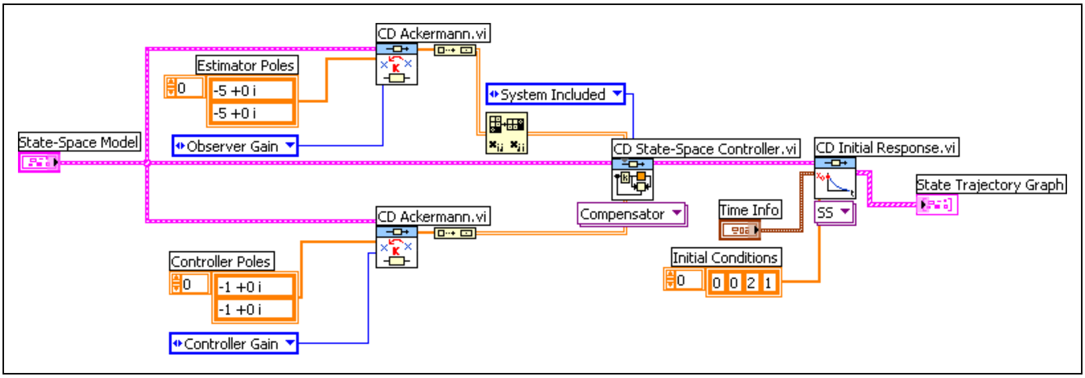

Figure 14-14. System Included State Compensator

The CD Initial Response VI uses [0, 0, 2, 1] as the initial conditions. As in the *Example System Included State Estimator* section of Chapter 13, *Defining State Estimator Structures*, these initial conditions mean that the initial conditions of the actual states are [2, 1], whereas the initial conditions of the estimated states are [0, 0]. Figure 14-15 shows the response of the system to those initial conditions.

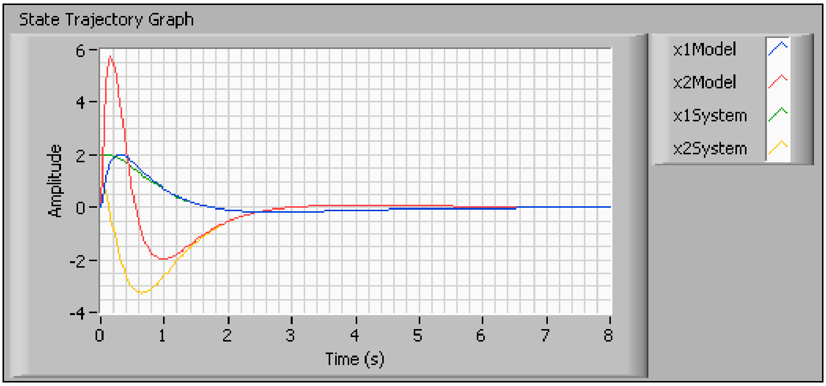

Figure 14-15. State Trajectory of a System Included State Compensator

Notice that the time the estimator takes to track the actual states is much shorter than the time the actual states take to reach a steady state. The estimator takes between 1 and 1.5 seconds to track the actual states, whereas the actual states take approximately six seconds to reach a steady state. The estimator tracks the actual states faster than the controller stabilizes the system because the estimator poles are at [–5, –5] and the controller poles are at [–1, –1]. Placing the poles of the estimator farther to

the left than the controller poles makes the performance of the estimator faster than the controller.

Example System Included State Compensator with Noise

In general, the compensator accepts two inputs, $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$. The input $\boldsymbol{r}_x$ represents state references. The input $\boldsymbol{r}_y$ represents measurement noise and is available only in the system included configuration with noise. Figure 14-16 shows the use of both types of inputs for the compensator.

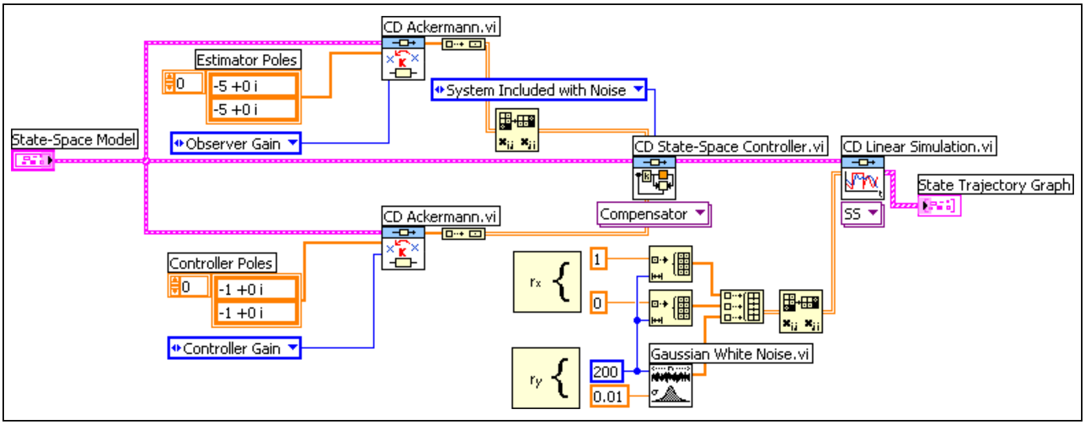

Figure 14-16. System Included State Compensator with Noise

The system included configuration with noise analyzes the effect of output noise on the system. This example has a total of three inputs to the compensator structure. The first two inputs are setpoints to the controller, given by $\boldsymbol{r}_x = [1, 0]$. The last input represents the output noise $\boldsymbol{r}_y$, which has a standard deviation of 0.01. Figure 14-17 shows the response to these inputs.

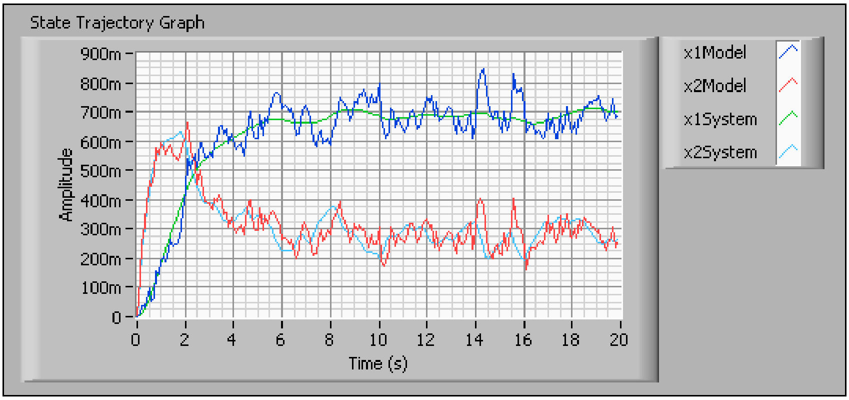

Figure 14-17. State Trajectory of System Included State Compensator with Noise

Notice that the state compensator lacks integral action, which originates the offsets on the state responses with respect to their respective setpoints. Therefore, the states do not reach the specified setpoints $r_x = [1, 0]$.

Example Standalone State Compensator with Estimator

Most systems are complex and have many parameters and uncertainties. You often do not know all the parameters of a system when you create a model of that system, or you cannot create a model that encompasses all the uncertainties of the system. Thus, the actual system and the model of the system do not match.

When you build a state compensator based on a model that does not match the actual system, the result is a system-model mismatch. In this situation, you need to use the standalone configuration with estimator. This configuration detaches the system from the model so you can determine the effect of the system-model mismatch. Consider the following state-space model:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} -0.2 & 0.5 \\ 0.1 & -0.2 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u}$$

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \end{bmatrix} \boldsymbol{u}$$

This model is similar to the model in the *Example System Configurations* section of this chapter. For this example, however, assume that the actual system contains uncertainties that cause this state-space model to be an

inaccurate representation of the system. The difference is in the last entry of the system matrix ***A***, –0.2.

Figure 14-18 shows how this configuration uses the mismatched model, **State-Space Model**, to create the standalone state compensator with estimator. Note that the CD State-Space Controller VI uses the **Compensator** instance. This configuration connects the actual system, **System**, and the mismatched model, **State-Space Model**, in series. **System** uses this connection to provide the output ***y*** to the state compensator.

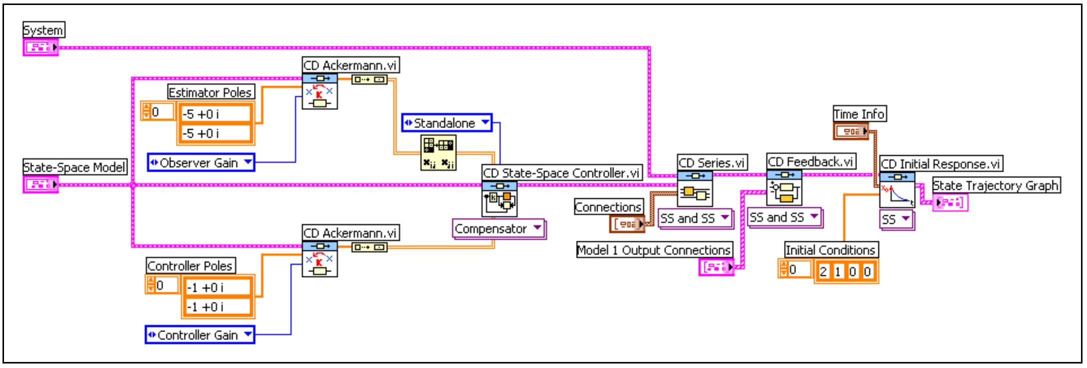

Figure 14-18. Standalone State Compensator with Estimator

This example sends the input ***u***, which the compensator calculates, to the actual system using the CD Feedback VI. The CD Initial Response VI uses the same initial conditions to test the effectiveness of the controller and estimator. Figure 14-19 shows the effect of a using a model that does not match the actual system.

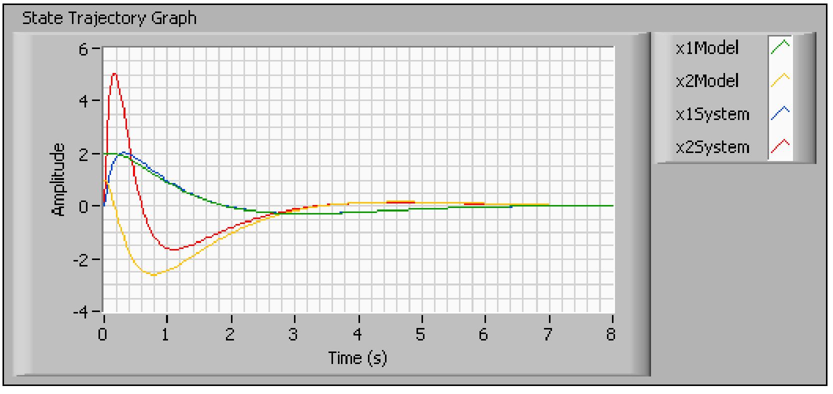

Figure 14-19. State Trajectory of Standalone State Compensator with Estimator

Notice how Figure 14-15 and Figure 14-19 respond differently even though both figures represent responses to the same system with the same initial conditions. The example in Figure 14-15 takes 1 to 1.5 seconds to track the actual states. The example in Figure 14-19, however, takes approximately four seconds to track the actual states. The system-model mismatch in the latter example accounts for this difference.

This example is similar to real-world applications where you do not know what the actual system is. Therefore, these tests are important in determining how sensitive the controller is to the system-model mismatches. You perform these tests before deploying the controller to a real-time (RT) target. Using a design method called robust control design, you can create model-based controllers that take into account possible modeling errors. Refer to *Essentials of Robust Control*[1] for information about robust control design.

Example Standalone State Compensator without Estimator

This state compensator uses the standalone configuration without estimator, which indicates that you do not need a state estimator because the states are directly available for control. The following equations describe the compensator model.

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{BK})\boldsymbol{x} + \boldsymbol{BK}\boldsymbol{r}_x$$
$$\boldsymbol{y} = \boldsymbol{Cx}$$

Note The direct transmission matrix $\boldsymbol{D}$ is not part of this expression because $\boldsymbol{D}$ is null in this example.

The poles, or the eigenvalues of $\boldsymbol{A} - \boldsymbol{B}K$, of the closed-loop system are in the left side of the complex plane. If you set the output noise $\boldsymbol{r}_x$ to zero, the controller gain matrix $\boldsymbol{K}$ immediately drives the states to zero.

Figure 14-20 shows how you use the CD Ackermann VI to calculate the controller gain matrix $\boldsymbol{K}$, which you then use to study the performance of the state compensator.

[1] Zhou, Kemin and John C. Doyle. *Essentials of Robust Control*, 1st ed. Upper Saddle River, NJ: Prentice Hall, 1998.

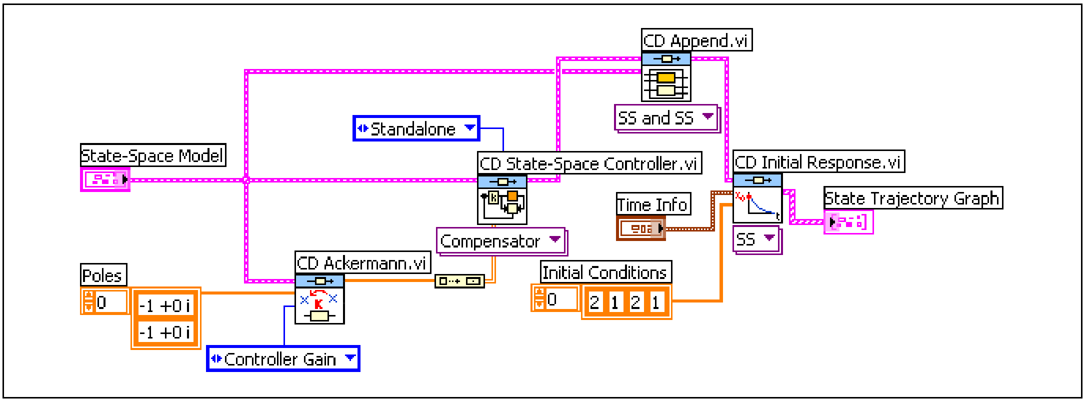

Figure 14-20. Standalone Compensator without Estimator

Note To view both the original response of the actual system and the response of the system controlled by the state compensator, you must append the model of the actual system, **State-Space Model**, to the model of the state compensator. Therefore, in the **State Trajectory Graph**, shown in Figure 14-21, you can see the difference in the system response due to the effect of the compensator gain ***K***.

By adding a state compensator to the actual system, you create a closed-loop model of the resulting system. The actual system, without a state compensator, is an open-loop system. Figure 14-21 shows the response of the open-loop and closed-loop systems to initial conditions of [2, 1].

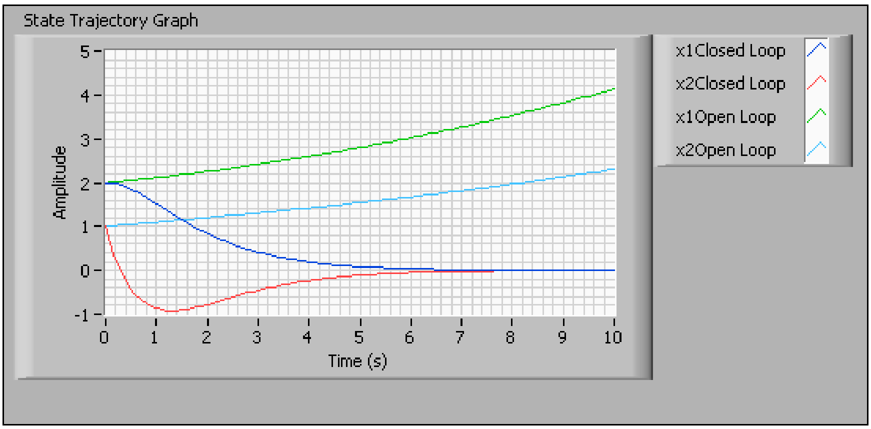

Figure 14-21. State Trajectory for Standalone Compensator without Estimator

Notice that despite the instability of the actual system, the state compensator is able to drive the closed-loop states toward zero. Thus the addition of a state compensator to the actual system stabilizes the resulting system.

Because the standalone state compensator stabilizes the actual system, you must use a state compensator with this system.

A

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at `ni.com` for technical support and professional services:

- **Support**—Online technical support resources at `ni.com/support` include the following:
  - **Self-Help Resources**—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Developer Exchange at `ni.com/exchange`. National Instruments Application Engineers make sure every question receives an answer.

    For information about other technical support options in your area, visit `ni.com/services` or contact your local office at `ni.com/contact`.
- **Training and Certification**—Visit `ni.com/training` for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit `ni.com/alliance`.

If you searched `ni.com` and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of `ni.com/niglobal` to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.