Modern Control Systems with LabVIEW™

Robert H. Bishop

nts PRESS

Modern Control Systems with LabVIEW™

Robert H. Bishop
Marquette University

NATIONAL TECHNOLOGY & SCIENCE PRESS

ISBN-13: 978-1-934891-18-6
ISBN-10: 1-934891-18-5

10 9 8 7 6 5 4 3 2

Publisher: Tom Robbins
General Manager: Erik Luther
Technical Oversight: Andy Chang
Development Editor: Catherine Peacock
Compositor: Paul Mailhot, PreTeX Inc.

CONTENTS

PREFACE

Welcome to Control System Design with LabVIEW™

As a companion to the textbook, *Modern Control Systems* by Richard C. Dorf and Robert H. Bishop, this supplement provides a set of comprehensive tutorials and exercises utilizing the LabVIEW Control Design and Simulation Module. NI LabVIEW is a graphical and textual language for prototyping real control and signal processing systems enabling a seamless flow from simulation to deployment on real-time control hardware. This text guides students through the process of modeling a system, analyzing the model to build a controller, and validating the robustness of the control system thorough simulation. Its contents and organization mirror the corresponding sections in MCS making it an ideal teaching companion, especially when LabVIEW is being utilized in a corresponding controls laboratory.

It is assumed readers will have access to LabVIEW 2010 or later, the Control Design and Simulation Module, and Mathscript RT. With these tools the reader can easily build, simulate, and analyze the examples and problems included in the text. Solution VIs for many of the problems are included for use as a programming reference and can also be used as a starting point for solving more advanced design problems. All of the LabVIEW examples were developed and tested on a PC compatible with LabVIEW Express 2010. The available VIs can be downloaded from http://www.ntspress.com/publications/modern-control-systems-with-labview.

For students unfamiliar with LabVIEW, it will be very helpful to have access to the *Learning with LabVIEW* textbook by Robert H. Bishop, available from Prentice Hall. For readers new to LabVIEW control, a wealth of documentation exists. Please see www.ni.com/academic/controls.htm.

We wish to express appreciation to Andy Chang at National Instruments and to Jorge Alvarez at The University of Texas at Austin for their work in creating the LabVIEW VIs and the screen captures used in the making of this companion text. Special thanks also to the folks at National Instruments, especially Erik Luther and Dr. Jeannie Falcon, for their continued support.

ROBERT H. BISHOP

CHAPTER 1

Mathematical Models of Systems

Application of the many classical and modern control system design and analysis tools is based on mathematical models. LabVIEW can be used with systems given in the form of transfer function descriptions. We begin this chapter by showing how to use LabVIEW to assist in the analysis of a typical spring-mass-damper mathematical model of a mechanical system. Using a LabVIEW virtual instrument (or VI) we can develop an interactive analysis capability to analyze the effects of natural frequency and damping on the unforced response of the mass displacement. We also discuss transfer functions and block diagrams. In particular, we are interested in how LabVIEW can assist us in manipulating polynomials, computing poles and zeros of transfer functions, computing closed-loop transfer functions, computing block diagram reductions, and computing the response of a system to a unit step input. The chapter concludes with a design example for an electric traction motor control design.

1.1 Simulating Spring-Mass-Damper Systems

A spring-mass-damper mechanical system is shown in Fig. 1.1. The motion of the mass, denoted by $y(t)$, is described by the differential equation

$$M\ddot{y}(t) + b\dot{y}(t) + ky(t) = r(t).$$

The unforced dynamic response of the spring-mass-damper mechanical system is

$$y(t) = \frac{y(0)}{\sqrt{1-\zeta^2}}\, e^{-\zeta\omega_n t} \sin\left(\omega_n\sqrt{1-\zeta^2}\, t + \theta\right),$$

where $\theta = \cos^{-1}\zeta$, $\omega_n^2 = k/M$ and $2\zeta\omega_n = b/M$. The initial displacement is $y(0)$ and $\dot{y}(0) = 0$. The transient system response is **underdamped** when $\zeta < 1$, **overdamped** when $\zeta > 1$, and **critically damped** when $\zeta = 1$.

Example 1.1 Spring-Mass-Damper Simulation

We can use LabVIEW to visualize the unforced time response of the mass displacement following an initial displacement of $y(0)$. Consider the underdamped case, where

$$y(0) = 0.15 \text{ m}, \quad \omega_n = \sqrt{2} \text{ rad/sec}, \quad \zeta = \frac{1}{2\sqrt{2}}, \quad (k/M = 2,\ b/M = 1).$$

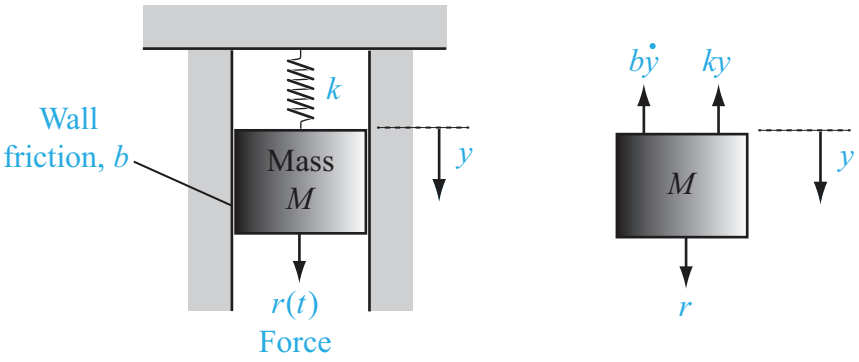


Figure 1.1: A mass-spring-damper system.

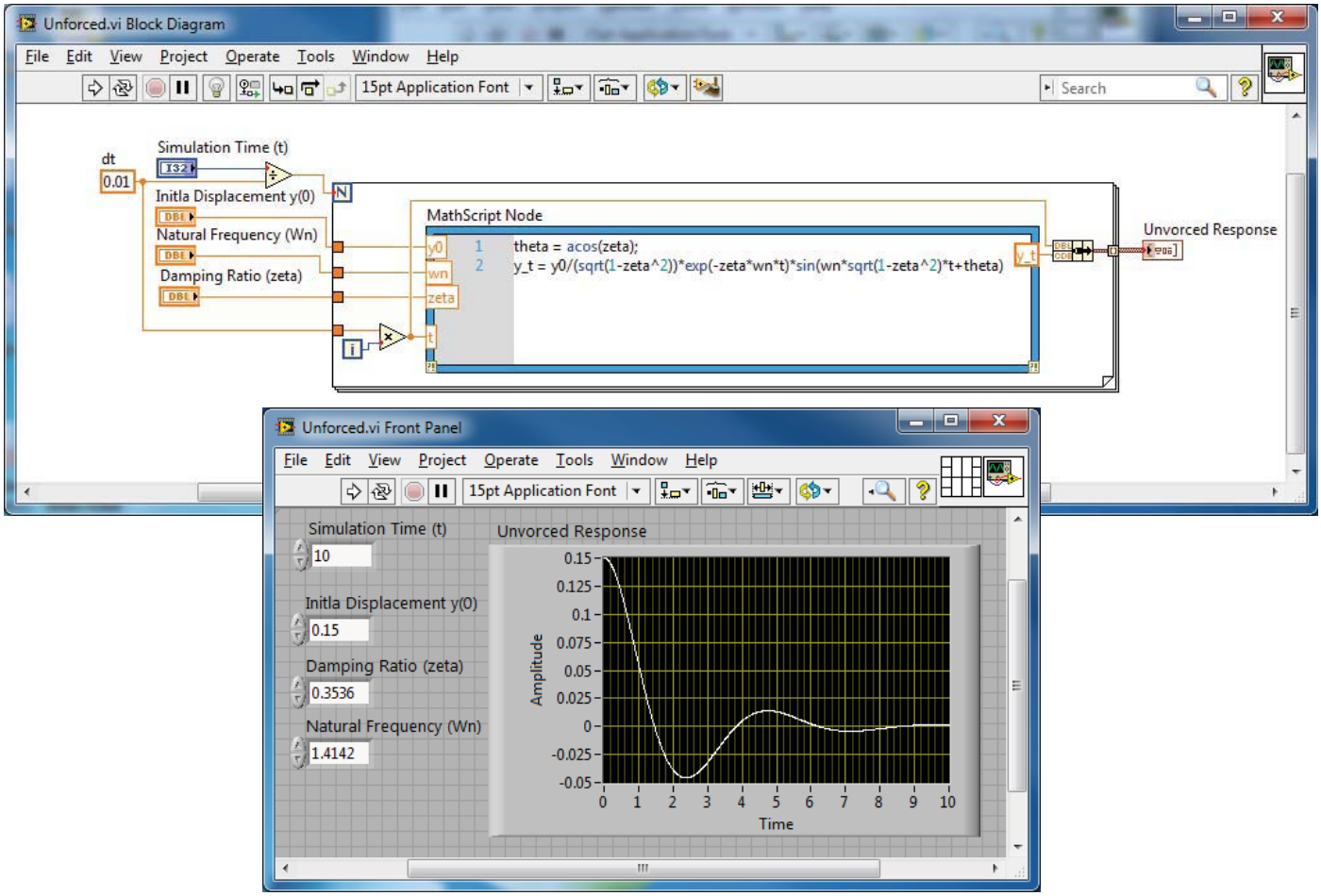


Figure 1.2: VI to analyze the spring-mass-damper.

The LabVIEW commands to generate the plot of the unforced response are shown in Fig. 1.2.

In the LabVIEW setup, the variables $y(0)$, ω_n, ζ, and t are input to the user interface part of the VI. Then the Unforced.vi is executed to generate the desired plots. This creates an interactive analysis capability to analyze the effects of natural frequency and damping on the unforced response of the mass displacement. One can investigate the effects of the natural frequency and the damping on the time response by simply entering new values of ω_n and ζ and rerun the Unforced.vi. ◇

For the spring-mass-damper problem, the unforced solution to the differential equation was readily available. In general, when simulating closed-loop feedback control systems subject to a variety of inputs and initial conditions, it is difficult to obtain the solution analytically. In these cases we can use LabVIEW to compute the solutions numerically and to display the solution graphically.

1.2 Analyzing Systems Using LabVIEW

LabVIEW can be used to analyze systems described by transfer functions. Since the transfer function is a ratio of polynomials, we begin by investigating how LabVIEW handles polynomials, remembering that working with transfer functions means that both a numerator polynomial and a denominator polynomial must be specified. In LabVIEW, polynomials are represented by row vectors containing the polynomial coefficients. For example, the polynomial

$$P(s) = s^3 + 3s^2 + 2s + 3$$

is entered as [3 2 3 1] as shown in Fig. 1.3.

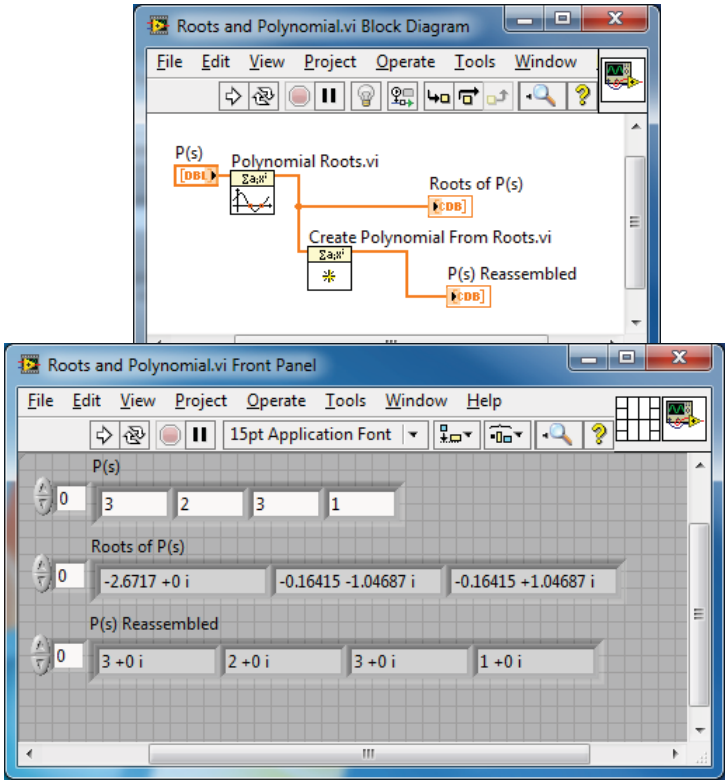


Figure 1.3: Entering the polynomial $P(s) = s^3 + 3s^2 + 2s + 3$ and calculating the roots of $P(s) = 0$.

If **p** is a row vector containing the coefficients of $P(s)$ in ascending order, then Polynomial Roots (**p**) is a row vector containing the roots of the polynomial. Conversely, if **r** is a row vector containing the roots of the polynomial, then Create Polynomial from Roots(**r**) is a row vector with the polynomial coefficients in ascending order. We can compute the roots of the polynomial $P(s) = s^3 + 3s^2 + 2s + 3$ with the Polynomial Roots function. In Fig. 1.3, we show how to reassemble the polynomial with the Create Polynomial from Roots function.

Multiplication of polynomials is accomplished with the Multiply Polynomials.vi function. Suppose we want to expand the polynomial $N(s)$, where

$$N(s) = (3s^2 + 2s + 1)(s + 4).$$

The associated LabVIEW commands using the Multiply Polynomials.vi function are shown in Fig. 1.4. Thus, the expanded polynomial is

$$N(s) = 3s^3 + 14s^2 + 9s + 4.$$

The function Polynomial Evaluation is used to evaluate the value of a polynomial at the given value of the variable. The polynomial $N(s)$ has the value $N(-5) = -66$, as shown in Fig. 1.4.

The LabVIEW Control Design & Simulation Toolbox treats linear, time-invariant system models as **objects**, allowing you to manipulate the system models as single entities. In the case of transfer functions, you create the system models using the CD Construct Transfer Function Model; for state variable models you employ the CD Construct State Space Function Model. The use of CD Construct Transfer Function Model is illustrated in Fig. 1.5.

Based on the LabVIEW object-oriented programming capabilities, the system model objects possess object properties that can be modified; likewise functions that operate on system model objects are called *methods*. For example, if you have the two system models

$$G_1(s) = \frac{10}{s^2 + 2s + 5} \qquad \text{and} \qquad G_2(s) = \frac{1}{s + 1},$$

you can add them using the Add Rational Polynomials.vi function to obtain

$$G(s) = G_1(s) + G_2(s) = \frac{s^2 + 12s + 15}{s^3 + 3s^2 + 7s + 5}.$$

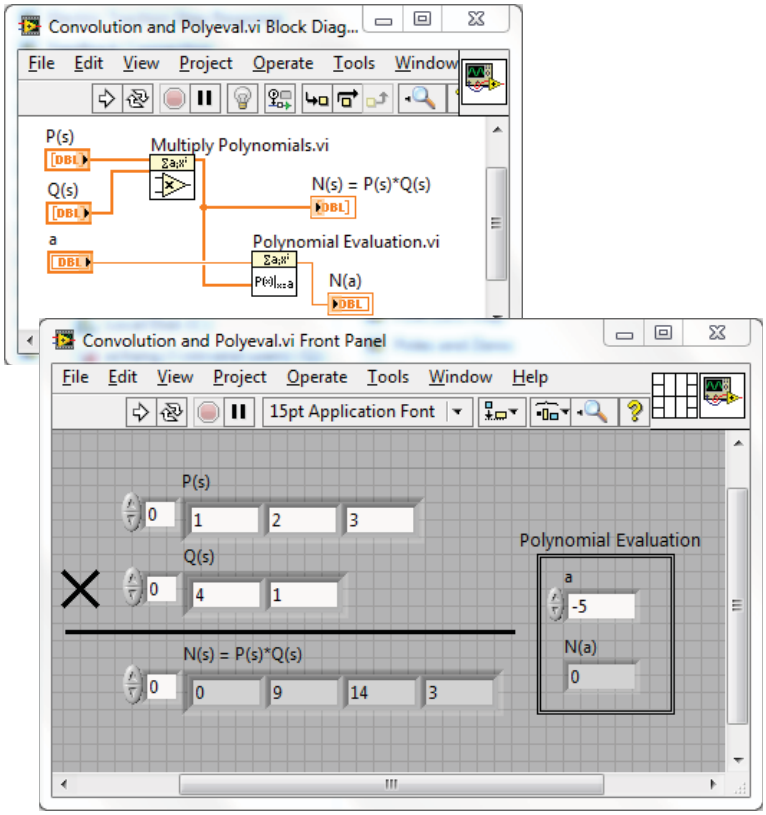


Figure 1.4: Using Multiply Polynomials and Polynomial Evaluation to multiply and evaluate the polynomials $N(s) = (3s^2 + 2s + 1)(s + 4)$.

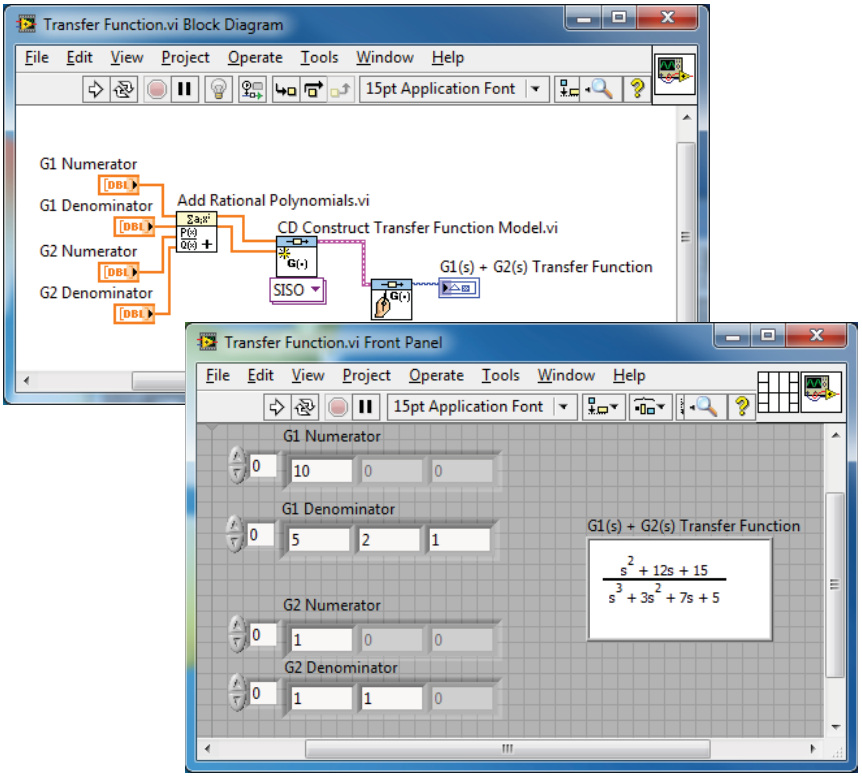


Figure 1.5: Using the CD Construct Transfer Function Model, the Add Polynomials, and the Draw Transfer Function Equations functions.

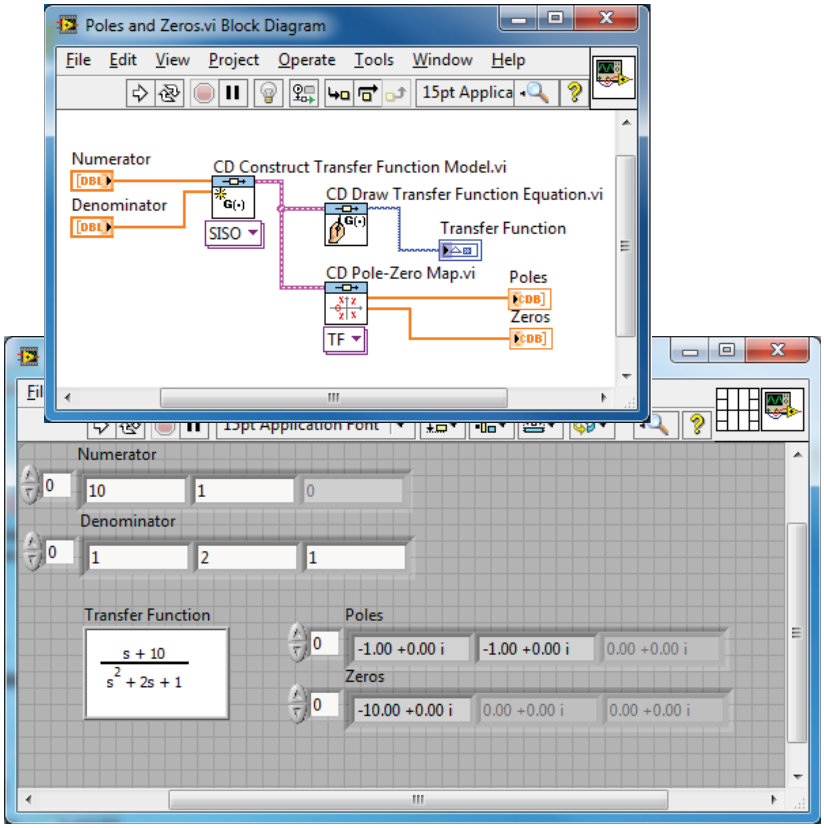


Figure 1.6: Using the CD Pole-Zero Map function to compute the poles and zeros of a linear system.

The corresponding LabVIEW commands are shown in Fig. 1.5. Computing the poles and zeros associated with a transfer function is accomplished by operating on the system model object with the CD Pole-Zero Map function, as illustrated in Fig. 1.6. In the next example, we obtain a plot of the pole and zero locations in the complex plane. On the pole and zero map, zeros are denoted by an “o" and poles are denoted by a “x".

Example 1.2 Transfer Functions

Consider the transfer functions

$$G(s) = \frac{6s^2 + 1}{s^3 + 3s^2 + 3s + 1}$$

and

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

Utilizing LabVIEW, we can compute the poles and zeros of $G(s)$, the characteristic equation of $H(s)$, and divide $G(s)$ by $H(s)$. We can also obtain a plot of the pole-zero map of $G(s)/H(s)$ in the complex plane. The pole-zero map of the transfer function $G(s)/H(s)$ and the associated LabVIEW commands are shown in Fig. 1.7.

The pole-zero map shows clearly the five zero locations, but it appears that there are only two poles. This cannot be the case, since we know that the number of poles must be greater than or equal to the number of zeros. Using the CD Pole-Zero Map function, we ascertain that there are in fact four poles at $s = -1$. Hence, multiple poles or multiple zeros at the same location cannot be discerned on the pole-zero map. ◇

1.3 Block Diagram Models

Suppose we have developed mathematical models in the form of transfer functions for the plant, represented by $G(s)$, and the controller, represented by $G_c(s)$, and possibly other system components such as sensors and actuators. Our objective is to interconnect

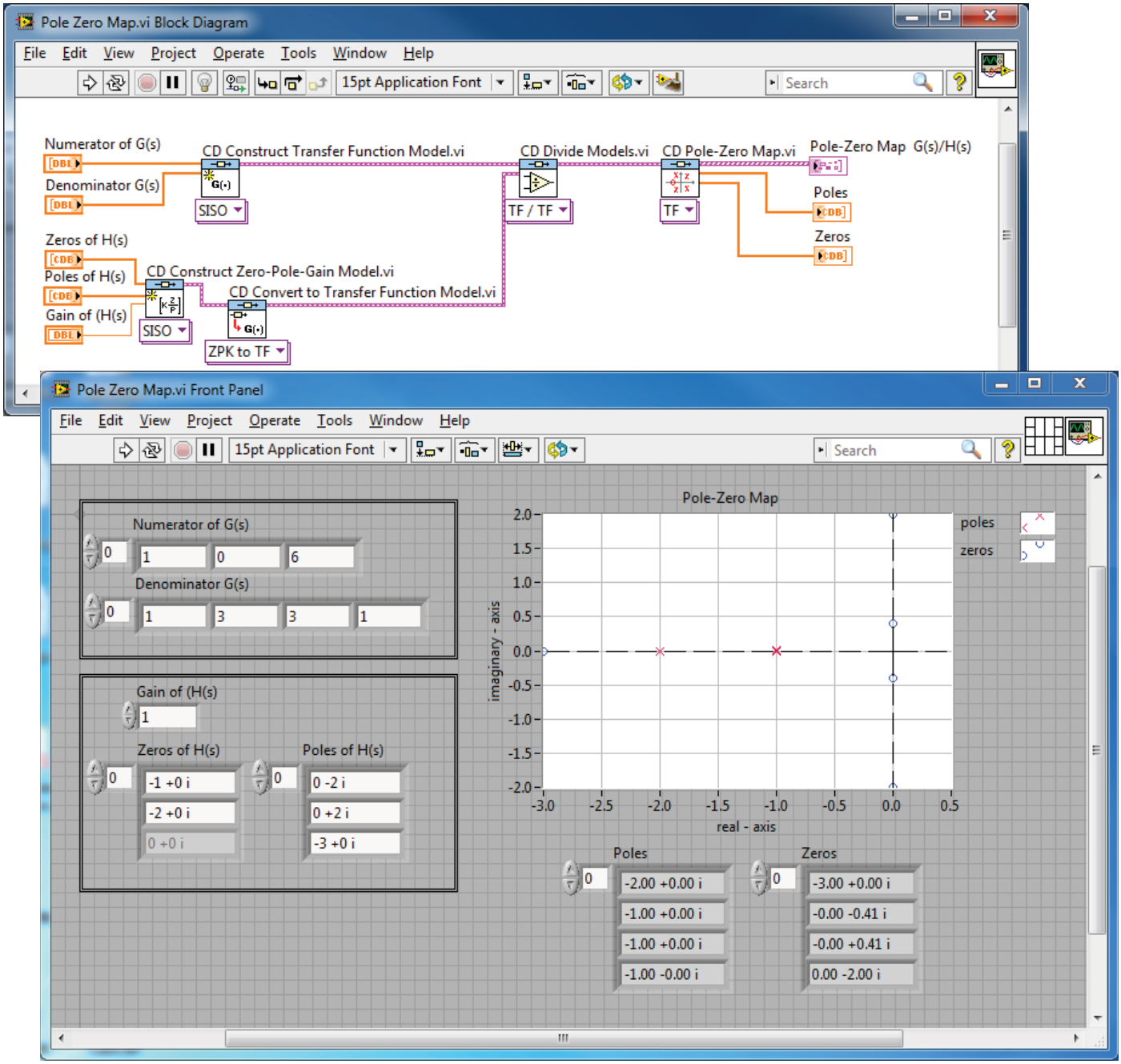


Figure 1.7: The CD Pole-Zero Map function.

these components to form a control system. We can use LabVIEW functions to carry out the block diagram transformations. A simple open-loop control system can be obtained by interconnecting a plant and a controller in series as illustrated in Fig. 1.8. We can use LabVIEW to compute the transfer function from $R(s)$ to $Y(s)$, as will be illustrated in Example 1.3.

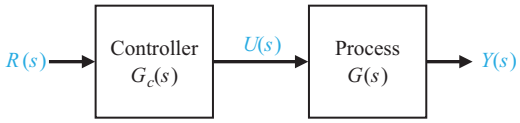


Figure 1.8: Open-loop control system (without feedback).

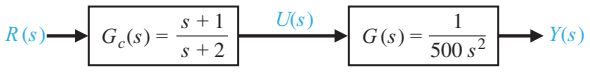


Figure 1.9: Series connection block diagram.

Example 1.3 Series Connection

Let the process represented by the transfer function $G(s)$ be

$$G(s) = \frac{1}{500s^2},$$

and let the controller represented by the transfer function $G_c(s)$ be

$$G_c(s) = \frac{s+1}{s+2}.$$

Suppose we want to cascade two transfer functions $G(s)$ and $G_c(s)$, as illustrated in Fig. 1.9. The transfer function $G_c(s)G(s)$ is computed using the CD Series function as shown in Fig. 1.10.

The resulting transfer function, $G_c(s)G(s)$, is

$$G_c(s)G(s) = \frac{s+1}{500s^3 + 1000s^2}.$$

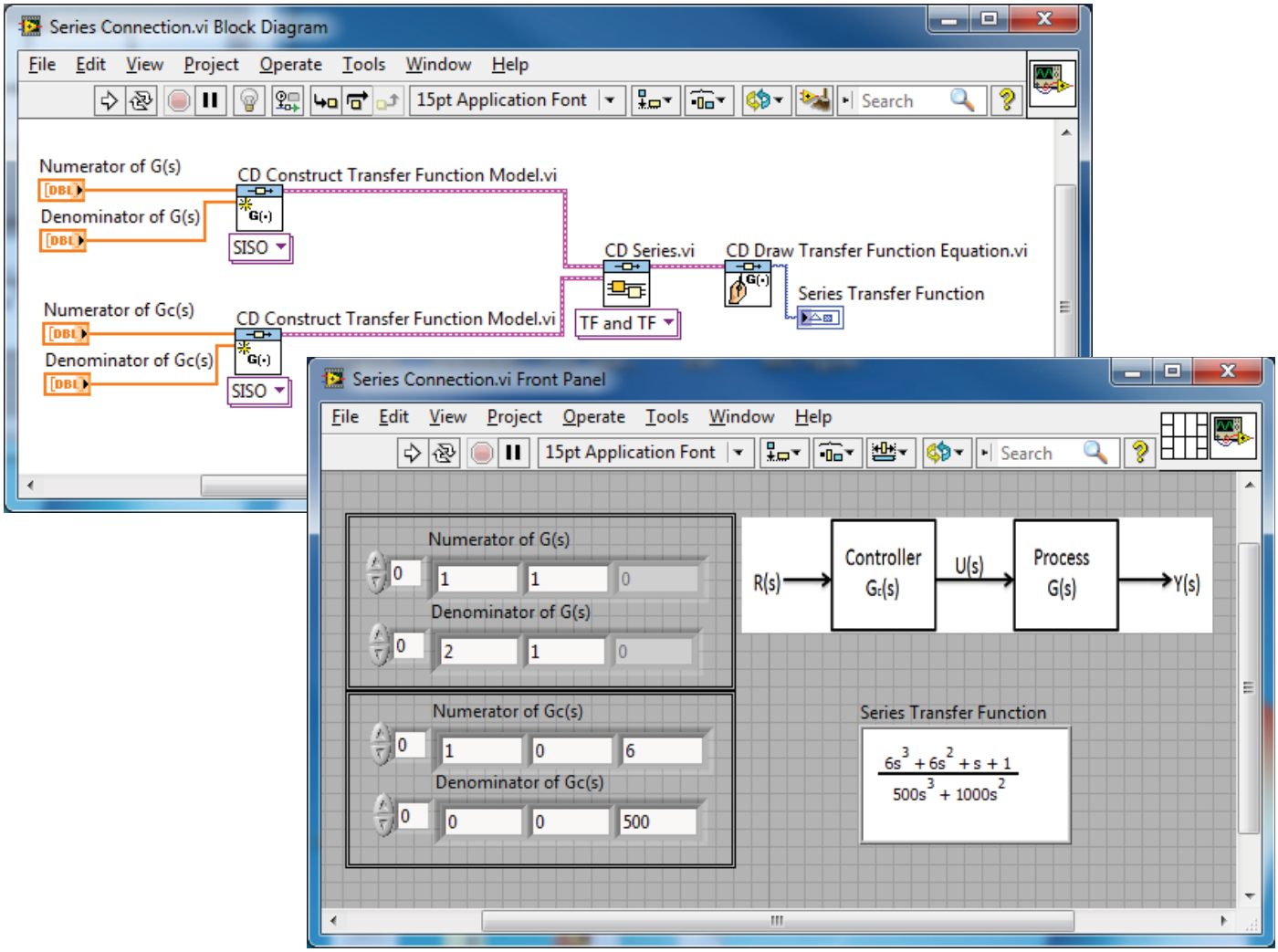


Figure 1.10: The CD Series vi.

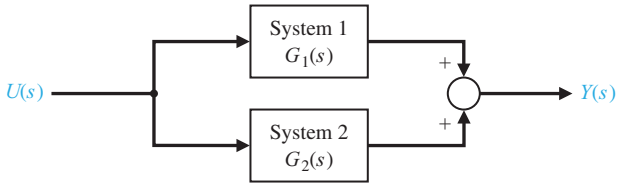


Figure 1.11: Parallel connection block diagram.

◇

Block diagrams often have transfer functions in parallel as shown in Fig. 1.11. In such cases, the function CD Parallel can be useful.

Example 1.4 Parallel Connection

Suppose we desire to connect two systems $G_1(s)$ and $G_2(s)$ in parallel as shown in Fig. 1.11, where

$$G_1(s) = \frac{1}{500s^2} \quad \text{and} \quad G_2(s) = \frac{s+1}{s+2}.$$

The CD Parallel function is illustrated in Fig. 1.12. The resulting transfer function $G_1(s)G_2(s)$ is

$$G_1(s)G_2(s) = \frac{500s^3 + 500s^2 + s + 2}{500s^3 + 1000s^2}.$$

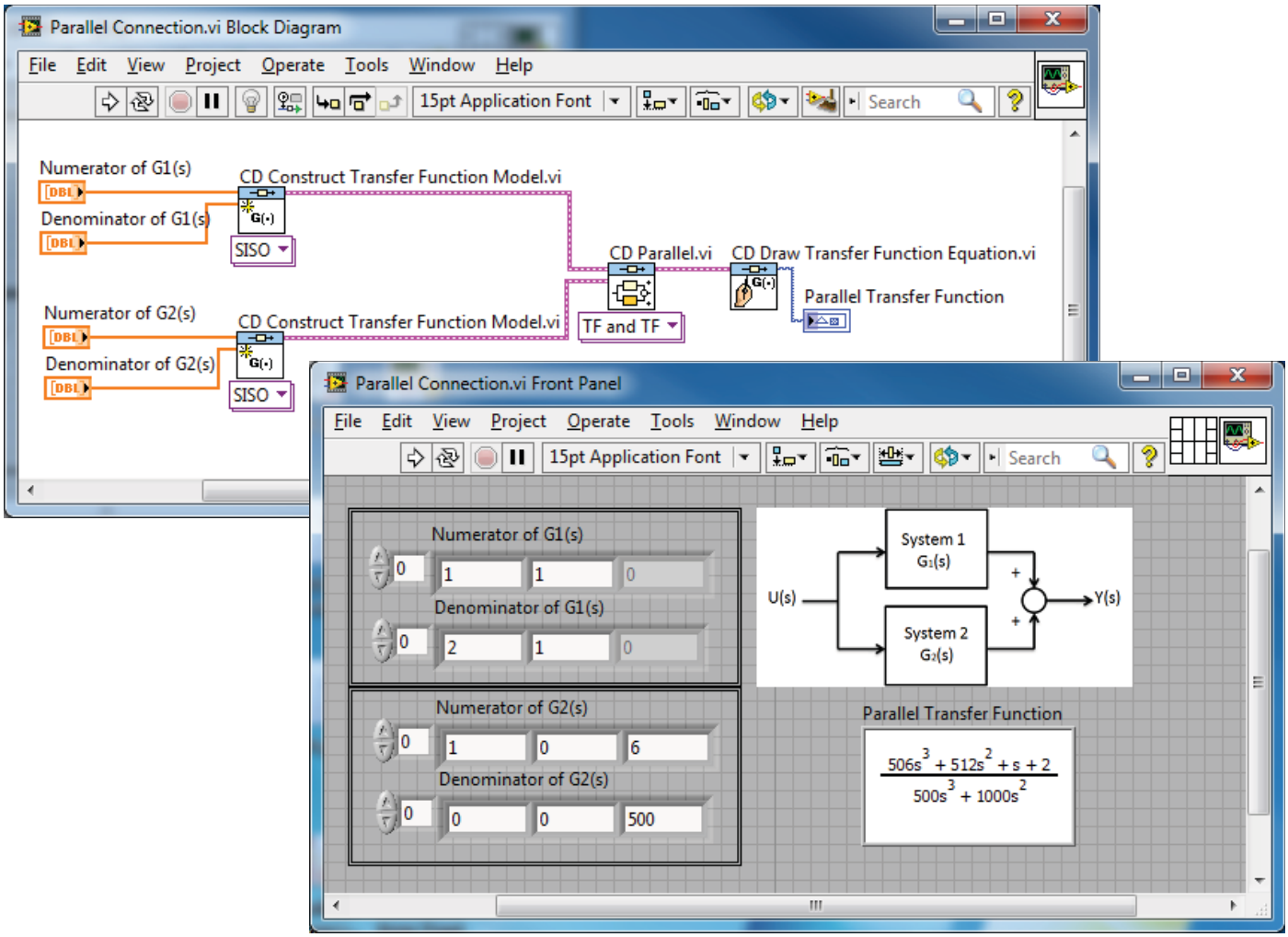


Figure 1.12: The CD Parallel vi.

◇

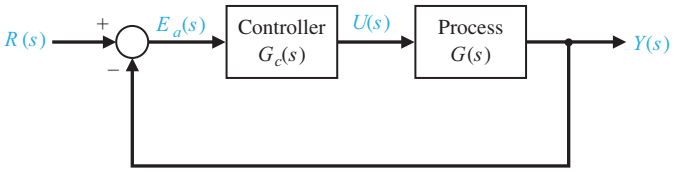


Figure 1.13: A basic control system with unity feedback.

We can introduce feedback into the control system by closing the loop with **unity** feedback, as shown in Fig. 1.13. The signal $E_a(s)$ is an **error signal**; the signal $R(s)$ is a **reference input**. In this control system, the controller is in the forward path and the closed-loop transfer function is

$$T(s) = \frac{G_c(s)G(s)}{1 \mp G_c(s)G(s)}.$$

We can utilize the CD Feedback function to aid in the block diagram reduction process to compute closed-loop transfer functions for single- and multiple-loop control systems. When the closed-loop control system has unity feedback, we can use the CD Unit Feedback function to compute the closed-loop transfer function.

Example 1.5 Unity Feedback

In this example, we consider the CD Unit Feedback function. Let the process, $G(s)$, and the controller, $G_c(s)$, be as in Fig. 1.14. We compute $G_c(s)G(s)$ using the CD Unit Feedback function to close the loop. The command sequence is shown in Fig. 1.15 and results in the closed-loop transfer function

$$T(s) = \frac{G_c(s)G(s)}{1 + G_c(s)G(s)} = \frac{s+1}{500s^3 + 1000s^2 + s + 1}.$$

◇

Another basic feedback control configuration is shown in Fig. 1.16. In this case, the controller is located in the feedback path. The closed-loop transfer function is

$$T(s) = \frac{G(s)}{1 \mp G(s)H(s)}.$$

The use of the CD Feedback function for a nonunity feedback system is illustrated in Example 1.6.

Example 1.6 Nonunity Feedback

Let the process, $G(s)$, and the controller, $H(s)$, be as in Fig. 1.17. To compute the closed-loop transfer function with the controller in the feedback loop we use the CD Feedback function as shown in Fig. 1.18. The closed-loop transfer function is

$$T(s) = \frac{s+1}{500s^3 + 1000s^2 + s + 1}.$$

◇

The LabVIEW functions CD Series, CD Parallel, and CD Feedback can be used as aids in block diagram manipulations for multiple-loop block diagrams, as will be illustrated in Example 1.7.

Example 1.7 Multiloop Reduction

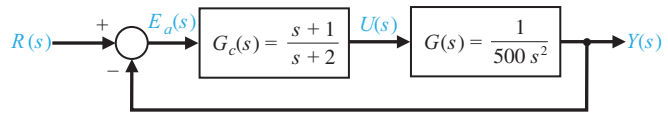


Figure 1.14: Feedback connection block diagram.

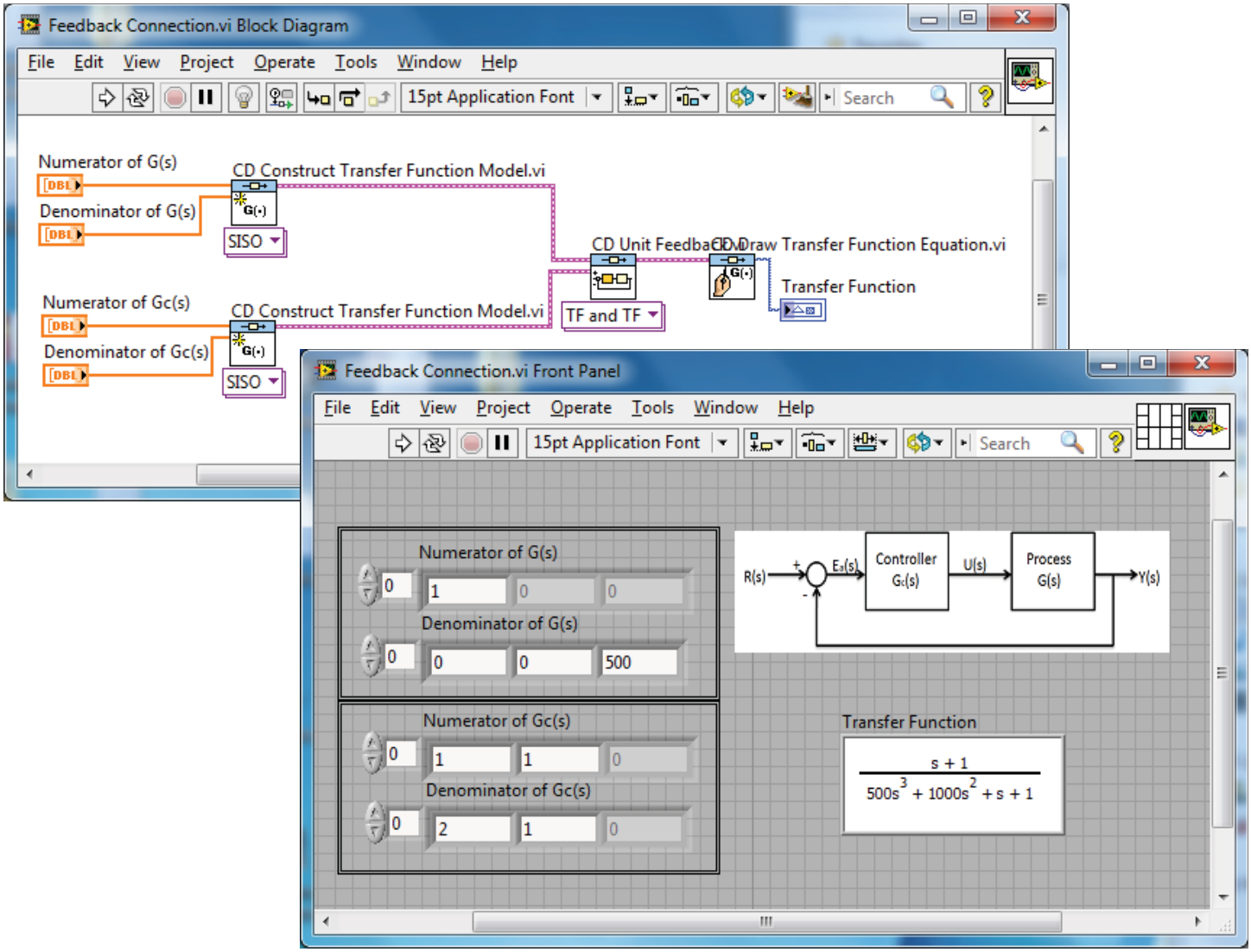


Figure 1.15: Application of the CD Feedback function.

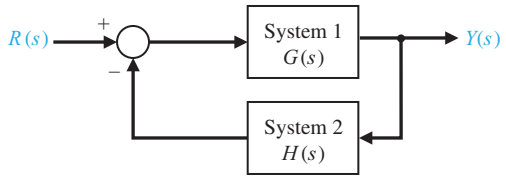


Figure 1.16: A basic control system with the controller in the feedback loop.

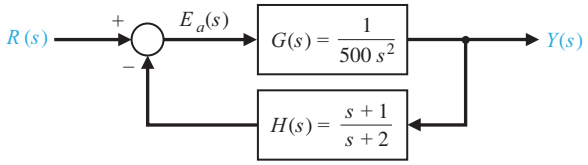


Figure 1.17: Application of the CD feedback.

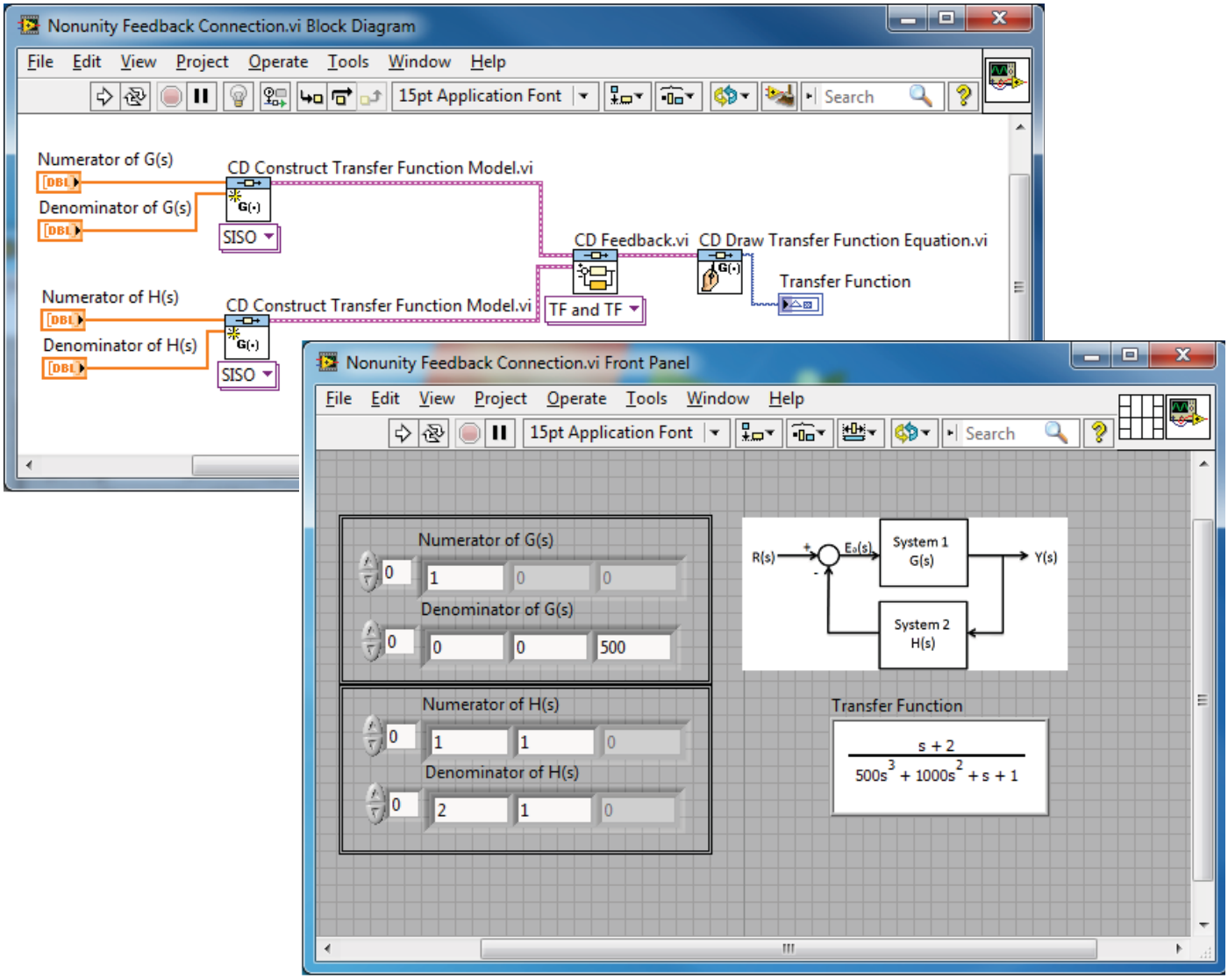


Figure 1.18: Application of the CD feedback function for nonunity feedback.

A multiloop feedback system is shown in Fig. 1.19. The objective is to compute the closed-loop transfer function

$$T(s) = \frac{Y(s)}{R(s)},$$

when

$$G_1(s) = \frac{1}{s+10}, \qquad G_2(s) = \frac{1}{s+1},$$
$$G_3(s) = \frac{s^2+1}{s^2+4s+4}, \qquad G_4(s) = \frac{s+1}{s+6},$$

and

$$H_1(s) = \frac{s+1}{s+2}, \qquad H_2(s) = 2, \qquad H_3(s) = 1.$$

For this example, a five-step procedure is followed:

Step 1: Input the system transfer functions into LabVIEW.

Step 2: Move H_2 behind G_4.

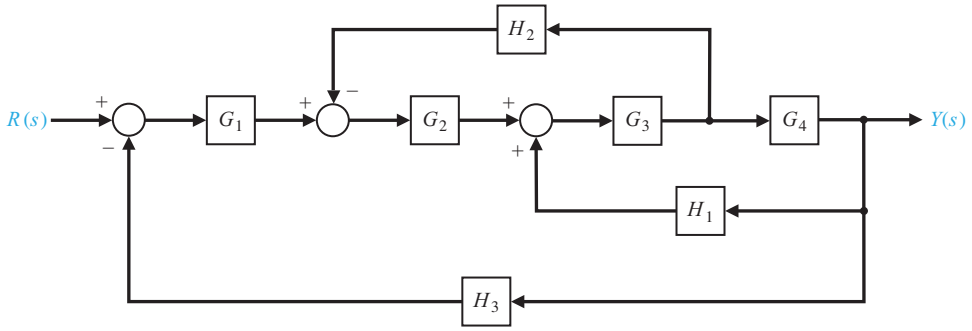


Figure 1.19: A multiloop feedback system.

Step 3: Eliminate the $G_3G_4H_1$ loop.

Step 4: Eliminate the loop containing H_2.

Step 5: Eliminate the remaining loop and calculate $T(s)$.

The five steps are utilized in Fig. 1.20, and the corresponding block diagram reduction is shown in Fig. 1.21.

The result of executing the LabVIEW commands is

$$\overline{T}(s) = \frac{0.083s^4 + 0.25s^3 + 0.25s^2 + 0.25s + 0.167}{s^5 + 16.083s^4 + 72.75s^3 + 137s^2 + 123{,}667s + 59.333}.$$

If we had performed the calcuations by hand, we would have obtained

$$\overline{T}(s) = \frac{s^5 + 4s^4 + 6s^3 + 6s^2 + 5s + 2}{12s^6 + 205s^5 + 1066s^4 + 2517s^3 + 3128s^2 + 2196s + 712}.$$

We must be careful in calling this the closed-loop transfer function. The transfer function is defined to the input–output relationship after pole–zero cancellations. If we compute the poles and zeros of $\overline{T}(s)$, we find that the numerator and denominator polynomials have $(s + 1)$ as a common factor. This must be canceled before we claim we have the closed-loop transfer function. To assist us in the pole-zero cancellation, we can use the CD Minimal Realization function. The CD Minimal Realization function, shown in Fig. 1.22, removes common pole-zero factors of a transfer function. After the application of the CD Minimal Realization function, we find that the order of the denominator polynomial has been reduced from six to five, implying one pole-zero cancellation. ◇

Example 1.8 Electric Traction Motor Control

In this example we consider the electric traction motor system represented by the block diagram shown in Fig. 1.23.

The objective is to compute the closed-loop transfer function and investigate the response of $\omega(s)$ to a commanded $\omega_d(s)$. The first step, as shown in Fig. 1.24, is to compute the closed-loop transfer function $\omega(s)/\omega_d(s) = T(s)$.

The closed-loop characteristic equation is second-order with $\omega_n = 52$ and $\zeta = 0.012$. Since the damping is low, we expect the response to be highly oscillatory. We can investigate the response $\omega(t)$ to a reference input, $\omega_d(t)$, by utilizing the CD Step Response function. The CD Step Response function calculates the unit step response of a linear system. The step response of the electric traction motor is shown in Fig. 1.25. As expected, the wheel velocity response, given by $y(t)$, is highly oscillatory. Note that the output is $y(t) = \omega(t)$.

◇

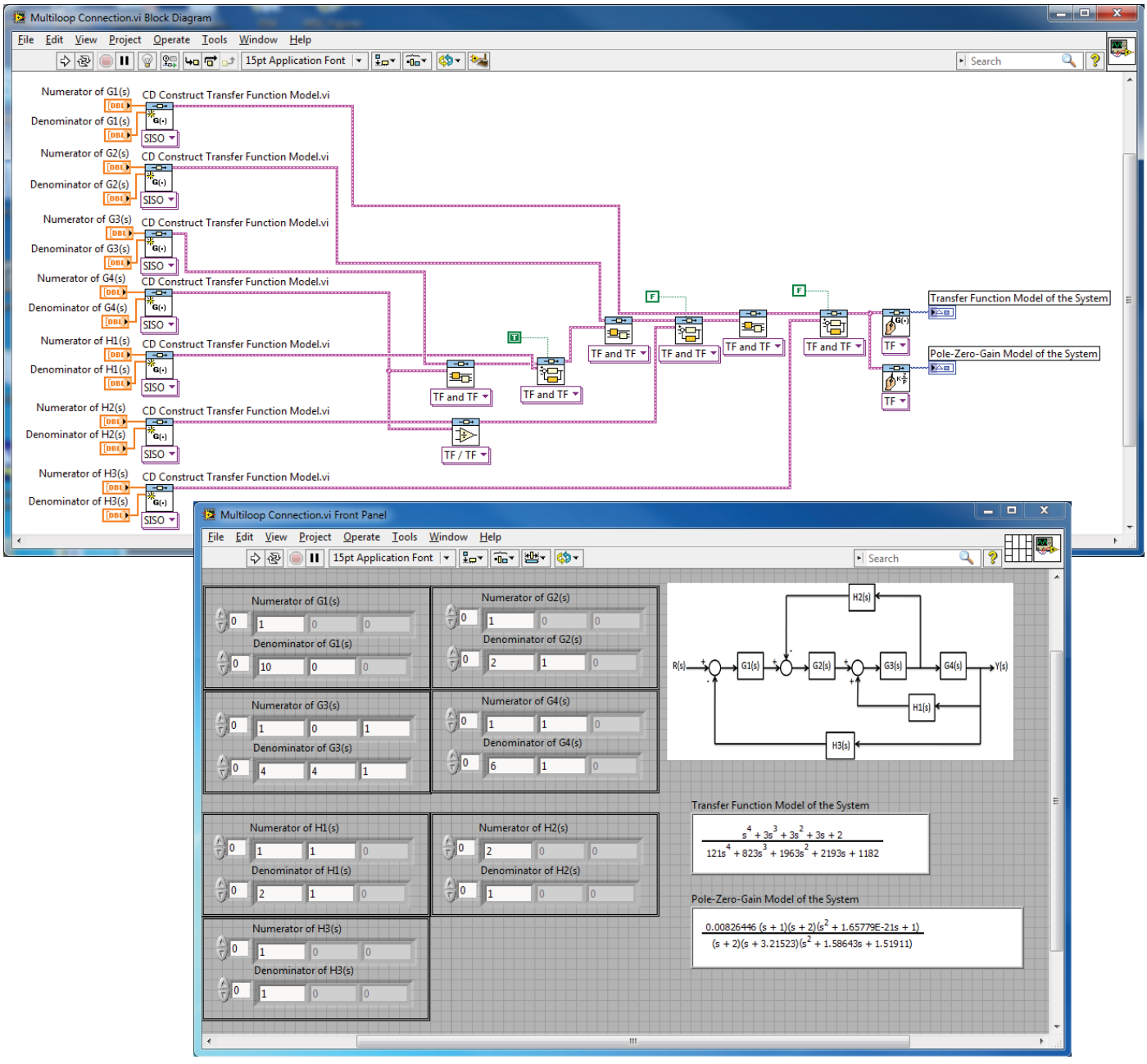
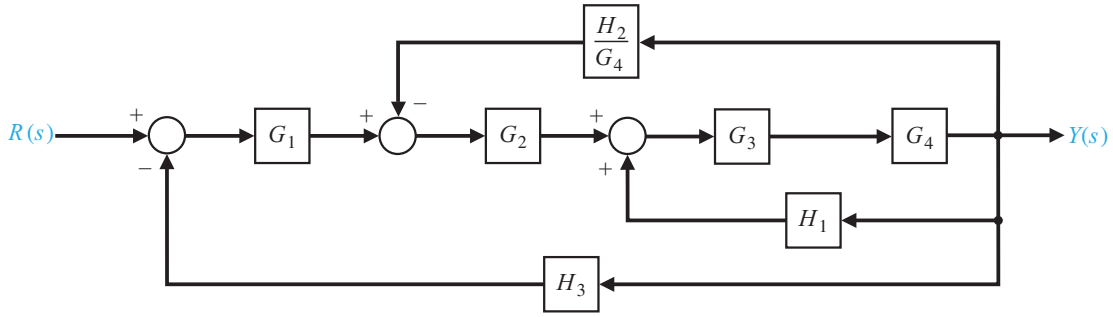
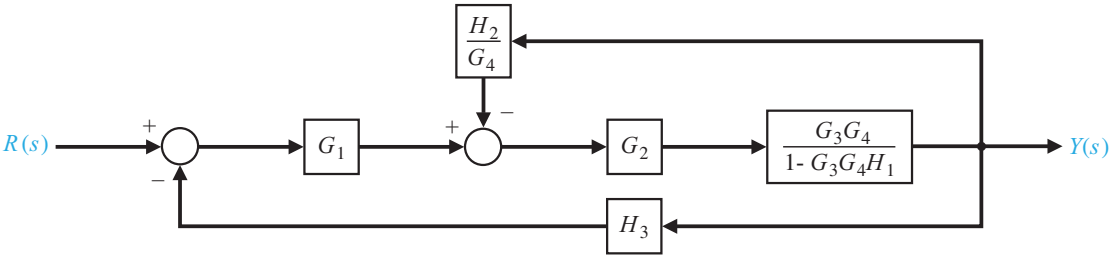


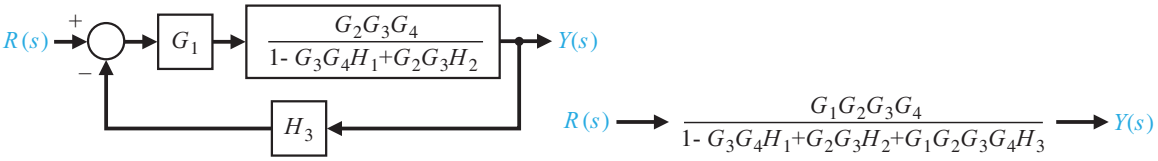
Figure 1.20: Reducing the multiloop feedback system.



(a) Step 2



(b) Step 3



(c) Step 4

(d) Step 5

Figure 1.21: Block diagram reduction of the system in Fig. 1.20.

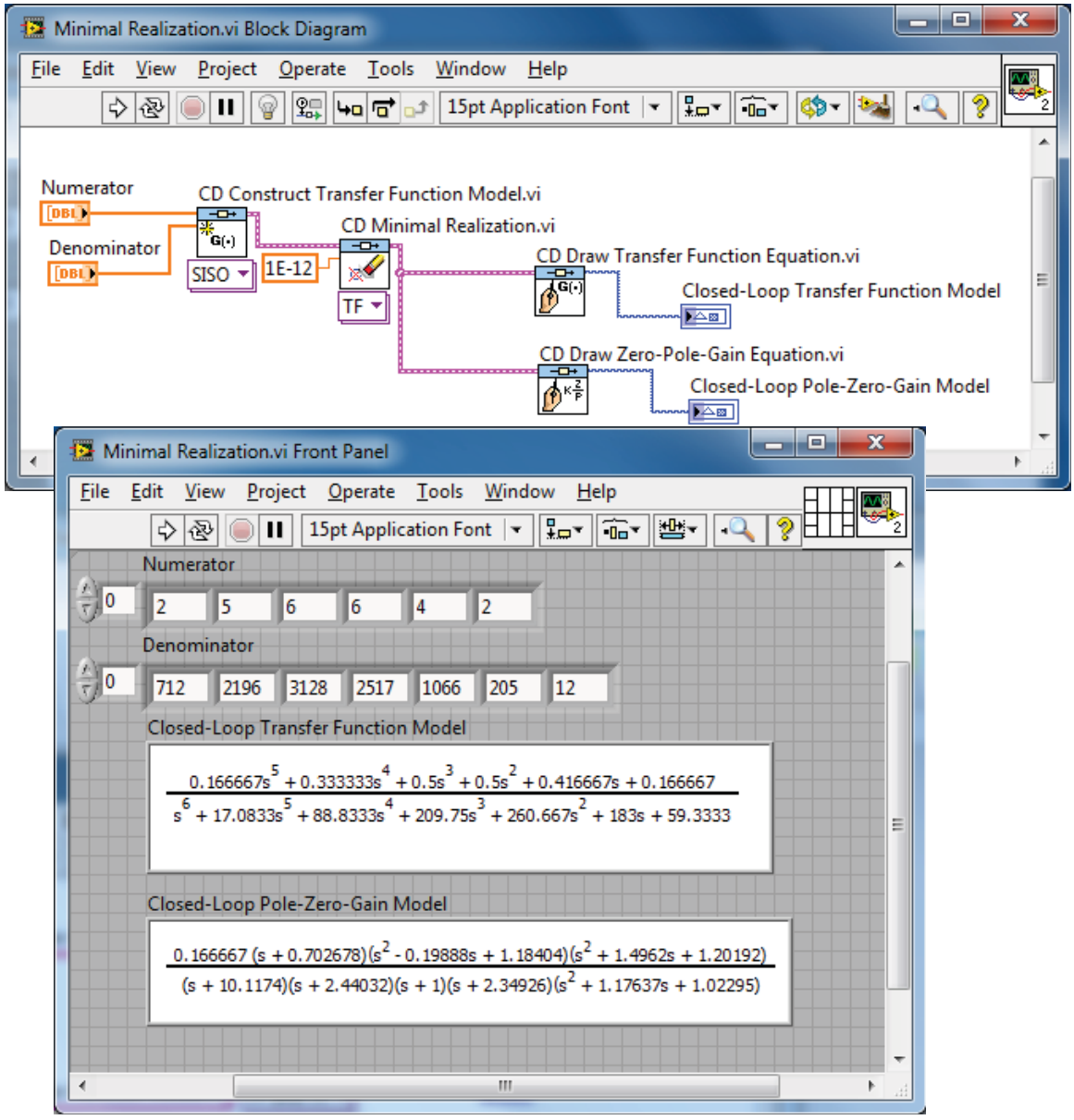


Figure 1.22: Application of the CD Minimal Realization function.

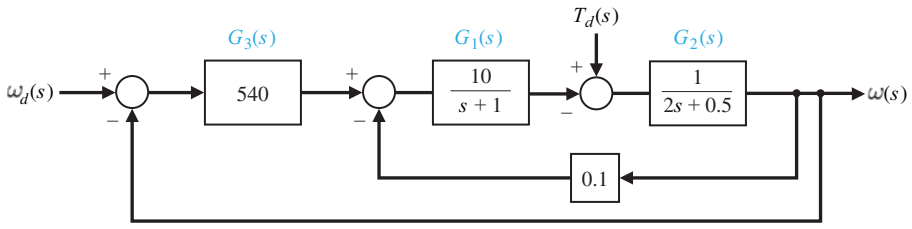


Figure 1.23: Block diagram of an electric traction motor system.

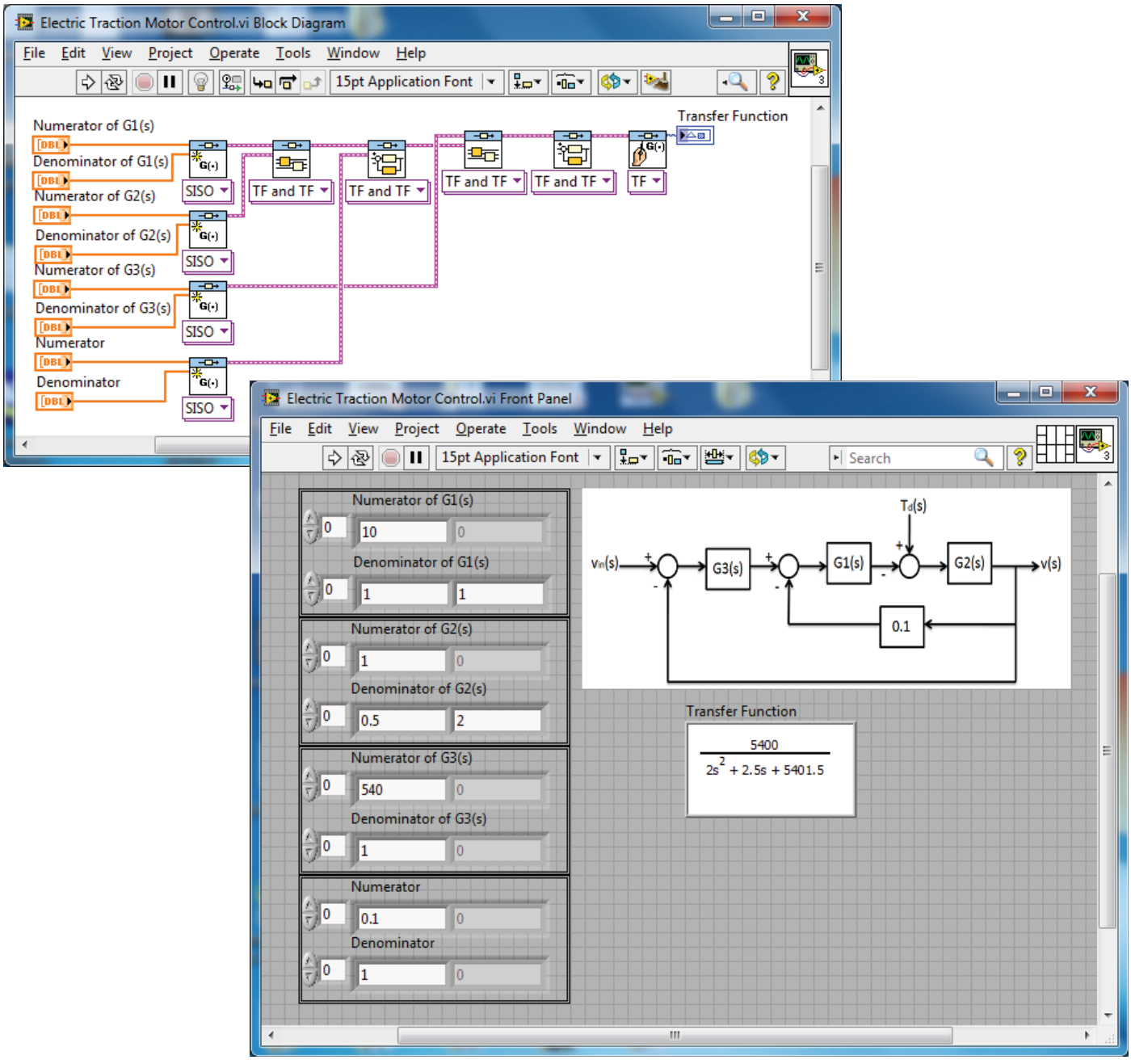


Figure 1.24: Electric traction motor block reduction.

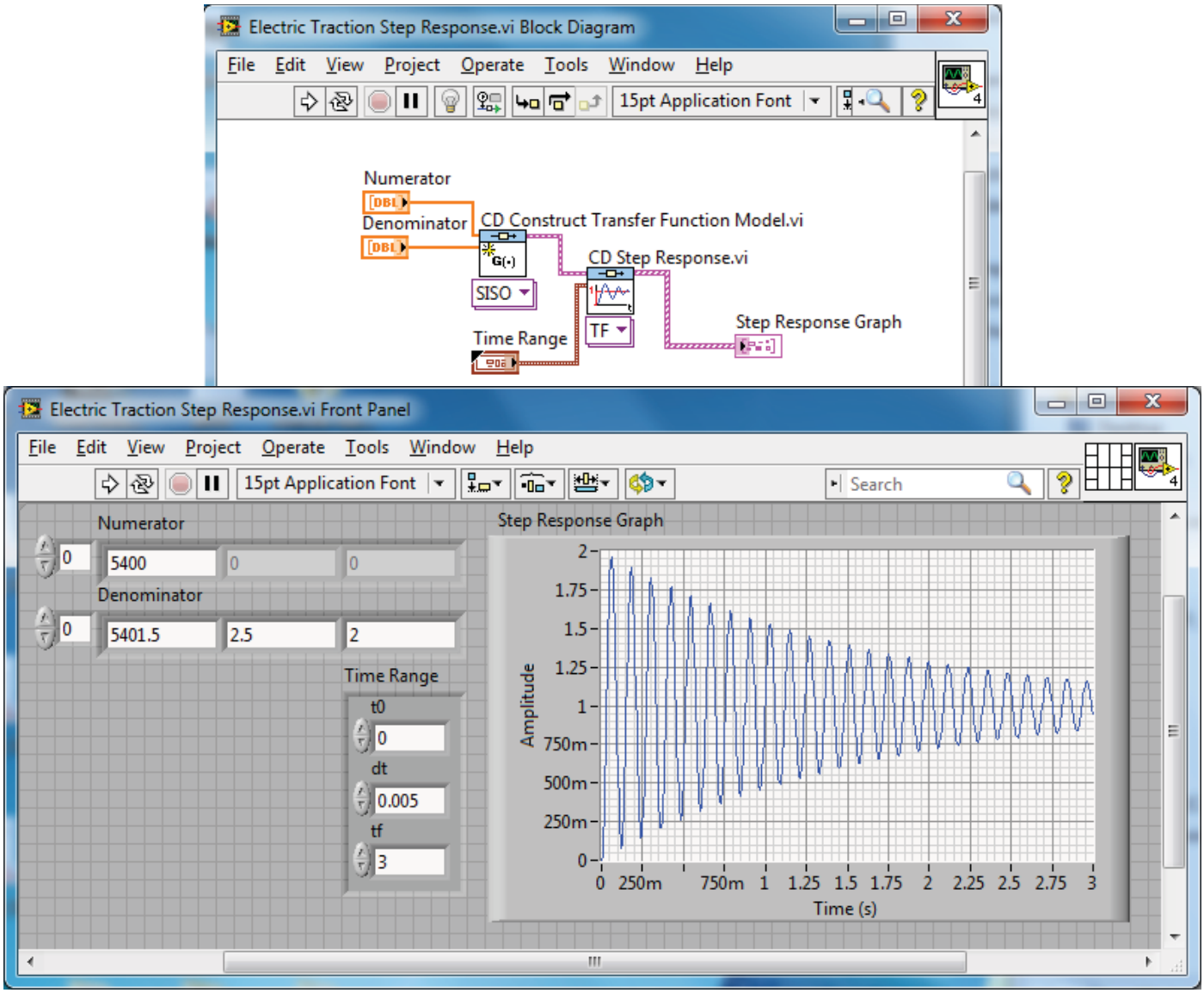


Figure 1.25: The CD Step Response function.

CHAPTER 2

State Variable Models

The time-domain method utilizes a **state-space representation** of the system model, given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \qquad \text{and} \qquad y = \mathbf{C}\mathbf{x} + \mathbf{D}u. \tag{2.1}$$

The vector $\mathbf{x}$ is the state of the system, $\mathbf{A}$ is the constant $n \times n$ system matrix, $\mathbf{B}$ is the constant $n \times m$ input matrix, $\mathbf{C}$ is the constant $p \times n$ output matrix, and $\mathbf{D}$ is a constant $p \times m$ matrix. The number of inputs, m, and the number of outputs, p, are taken to be one, since we are considering only single-input, single-output (SISO) problems. Therefore y and u are not bold (matrix) variables. The main elements of the state-space representation in Eq. (2.1) are the state vector $\mathbf{x}$, the input u, the output y, and the constant matrices ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$).

Given a transfer function, we can obtain an equivalent state-space representation and vice versa. The function CD Convert to State-Space Model is used to convert a transfer-function representation to a state-space representation; the function CD Convert to Transfer Function Model is used to convert a state-space representation to a transfer function. An analysis of a third-order system is presented in Example 2.1.

Example 2.1 Third-Order System

Consider the third-order system

$$T(s) = \frac{Y(s)}{R(s)} = \frac{2s^2 + 8s + 6}{s^3 + 8s^2 + 16s + 6}. \tag{2.2}$$

We can obtain a state-space representation using the CD Convert to Sate-Space Model function, as shown in Fig. 2.1. The state-space representation of Eq. (2.2) is given by Eq. (2.1) where

$$\mathbf{A} = \begin{bmatrix} -8 & -4 & -1.5 \\ 4 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0.25 \\ 0 \\ 0 \end{bmatrix},$$

and

$$\mathbf{C} = [8 \quad 8 \quad 6], \qquad \mathbf{D} = [0].$$

The state-space representation of the transfer function in Eq. (2.2) is depicted in block diagram form in Fig. 2.2. ◇

The time response of the system in Eq. (2.1) is given by the solution to the vector differential equation

$$\mathbf{x}(t) = \exp(\mathbf{A}t)\mathbf{x}(0) + \int_0^t \exp\left[\mathbf{A}(t-\tau)\right]\mathbf{B}u(\tau)d\tau. \tag{2.3}$$

The matrix exponential function in Eq. (2.3) is the state transition matrix, $\Phi(t)$, where

$$\Phi(t) = \exp(\mathbf{A}t).$$

We can use the function Matrix Exp to compute the transition matrix for a given time interval, as illustrated in Fig. 2.3. An RLC network is investigated in Example 2.2.

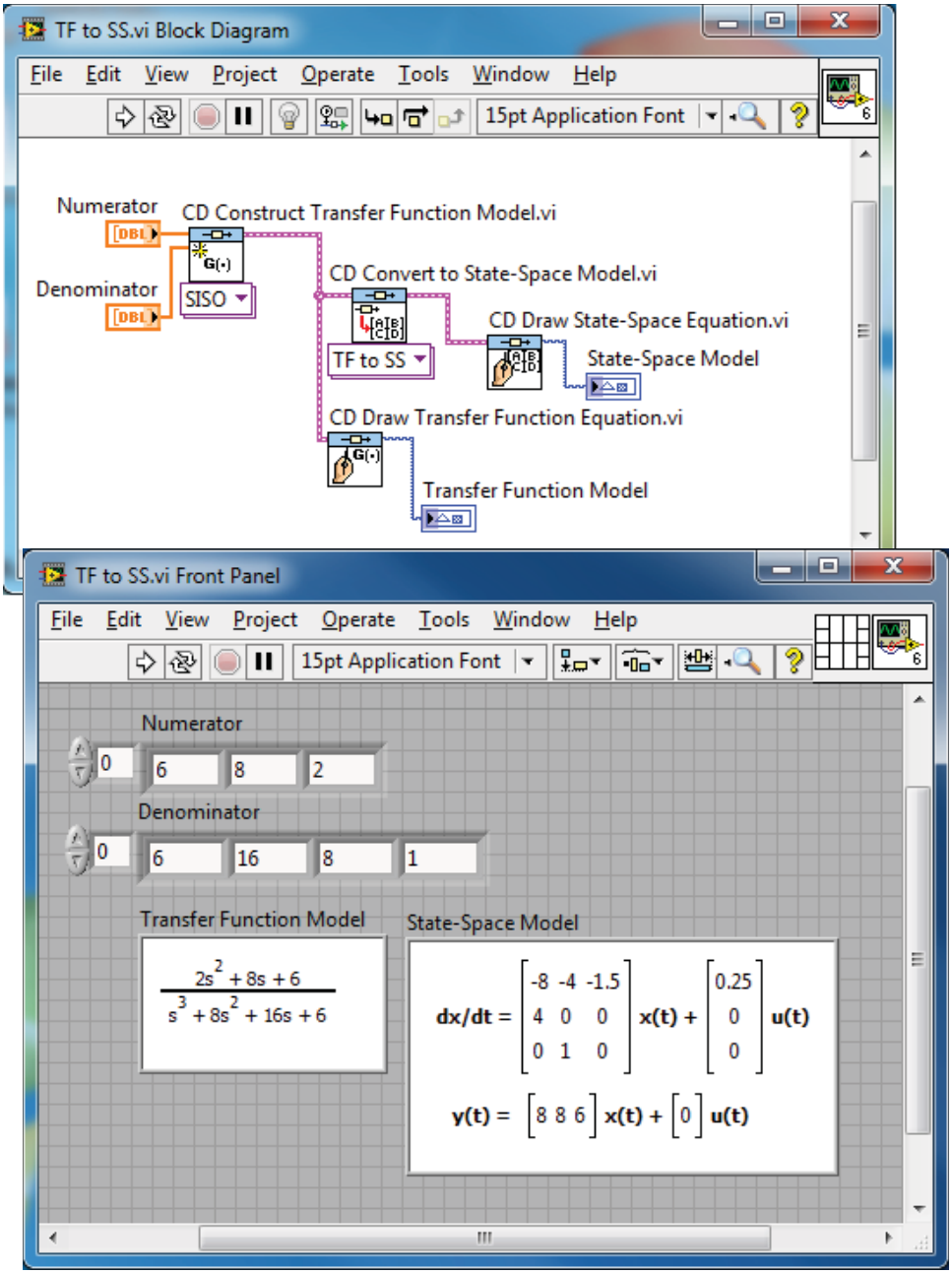


Figure 2.1: Conversion of transfer functions to a state-space representation.

Example 2.2 State Transition Matrix of a RLC Network

The state of the system can be described in terms of the state variables (x_1, x_2), where $x_1(t)$ is the capacitor voltage, $v_c(t)$, and $x_2(t)$ is the inductor current, $i_L(t)$. The state-space representation of the RLC circuit is

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & -\frac{1}{C} \\ \frac{1}{L} & -\frac{R}{L} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \frac{1}{C} \\ 0 \end{bmatrix} u \qquad \text{and} \qquad y = [0 \quad R]\,\mathbf{x} + [0]\,u.$$

where $L =$ inductance (H), $R =$ resistance (Ω), and $C =$ capacitance (F). When $R = 3\ \Omega$, $L = 1$ H, and $C = 1/2$ F, we have

$$\mathbf{A} = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \qquad \mathbf{C} = [0 \quad 3], \qquad \mathbf{D} = [0].$$

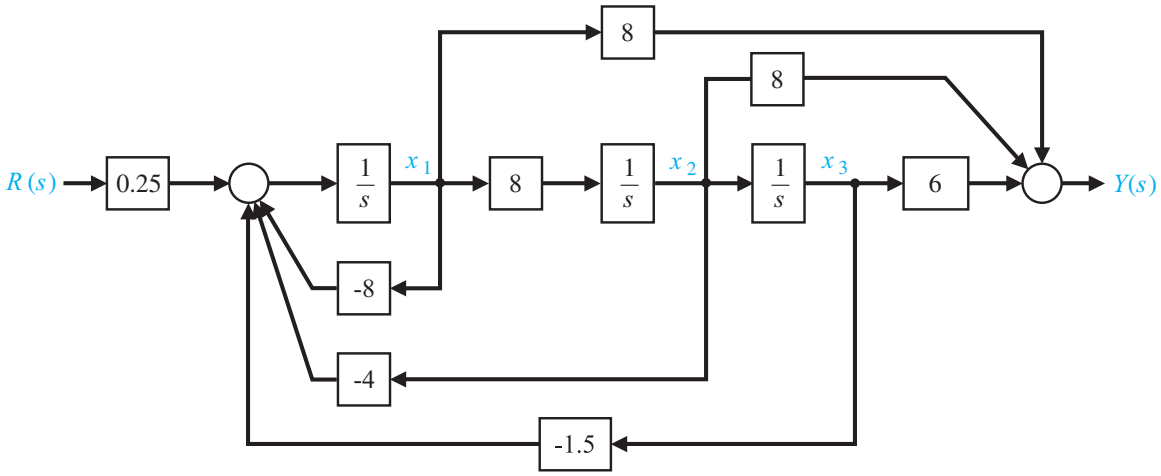


Figure 2.2: Block diagram with x_1 defined as the leftmost state variable.

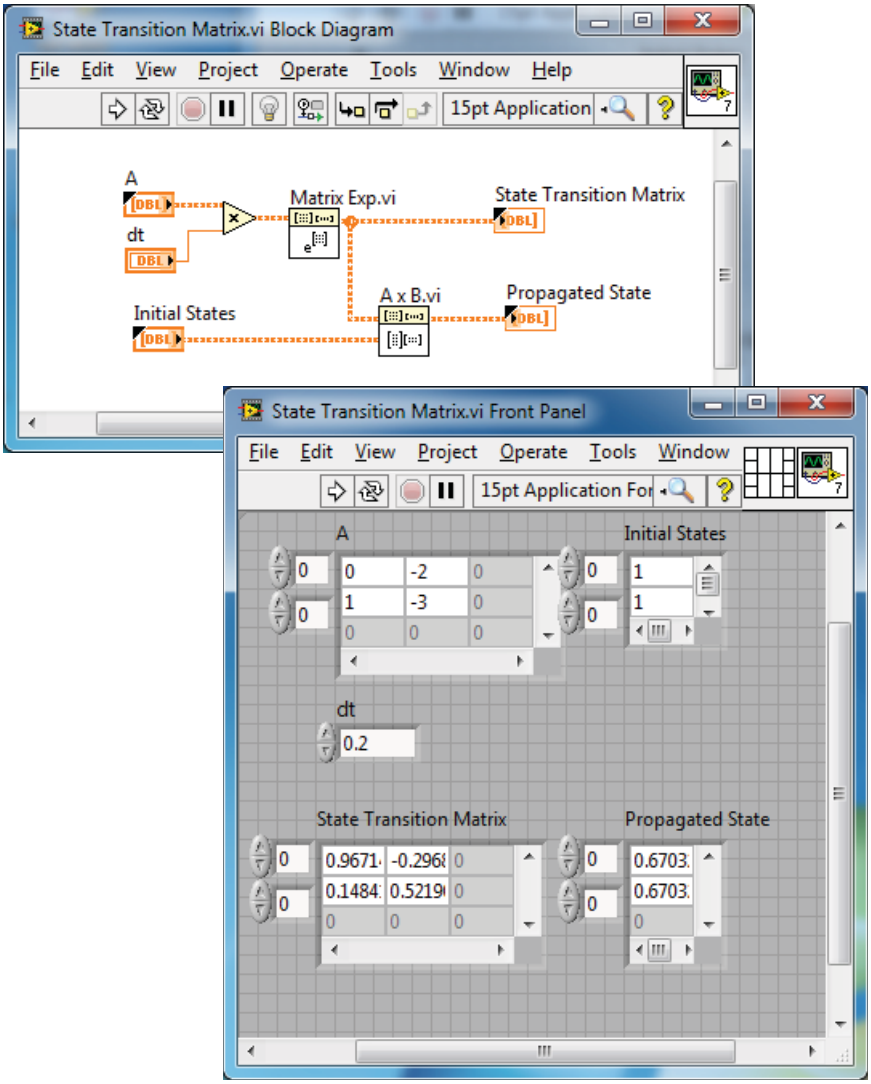


Figure 2.3: Computing the state transition matrix for a given time, $\Delta t = dt$.

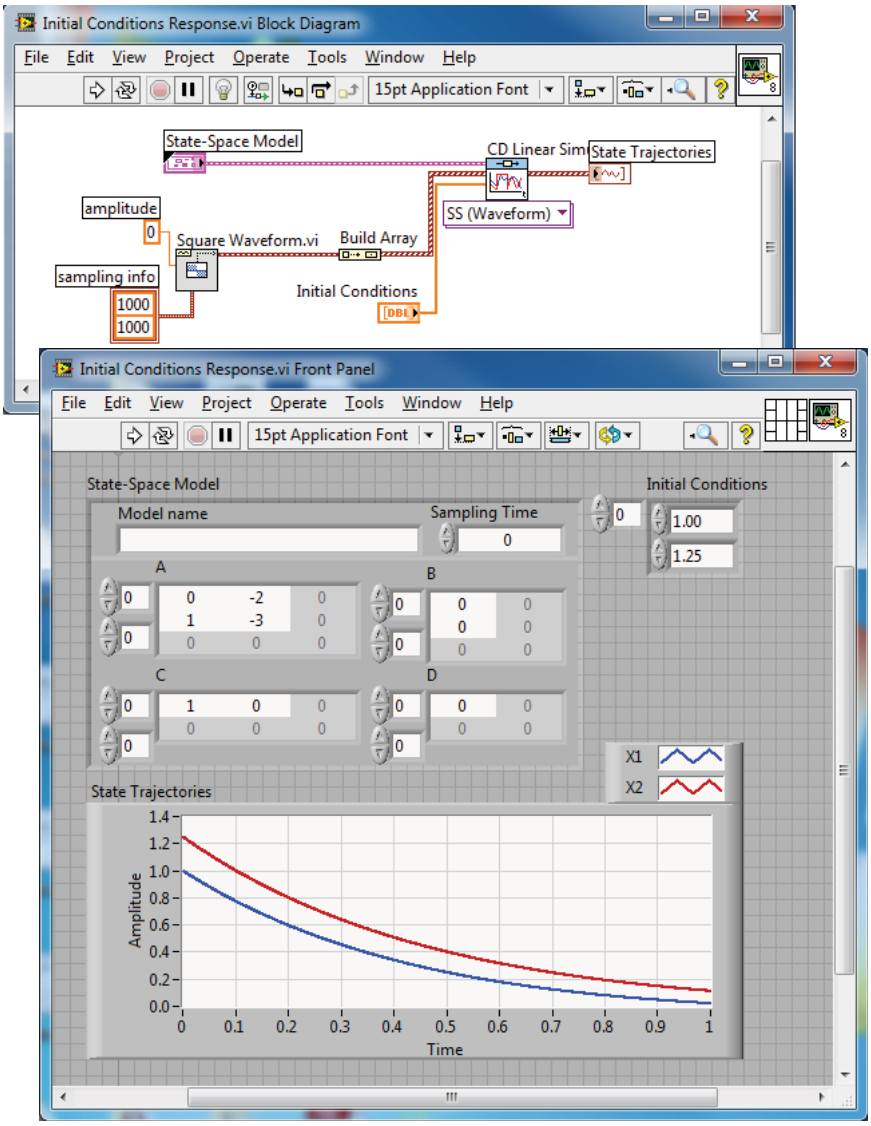


Figure 2.4: Computing the time response for nonzero initial conditions and zero input using CD Linear Simulation.

The initial conditions are $x_1(0) = 1$, $x_2(0) = 1.25$, and the input $u(t) = 0$. At $t = 0.2$, using the state transition matrix, the state is

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{t=0.2} = \begin{bmatrix} 0.9671 & -0.2968 \\ 0.1484 & 0.5219 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{t=0} = \begin{pmatrix} 0.6 \\ 0.8 \end{pmatrix}.$$

The time response of the system of Eq. 2.2 can also be obtained using the CD Linear Simulation function. The CD Linear Simulation function can accept nonzero initial conditions, as well as an arbitrary input function. Using the CD Linear Simulation function, we can calculate the response for the RLC network as shown in Fig. 2.4. The state at $t = 0.2$ is predicted with the CD Linear Simulation function to be $x_1(0.2) = 0.6$ and $x_2(0.2) = 0.8$. We can compare the results obtained by the CD Linear Simulation function by multiplying the initial condition state vector by the state transition matrix to find the same result. In the example illustrated in Fig. 2.4, we set $\mathbf{B} = \mathbf{0}$ to obtain the zero-input response. This allows us to keep the Square Waveform function input in the code for future analysis by setting $\mathbf{B}$ back to its non-zero values. ◇

CHAPTER 3

Feedback Control System Characteristics

The advantages of feedback will be illustrated with two examples in this chapter using LabVIEW in the control system analysis. In the first example, we introduce feedback control to a speed tachometer system in an effort to reject disturbances. The reduction in system sensitivity to plant variations, adjustment of the transient response, and reduction in steady-state error will be demonstrated in a second example, the English Channel boring machine.

Example 3.1 Speed Control System

The open-loop block diagram description of the armature-controlled dc motor with a load torque disturbance, $T_d(s)$, is shown in Fig. 3.1. The values for the various parameters are given in Table 3.1. We have two inputs to our system, $V_a(s)$ and $T_d(s)$. Relying on the **principle of superposition**, which applies to our linear system, we consider each input separately. To investigate the effects of disturbances on the system, we let $V_a(s) = 0$ and consider only the disturbance $T_d(s)$. Conversely, to investigate the response of the system to a reference input, we let $T_d(s) = 0$ and consider only the input $V_a(s)$. The closed-loop speed tachometer control system block diagram is shown in Fig. 3.2.

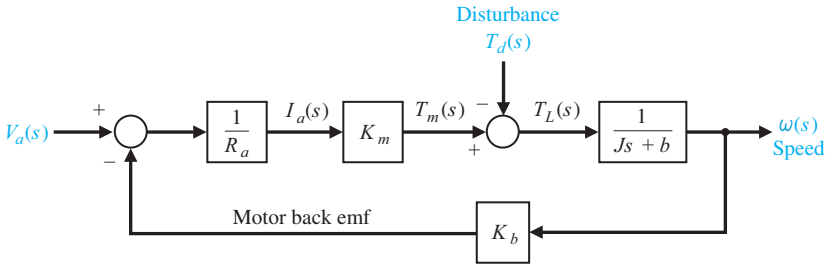


Figure 3.1: Open-loop speed control system (without external feedback).

Table 3.1: Tachometer Control System Parameters

R_a	K_m	J	b	K_b	K_a	K_t
1	10	2	0.5	0.1	54	1

If the system displays good disturbance rejection, then we expect the disturbance $T_d(s)$ to have a small effect on the output $\omega(s)$. Consider the open-loop system in Fig. 3.3 first. We can use LabVIEW to compute the transfer function from $T_d(s)$ to $\omega(s)$ and evaluate the output response to a unit step disturbance (that is, $T_d(s) = 1/s$). The time response to a unit step disturbance is shown in Fig. 3.3.

The open-loop transfer function is

$$G(s) = \frac{\omega(s)}{T_d(s)} = \frac{-1}{2s + 1.5}.$$

Since the desired value of $\omega(t)$ is zero (remember that $V_a(s) = 0$), the steady-state error is just the final value of $\omega(t)$, which is denoted by $\omega_o(t)$. The steady-state error, shown on the plot in Fig. 3.3, is approximately the value of the speed when $t = 7$ seconds. We can obtain an approximate value of the steady-state error by looking at the plot in Fig. 3.3. The approximate steady-state value of $\omega_o(t)$ is

$$\omega_o(\infty) \approx \omega_o(7) = -0.66 \text{ rad/s}.$$

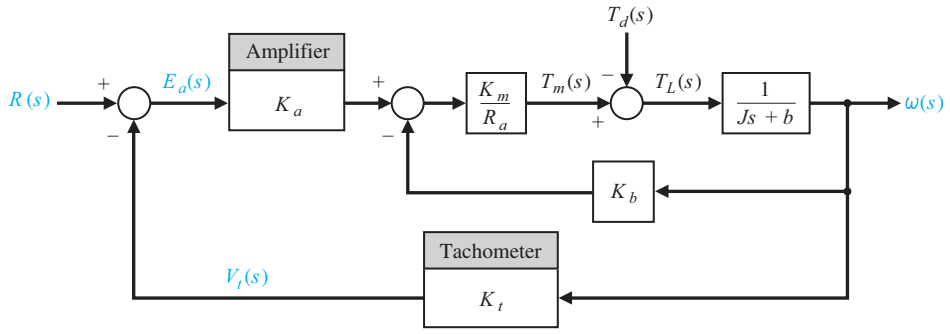


Figure 3.2: Closed-loop speed tachometer control system.

In a similar fashion, we begin the closed-loop system analysis by computing the closed-loop transfer function from $T_d(s)$ to $\omega(s)$ and then generating the time-response of $\omega(t)$ to a unit step disturbance input. The output response and the VI are shown in Fig. 3.4.

The closed-loop transfer function from the disturbance input is

$$\frac{\omega(s)}{T_d(s)} = \frac{-1}{2s + 541.5}.$$

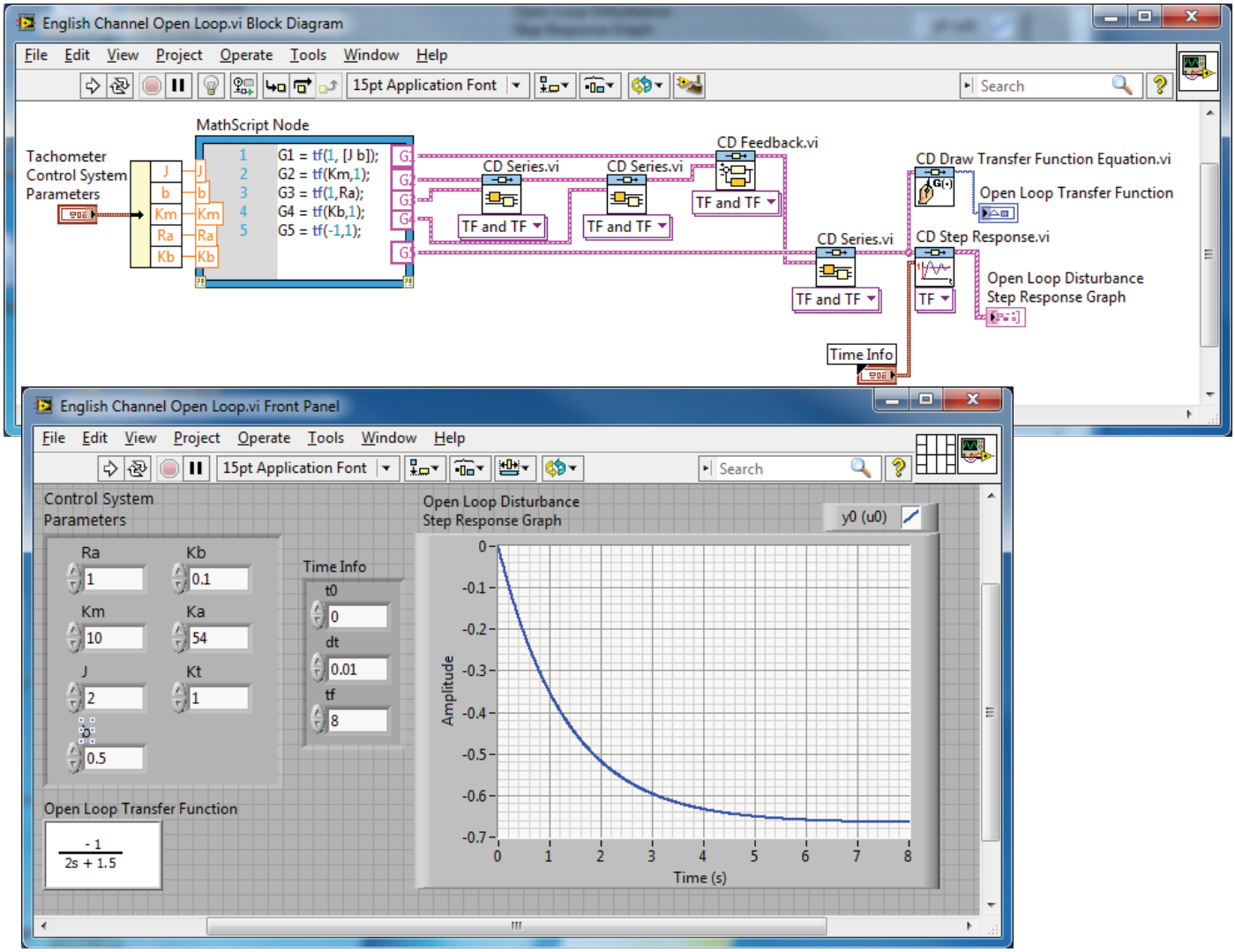


Figure 3.3: Analysis of the open-loop speed control system.

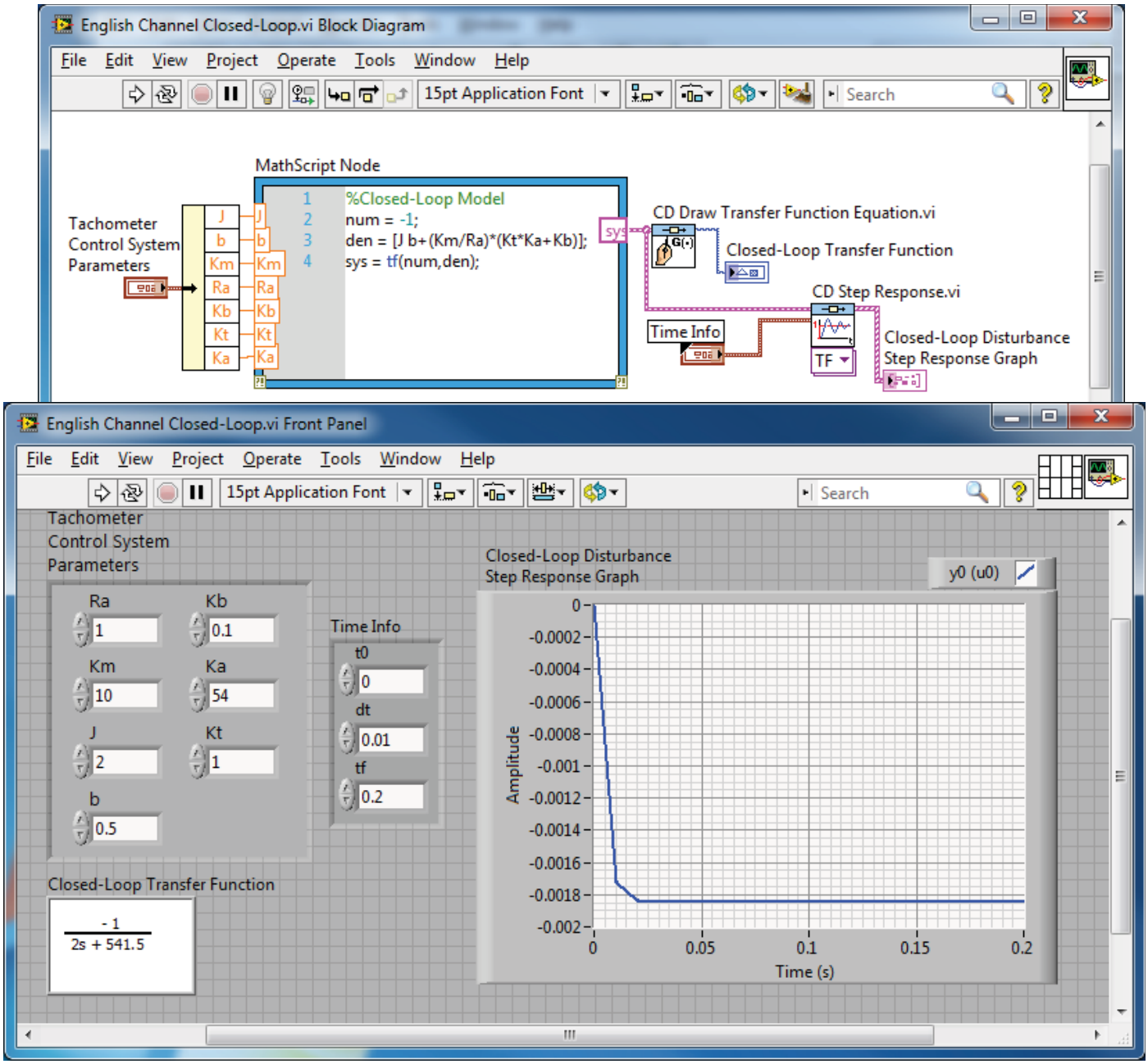


Figure 3.4: Analysis of the closed-loop speed control system.

As before, the steady-state error is just the final value of $\omega(t)$, which is denoted by $\omega_c(t)$. The steady-state error is shown on the plot in Fig. 3.4. We can obtain an approximate value of the steady-state error by looking at the plot in Fig. 3.4. The approximate steady-state value of $\omega(t)$ is

$$\omega_c(\infty) \approx \omega_o(0.02) = -0.002 \text{ rad/s}.$$

We generally expect that $|\omega_c(\infty)/\omega_o(\infty)| < 0.02$. The ratio of closed-loop to the open-loop steady-state speed output due to a unit step disturbance input, in this example, is

$$\frac{\omega_c(\infty)}{\omega_o(\infty)} = 0.003.$$

We have achieved a remarkable improvement in disturbance rejection. It is clear that the addition of the negative feedback loop reduced the effect of the disturbance on the output.This demonstrates the **disturbance rejection** property of closed-loop feedback systems. ◇

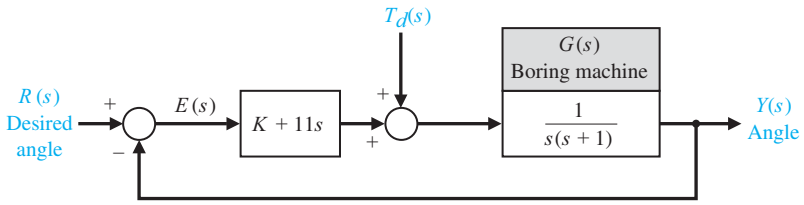


Figure 3.5: A block diagram model of a boring machine control system.

Example 3.2 English Channel Boring Machines

The block diagram description of the English Channel boring machine is shown in Fig. 3.5.

The transfer function of the output due to the two inputs is

$$Y(s) = \frac{K + 11s}{s^2 + 12s + K} R(s) + \frac{1}{s^2 + 12s + K} T_d(s).$$

The effect of the control gain K on the transient response is shown in Fig. 3.6 along with the VI used to generate the plots.

Comparing the two plots, it can be seen that decreasing K decreases the **percent overshoot**. Although it is not as obvious from the plots in Fig. 3.6, it is also true that decreasing K increases the **settling time**. This can be verified by taking a closer look at the data used to generate the plots. This example demonstrates how the transient response can be altered by feedback control gain

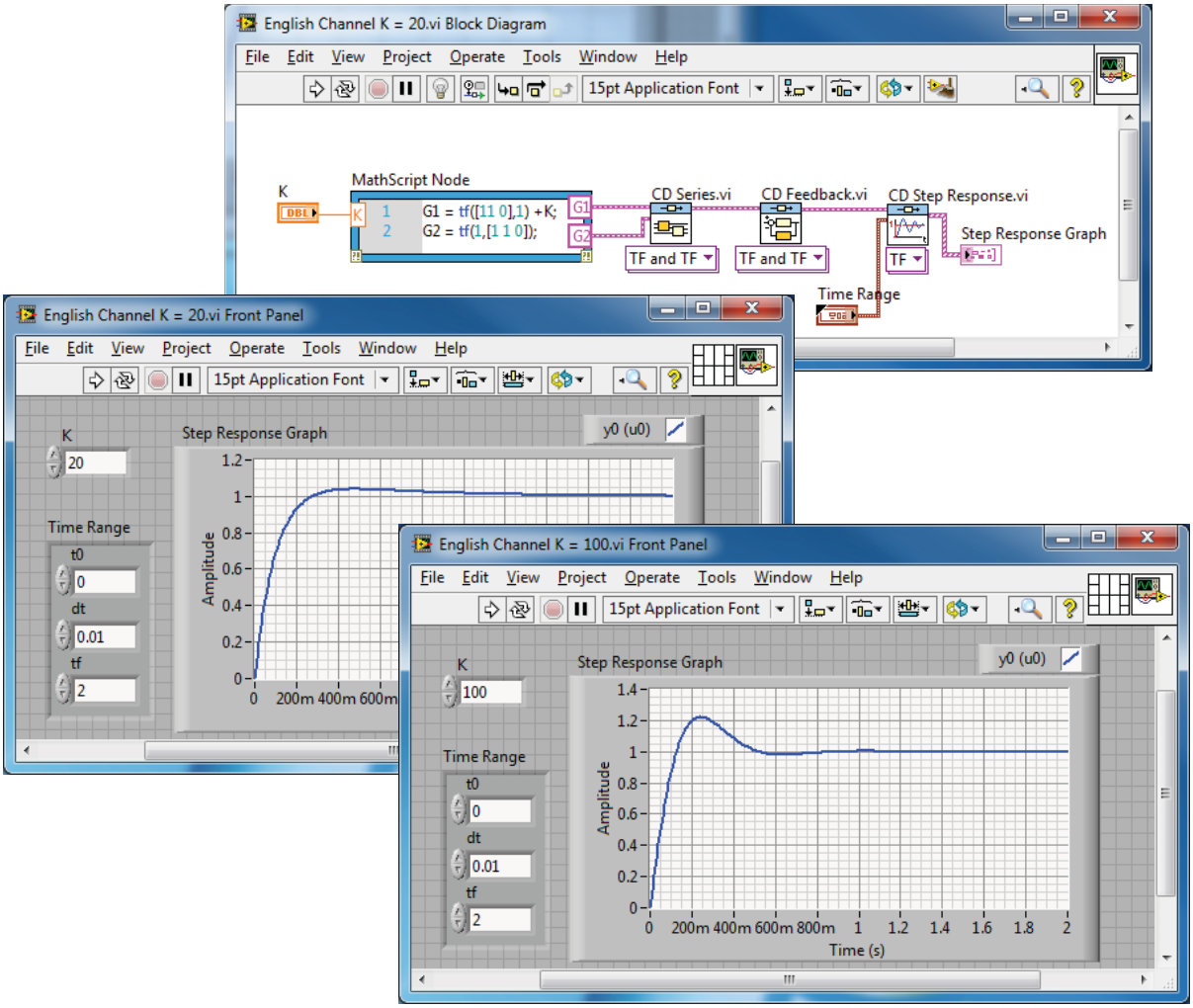


Figure 3.6: The response to a step input with $K = 100$ and $K = 20$ and LabVIEW script.

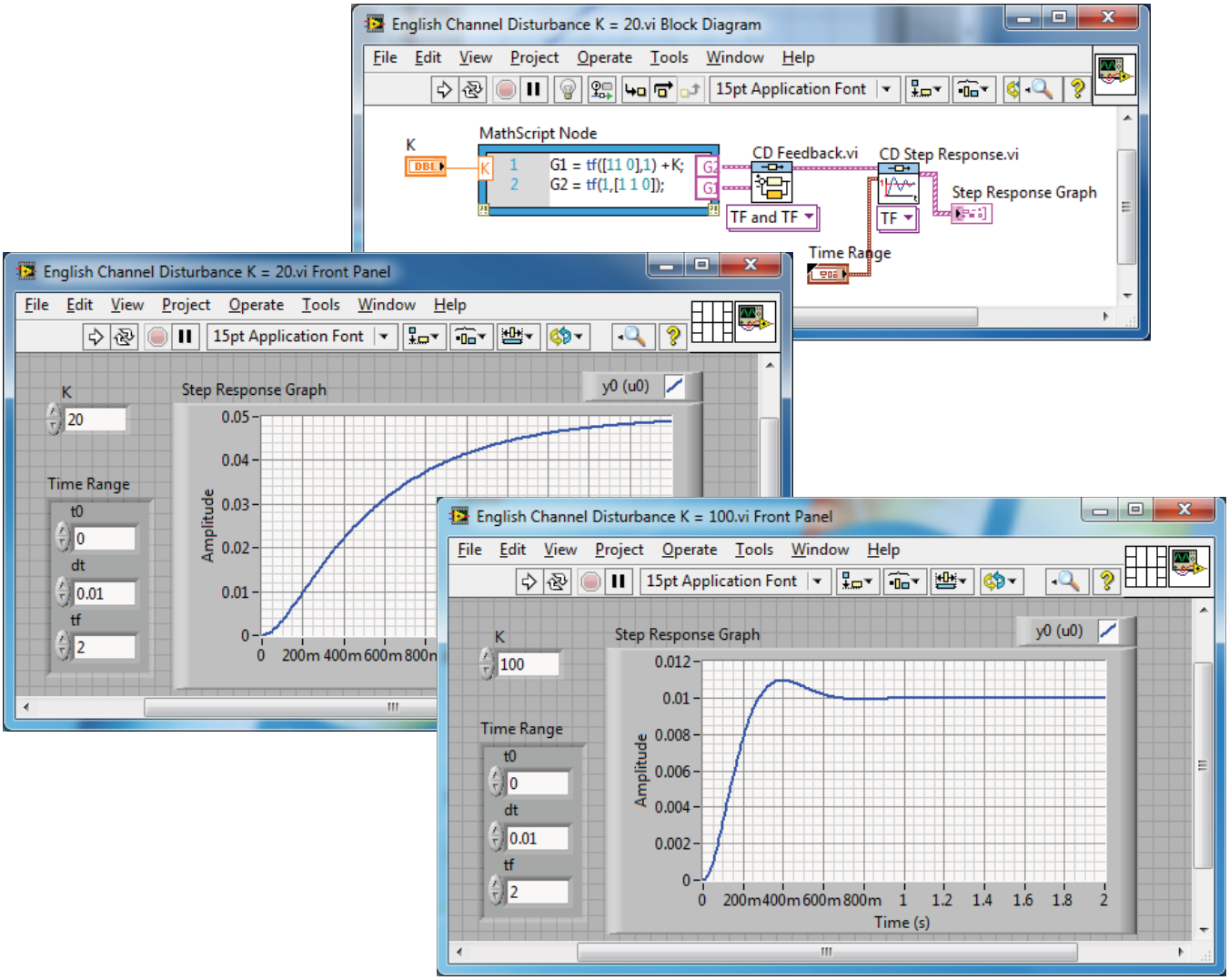


Figure 3.7: The response to a step disturbance with $K = 100$ and $K = 20$.

K. Based on our analysis thus far, we would prefer to use $K = 20$. Other considerations must be taken into account before we can establish the final design. Before making the final choice of K, it is important to consider the system response to a unit step disturbance, as shown in Fig. 3.7.

We see that increasing K reduces the steady-state response of $y(t)$ to the step disturbance. The steady-state value of $y(t)$ is 0.05 and 0.01 for $K = 20$ and 100, respectively. The steady-state errors, percent overshoot, and settling times (2% criteria) are summarized in Table 3.2.

Table 3.2: Response of the Boring Machine Control System for $K = 20$ and $K = 100$

	$K = 20$	$K = 100$
P.O.	4%	22%
T_s	1.0s	0.7s
e_{ss}	5%	1%

The steady-state values are predicted from the final-value theorem for a disturbance input as follows:

$$\lim_{t\to\infty} y(t) = \lim_{s\to 0} s\left[\frac{1}{s(s+12)+K}\right]\frac{1}{s} = \frac{1}{K}.$$

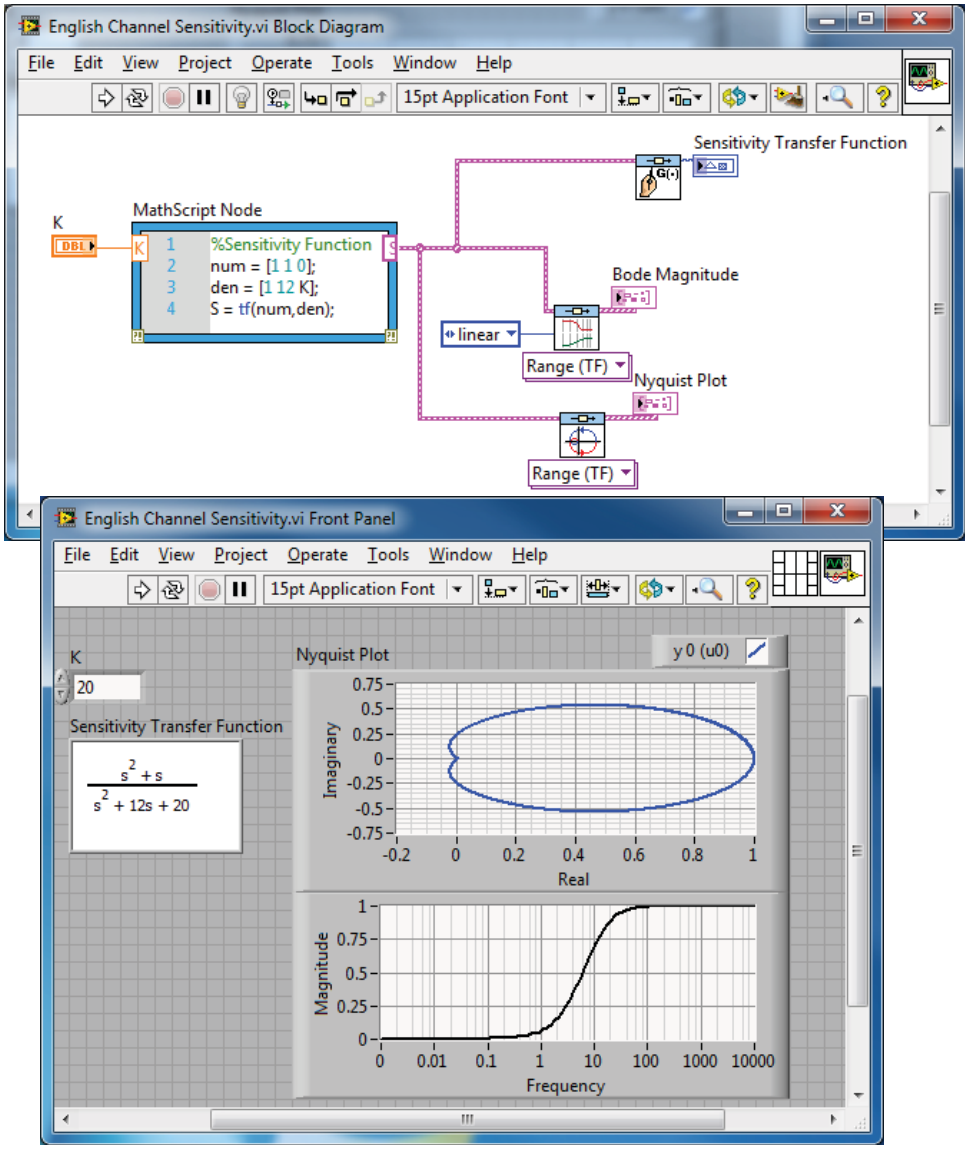


Figure 3.8: System sensitivity to plant variations ($s = j\omega$).

If our only design consideration is disturbance rejection, we would prefer to use $K = 100$. We have just experienced a common trade-off situation in control system design. In this particular example increasing K leads to better disturbance rejection, whereas decreasing K leads to better performance (that is, less overshoot). The final decision on how to choose K rests with the designer. Although LabVIEW can certainly assist in the control system design, it cannot replace the engineer's decision-making capability and intuition. The final step in the analysis is to look at the system sensitivity to changes in the plant. The sensitivity function is computed as

$$S(s) = \frac{s(s+1)}{s(s+12)+K}.$$

We can compute the values of $S(s)$ for different values of s and generate a plot of the system sensitivity. For low frequencies, we can approximate the system sensitivity by

$$S(s) \approx \frac{s}{K}.$$

Increasing the gain K reduces the system sensitivity. The system sensitivity plots when $s = j\omega$ are shown in Fig. 3.8 for $K = 20$. ◇

CHAPTER 4

Performance of Feedback Control Systems

In this chapter we begin our investigation of the performance of feedback control systems. As an illustrative example, we solve the problem of bank angle control of an aircraft using LabVIEW to build a reasonably simple yet effective model of the response to aileron deflections. Then using the idea of model simplification, we present a second-order approximate model. The second-order approximation gives us insight into the expected behavior of the system and how to obtain an initial design. Finally we use a LabVIEW simulation to perform analysis in an interactive mode. We begin by investigating time-domain performance specifications given in terms of transient response to a given input signal and the resulting steady-state tracking errors.

4.1 Time-Domain Specifications

Time-domain performance specifications are generally given in terms of the transient response of a system to a given input signal. Because the actual input signals are generally unknown, a standard test input signal is used. Consider the second-order system shown in Fig. 4.1.

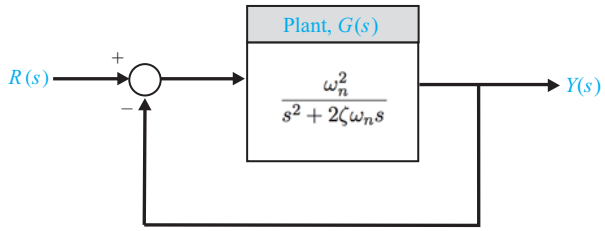


Figure 4.1: Single-loop second-order feedback system.

The closed-loop output is

$$Y(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} R(s).$$

We have already discussed the use of the step function to compute the step response of a system. Now we address another important test signal: the impulse function. The impulse response is the time derivative of the step response. The response of a second-order system to a step function is shown in Fig. 4.2 for various values of ω_n and ζ.

Similarly, the response of a second-order system to an impulse function is shown in Fig. 4.3. In the script, we set $\omega_n = 1$, which is equivalent to computing the step response versus $\omega_n t$. This gives us a more general plot valid for any $\omega_n > 0$.

We studied the CD Linear Simulation function in Chapter 2 for use with state-variable models; and now we consider its use with transfer function models as shown in Example 4.1.

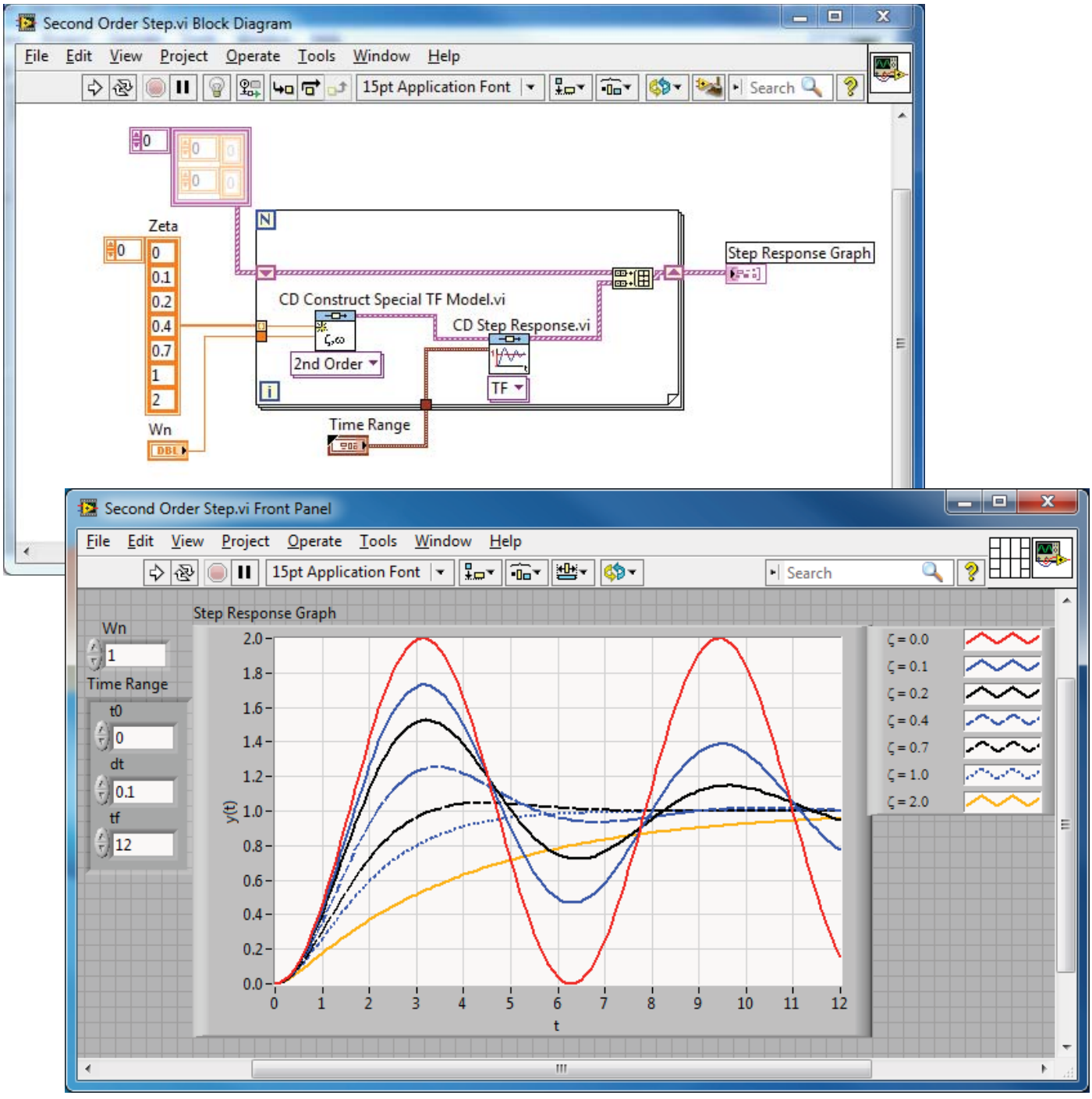


Figure 4.2: Response of a second-order system to a step function.

Example 4.1 Mobile Robot Steering Control

The block diagram for a steering control system for a mobile robot is shown in Fig. 4.4. Suppose the steering controller, $G_c(s)$, is

$$G_c(s) = K_P + \frac{K_I}{s}.$$

When the input is a ramp, the steady-state error is

$$e_{ss} = \frac{A}{K_v}, \tag{4.1}$$

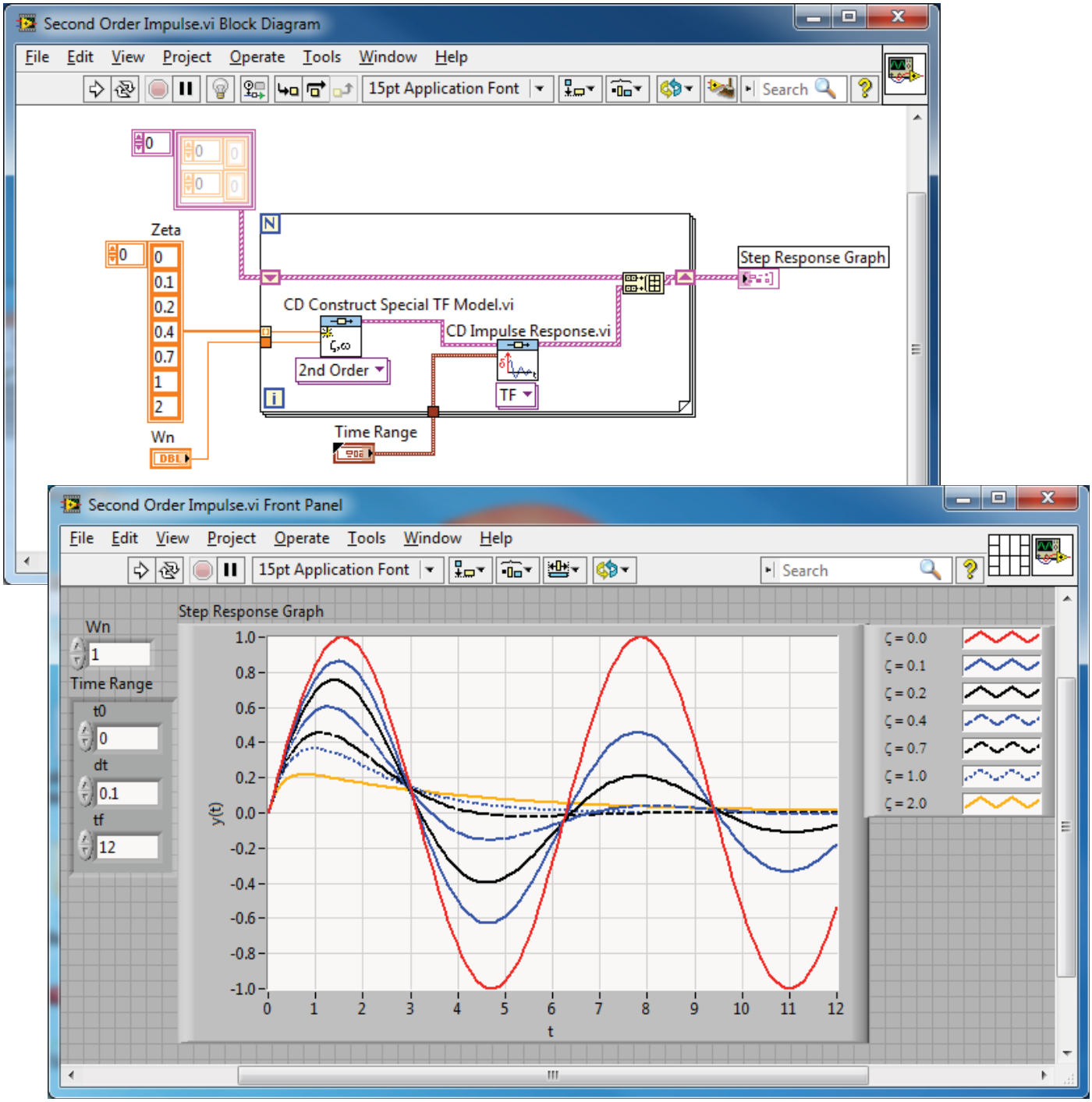


Figure 4.3: Response of a second-order system to an impulse function.

where $K_v = K_I K$. The effect of the controller constant, K_I, on the steady-state error is evident from Eq. (4.1). Whenever K_I is large, the steady-state error is small. We can simulate the closed-loop system response to a ramp input using the CD Linear Simulation function. The controller gains K_P, K_I, and the system gain K can be represented symbolically in the block diagram so that various values can be selected and simulated. The results are shown in Fig. 4.5 for $K_P = K = 1$, $K_I = 2$, and $\tau = 1/10$. ◇

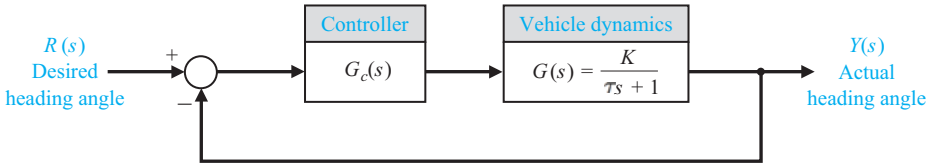


Figure 4.4: Steering control system block diagram.

4.2 Simplification of Linear Systems

It may be possible to develop a lower-order approximate model that closely matches the input-output response of a high-order model. We can use LabVIEW to compare the approximate model to the actual model, as illustrated in Example 4.2.

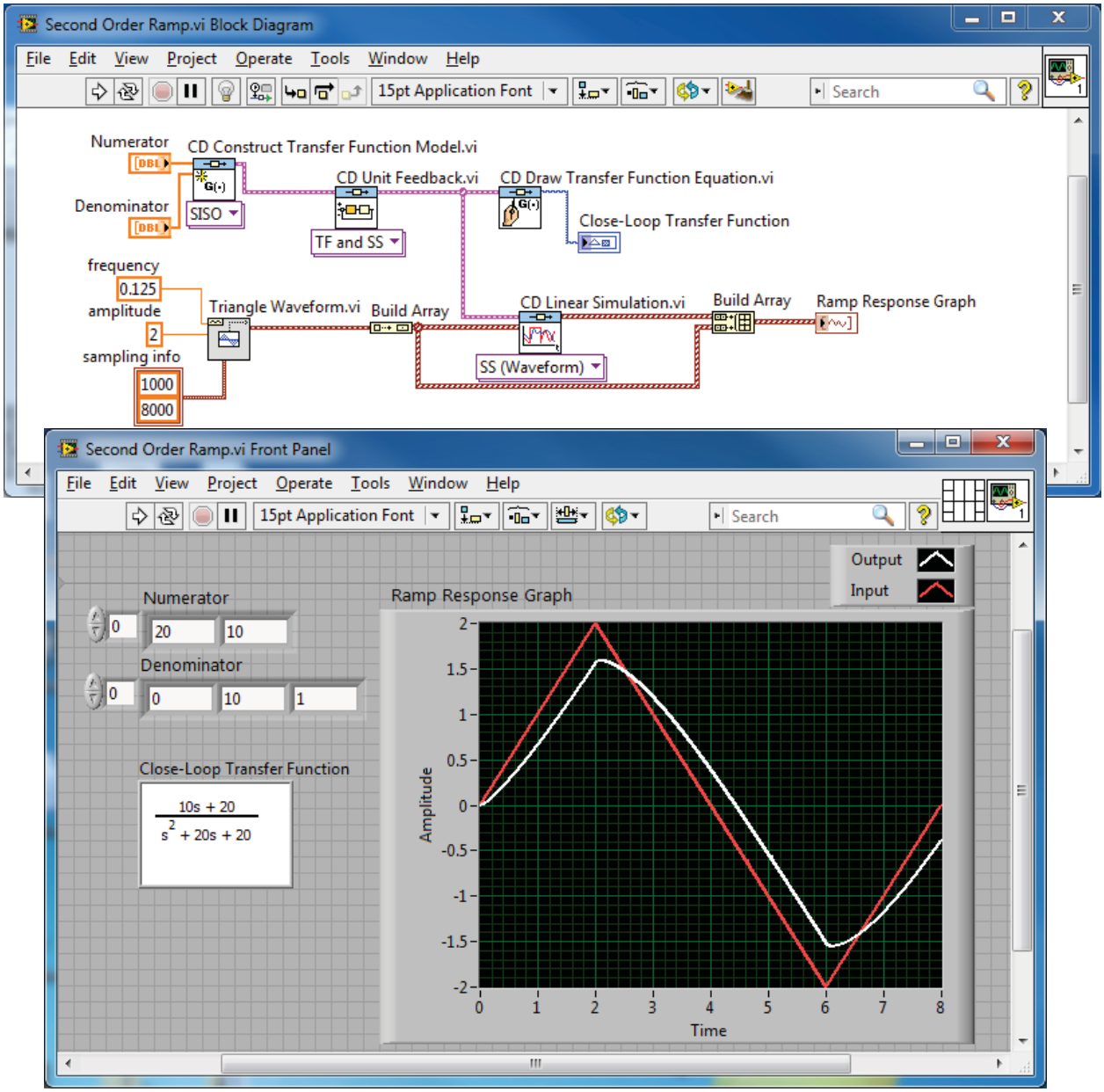


Figure 4.5: Transient response of the mobile robot steering control system to a ramp input.

Example 4.2 Simplified Model

Consider the third-order system

$$H(s) = \frac{6}{s^3 + 6s^2 + 11s + 6}.$$

A second-order approximation is

$$L(s) = \frac{1.60}{s^2 + 2.584s + 1.60}.$$

A comparison of their respective step responses is given in Fig. 4.6. ◇

Example 4.3 Aircraft Roll Control using LabVIEW

Each time we fly on a commercial airliner, we experience first-hand the benefits of automatic control systems. These systems assist pilots by providing pilot relief during extended flights and by improving the handling qualities of the aircraft over a wide range of flight conditions. The special relationship between flight and controls began in the early work of the Wright brothers. Using wind

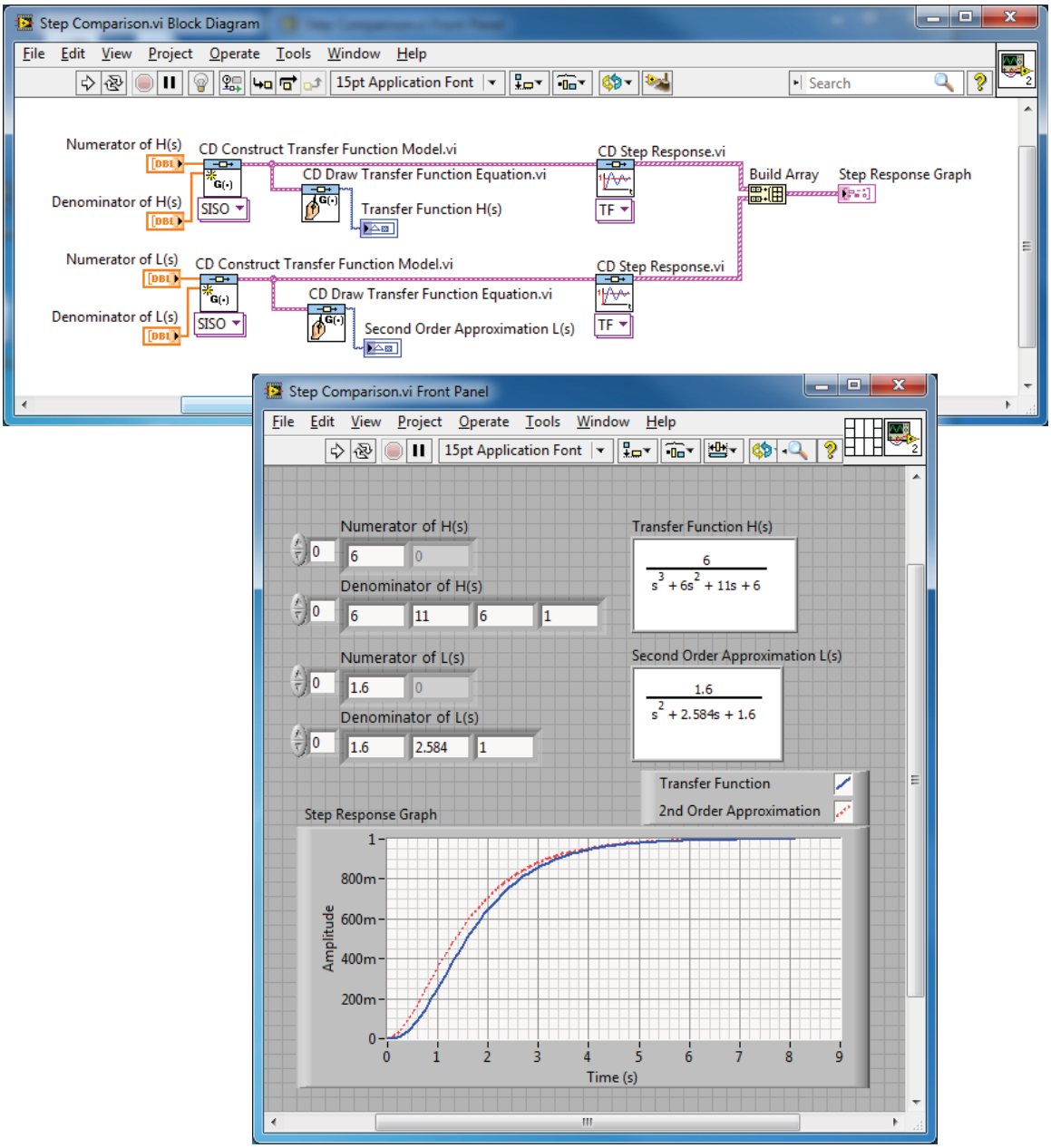


Figure 4.6: Step response comparison for an approximate transfer function versus the actual transfer function.

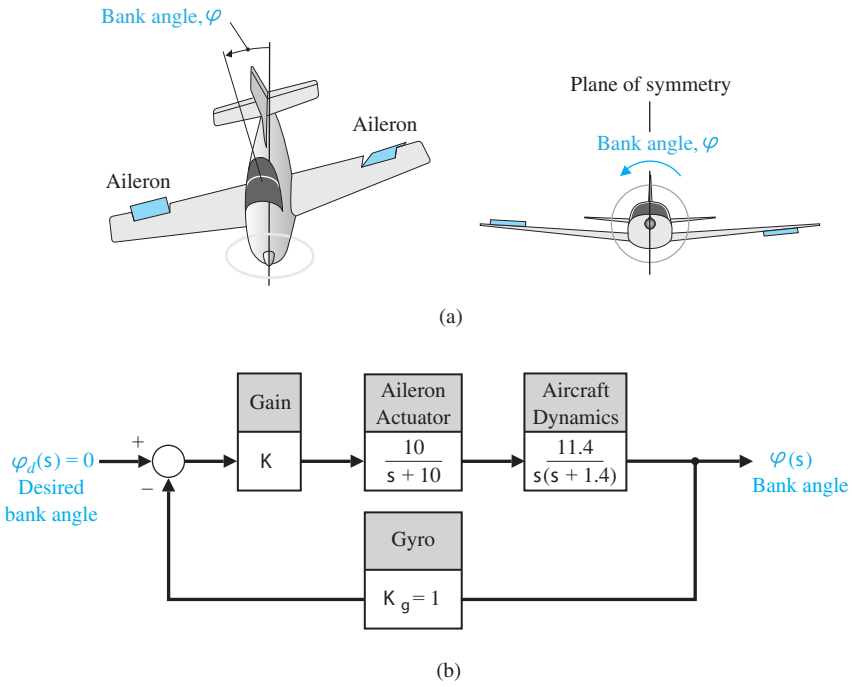


Figure 4.7: (a) Control of the bank angle of an airplane using differential deflections of the ailerons. (b) Bank angle control autopilot.

tunnels the Wright brothers applied systematic design techniques to make their dream of powered flight a reality. This systematic approach to design was partially responsible for their success. Another significant aspect of their approach was their emphasis on flight controls; the brothers insisted that their aircraft be pilot controlled. Observing birds control their rolling motion by twisting their wings, the Wright brothers built aircraft with mechanical mechanisms that twisted their airplane wings. Today we no longer use wing warping as a mechanism for performing a roll maneuver, instead we control rolling motion using ailerons, as shown in Fig. 4.7. The Wright brothers also used elevators (located forward) for longitudinal control (pitch motion) and rudders for lateral control (yaw motion). Today's aircraft still use both elevators and rudders, although the elevators are generally located on the tail (rearward).

The first controlled, powered, unassisted take-off flight occurred in 1903 with the Wright Flyer I (a.k.a. Kitty Hawk). The first practical airplane, the Flyer III could fly figure eights and stay aloft for half an hour. Three-axis flight control was a major (and often overlooked) contribution of the Wright brothers. A concise historical perspective is presented in Stevens and Lewis[1].

For our aircraft we select $e_o = 1.4$ and $k = 11.4$, where the aircraft dynamics are given by $G(s) = \frac{k}{s(s+e_0)}$. The associated time-constant of the roll subsidence is $\tau = 1/e_o = 0.7$ second. These values represent a fairly fast rolling motion response typical of an agile aircraft. For the aileron actuator model, we typically use a simple first-order system model,

$$\frac{\delta_a(s)}{e(s)} = \frac{p}{s+p}\,, \tag{4.2}$$

where $e(s) = \varphi_d(s) - \varphi(s)$. In this case we select $p = 10$. This corresponds to a time-constant of $\tau = 1/p = 0.1$ second. This is a typical value consistent with a fast response. We need to have an actuator with a fast response so that the dynamics of the actively controlled airplane will be the dominant component of the system response. A slow actuator is akin to a time-delay that can cause performance and stability problems.

The second-order system approximation has allowed us to gain insight into the relationship between the parameter K and the system response, as measured by percent overshoot and time-to-peak. Of course, the gain $K = 0.16$ is only a starting point in the design. Why? Because we in fact have a third-order system and must consider the effect of the third pole (which we have ignored so far).

We can develop a simulation of the control system to quantify the performance. We can readily vary important system parameters (such as K) and check the resulting performance (as measured by step response characteristics). LabVIEW provides a graphical methodology for developing the simulation using block diagrams. As shown in Fig. 4.8, LabVIEW presents the window of the system in block diagram form.

[1]Aircraft Control and Simulation (2nd Edition) by Brian L. Stevens and Frank L. Lewis, Wiley-Interscience, 2003.

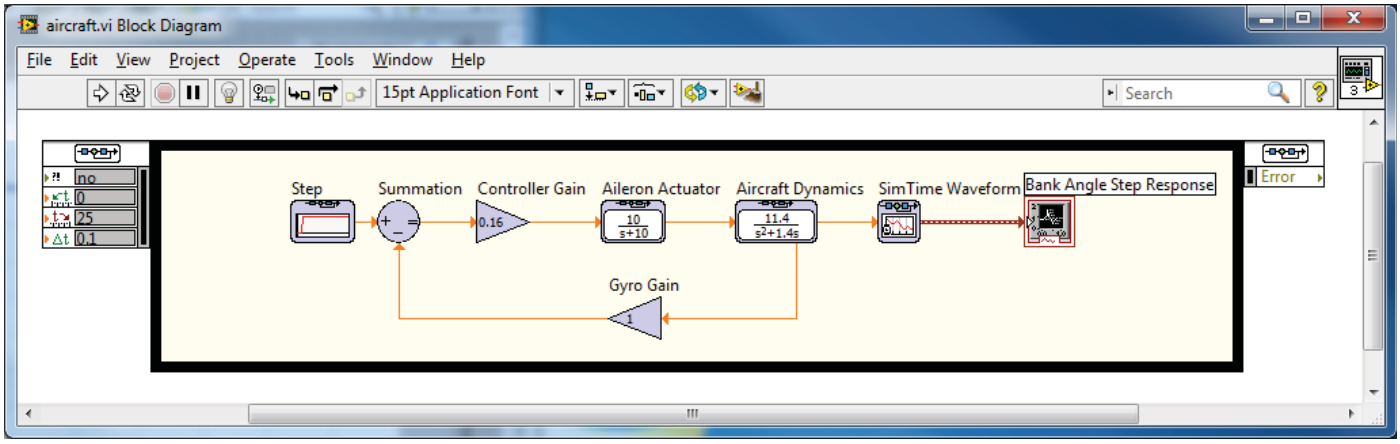


Figure 4.8: LabVIEW simulation window with aircraft, controller, and actuator dynamics.

Before starting the simulation we can opt to change many of the parameters of the system we are modeling. To change any part in the model, simply open (double-click with the mouse button) the Controller Gain, Aileron Actuator, Aircraft Dynamics, or Gyro Gain and enter new parameters.

For example, opening the Controller Gain displays the window shown in Fig. 4.9. We use this window to specify the value of the gain introduced into the system. To move the window and avoid obstructing the diagram, we can click and hold the mouse button

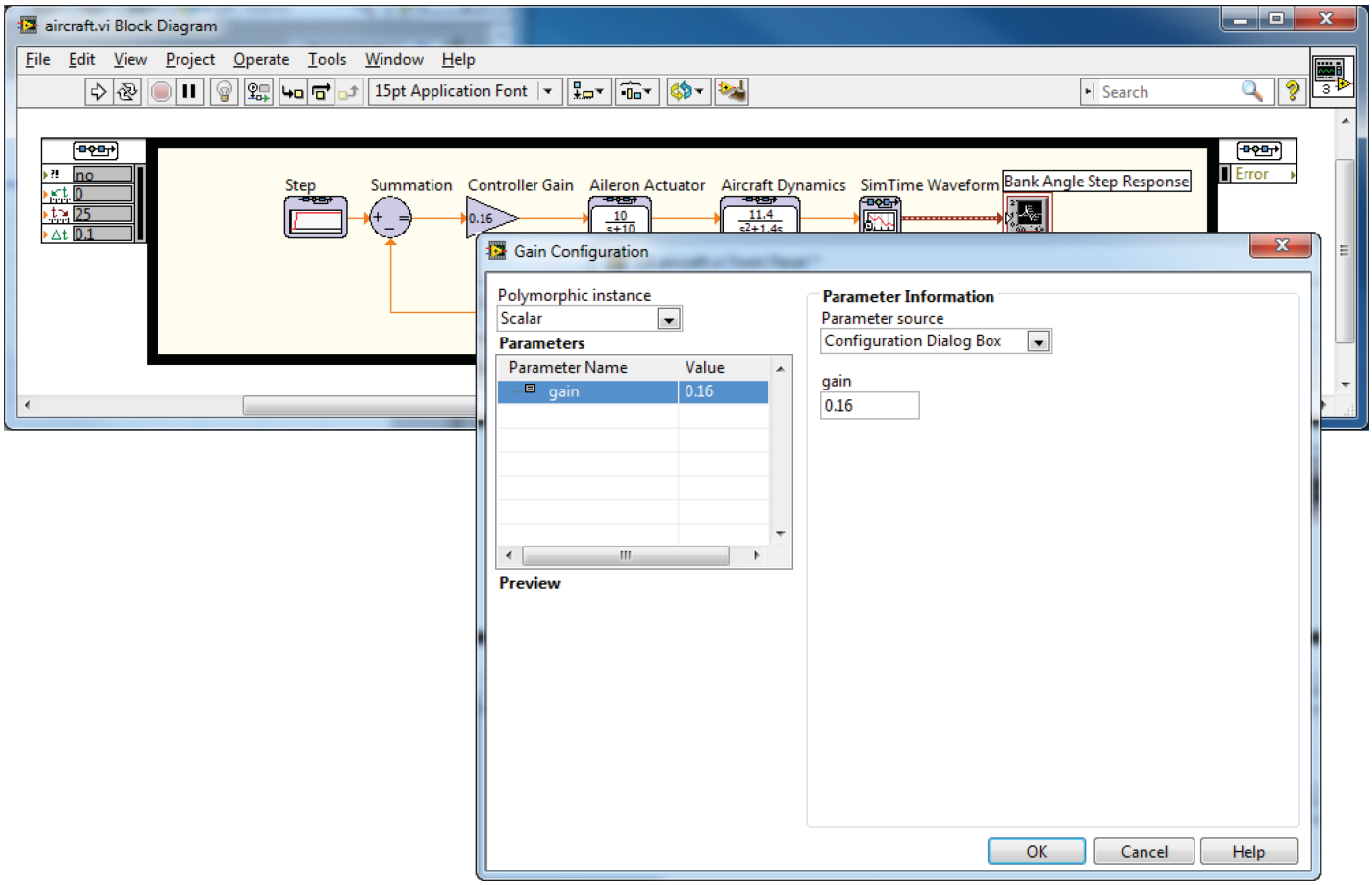


Figure 4.9: Pop-up window used to change the controller gain.

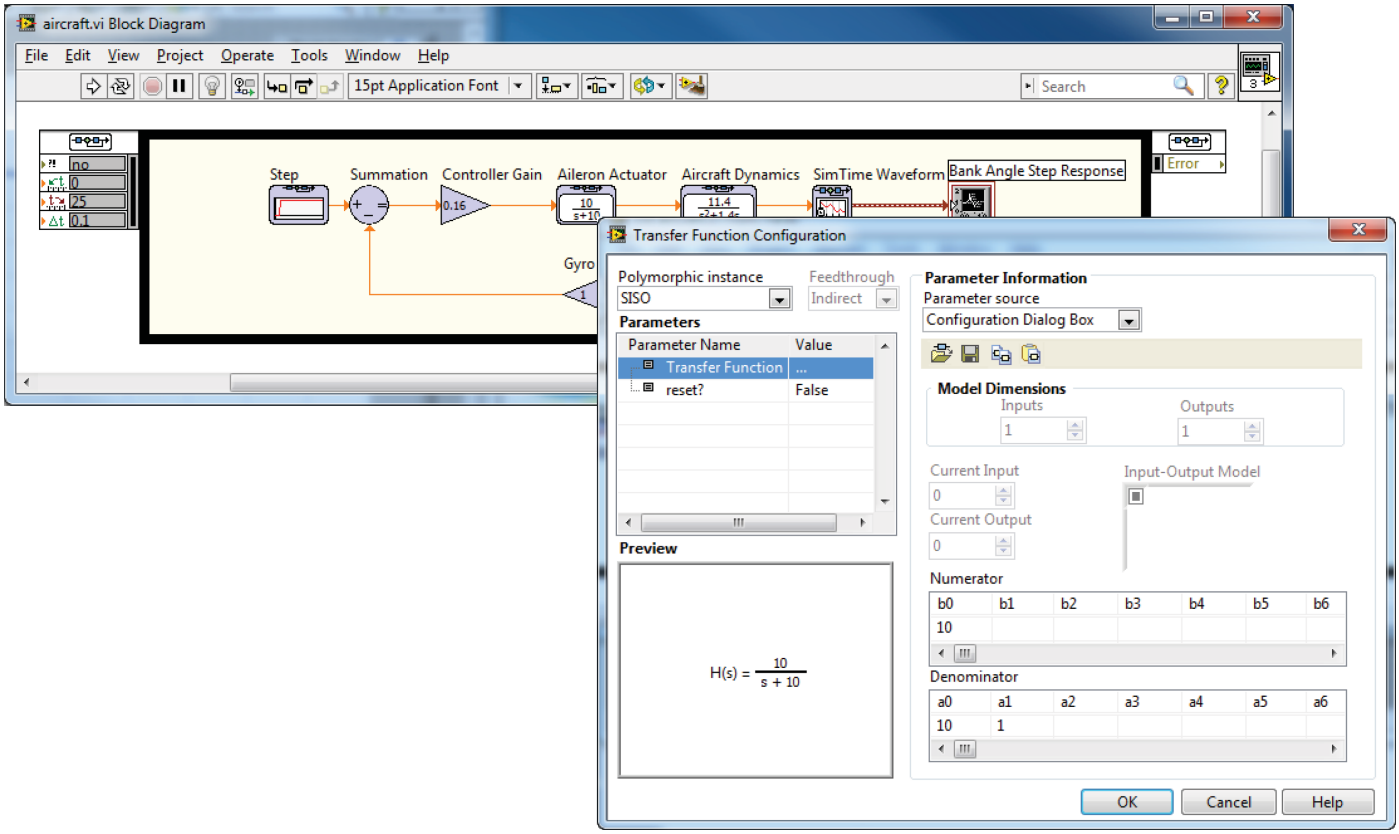


Figure 4.10: Pop-up window used to change the aileron actuator transfer function.

anywhere on the title bar and we drag the window some place else. As we change the parameters they are immediately updated on the diagram. Figures 4.10 and 4.11 show the display windows opened for the aileron actuator and the aircraft dynamics.

We need to set a few more parameters before the model is ready for simulation. Figure 4.12 shows the Configure Simulation Parameters window that allows users to select different numerical integration routines and the parameters for the initial and final time, maximum and minimum step size, and relative and absolute tolerances.

After making the desired changes, select the OK button. During the simulation execution, the time response plot is dynamically updated. The time response should now resemble Fig. 4.13.

To fine-tune the simulation, we can change any or all of the transfer functions or gains by the steps that we just covered. At this point the LabVIEW setup has $K = 0.16$. From our previous analytic analysis, we determined that as K decreased, the percent overshoot also decreased, while the time-to-peak simultaneously increased. With an interactive simulation we can verify this trend, except that now we have the original third-order system in the simulation rather than the approximate second-order system on which the analytic analysis was based. With the second-order system approximation, we estimate that with $K = 0.16$ the percent overshoot, $P.O.$, is 20% and the time-to-peak, T_p, is 2.62 seconds. What is the system performance with the third-order system? LabVIEW provides a way to investigate this easily. The results of the simulation are shown in Fig. 4.13. The percent overshoot is slightly more than 20% due to the presence of a third pole that we ignored in the second-order system analytic analysis. In fact, with $K = 0.16$ we have

$$P.O. = 20.5\% \quad \text{and} \quad T_p = 2.73 \text{ seconds.}$$

The percent overshoot is reduced from 20.5% (with $K = 0.16$) to 9.5% (with $K = 0.1$). Also the time-to-peak increases from 2.73 seconds (with $K = 0.16$) to 3.74 seconds (with $K = 0.1$). We can verify that when $K = 0.2$, the percent overshoot and time-to-peak are 26.5% and 2.38 seconds, respectively. ◇

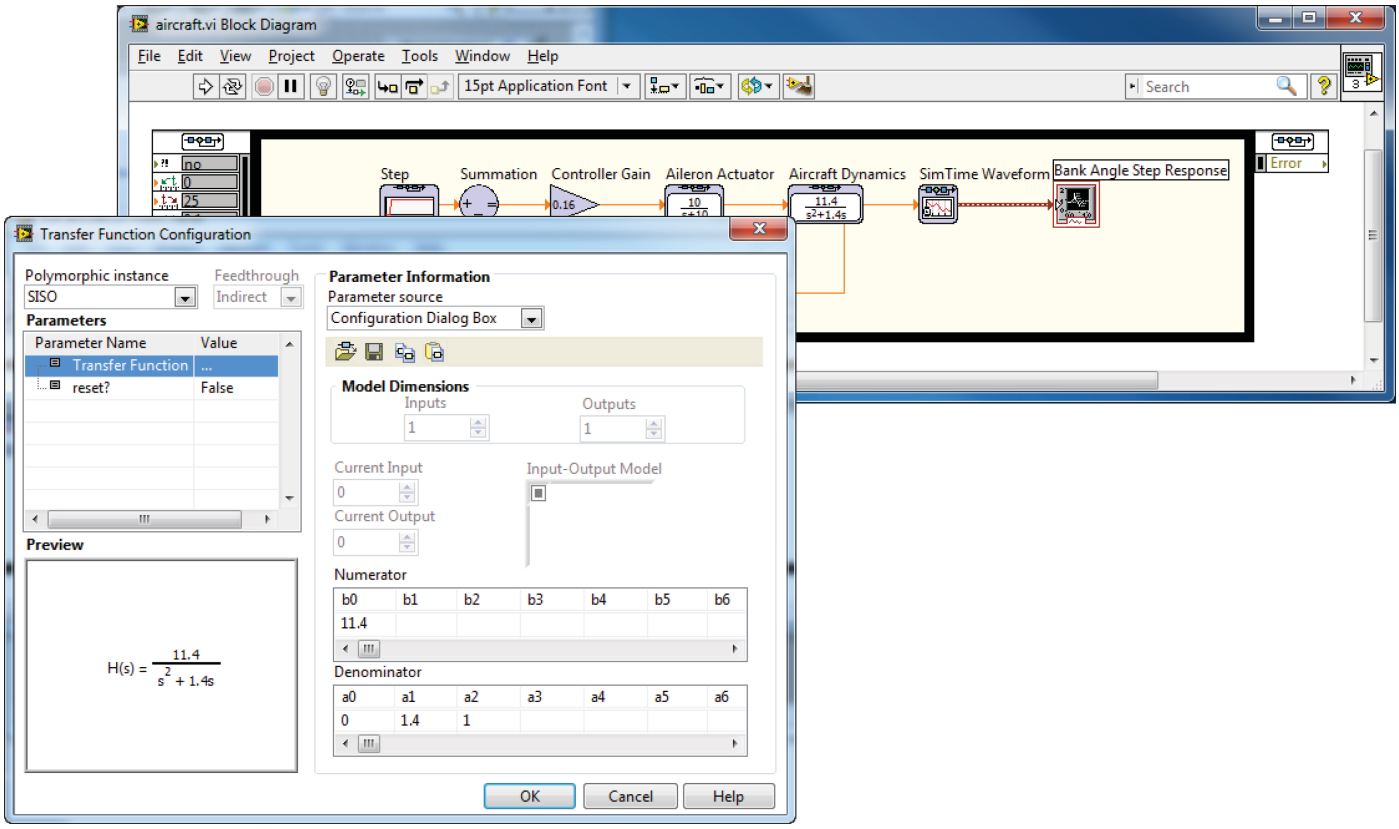


Figure 4.11: Pop-up window used to change the aircraft dynamics transfer function.

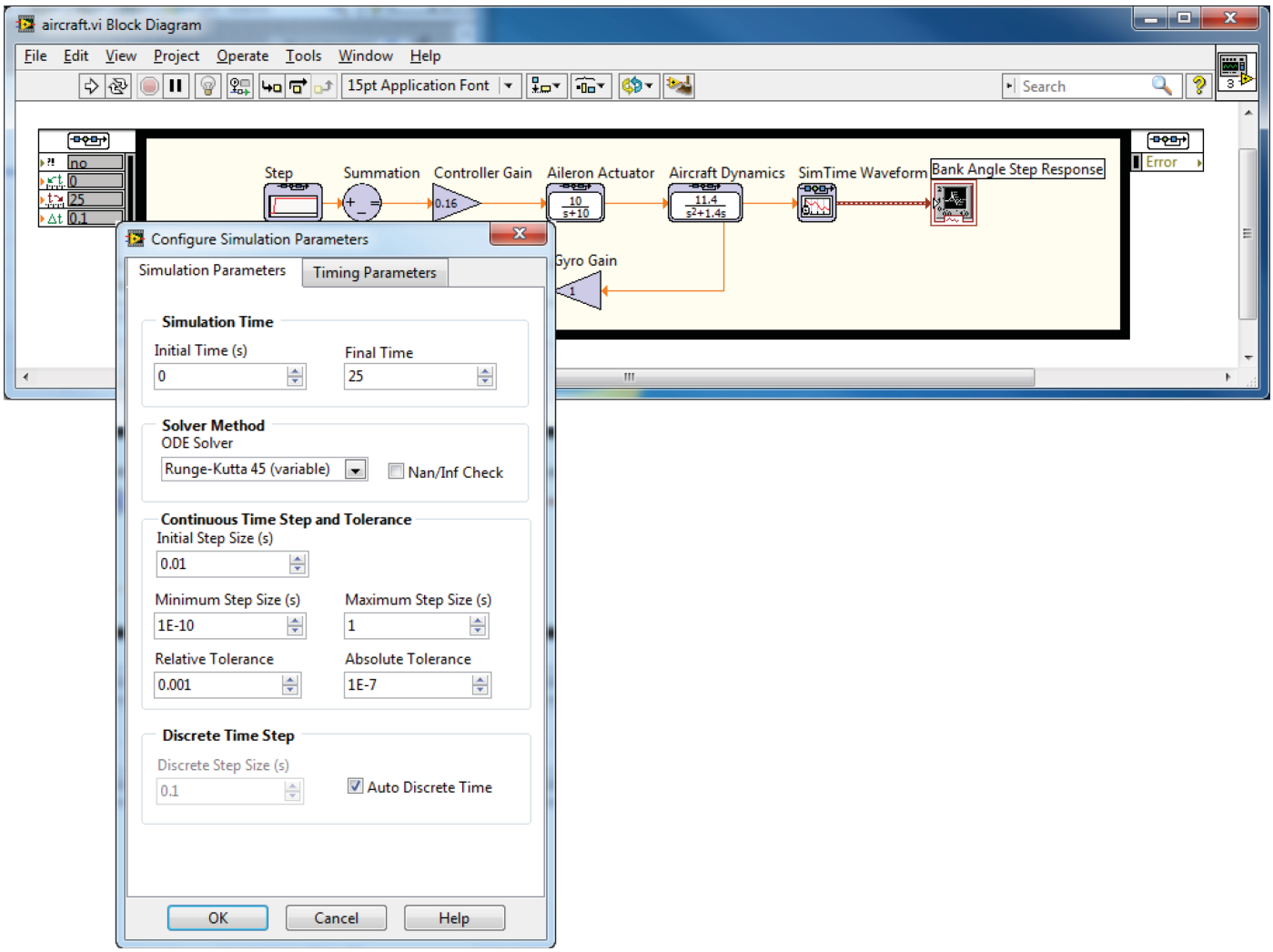


Figure 4.12: Selecting Parameters from the Configure Simulation Paraeters menu.

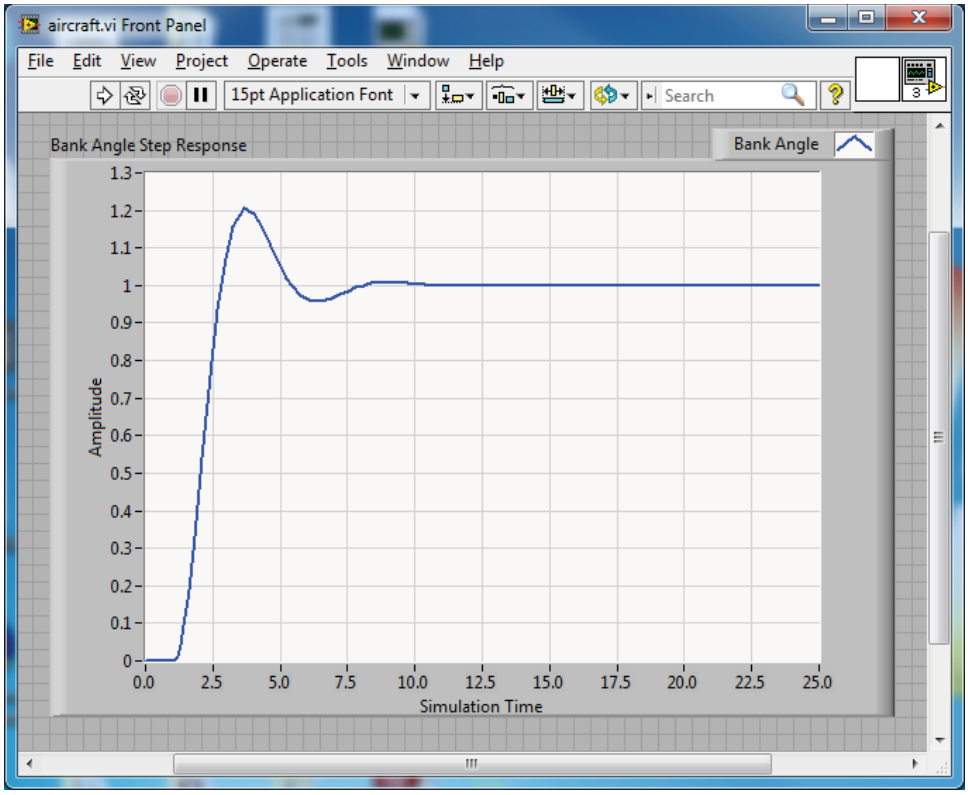


Figure 4.13: The LabVIEW simulation with the scaled graph showing the aircraft bank angle response to a step input.

CHAPTER 5

Stability of Linear Feedback Systems

This chapter begins with a discussion of the Routh–Hurwitz stability method. We will see how LabVIEW can assist us in stability analysis by providing a readily accessible method for computing the poles of the characteristic equation. For the case when the characteristic equation is a function of a single parameter, it will be possible to generate a plot displaying the movement of the poles as the parameter varies. The chapter concludes with discussion on stability for state variable systems.

5.1 Routh–Hurwitz Stability

The Routh–Hurwitz criterion is a necessary and sufficient criterion for stability. Given a characteristic equation with fixed coefficients, we can use Routh–Hurwitz to determine the number of roots in the right half-plane. For example, consider the characteristic equation

$$q(s) = s^3 + s^2 + 2s + 24 = 0$$

associated with the closed-loop control system shown in Fig. 5.1. The corresponding Routh array is shown in Fig. 5.2. The two sign changes in the first column indicate that there are two roots of the characteristic polynomial in the right half-plane; hence the closed-loop system is unstable. Using LabVIEW, we can verify the Routh–Hurwitz result by directly computing the roots of the characteristic equation using the CD Poles function (as shown in Fig. 5.3). Recall that the CD Poles function computes the system poles. Whenever the characteristic equation is a function of a single parameter, the Routh–Hurwitz method can be utilized to determine the range of values that the parameter may take while maintaining stability. Consider the closed-loop feedback system in Fig. 5.4. The characteristic equation is

$$q(s) = s^3 + 2s^2 + 4s + K = 0.$$

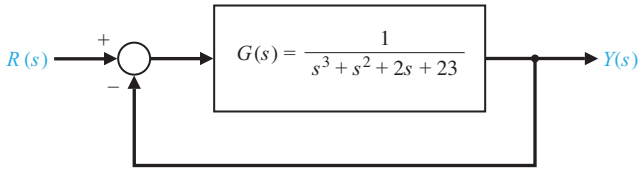


Figure 5.1: Closed-loop control system with $T(s) = Y(s)/R(s) = 1/(s^3 + s^2 + 2s + 24)$.

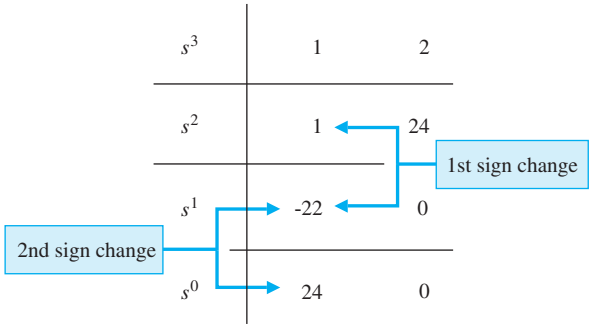


Figure 5.2: Routh array for the closed-loop control system with $T(s) = Y(s)/R(s) = 1/(s^3 + s^2 + 2s + 24)$.

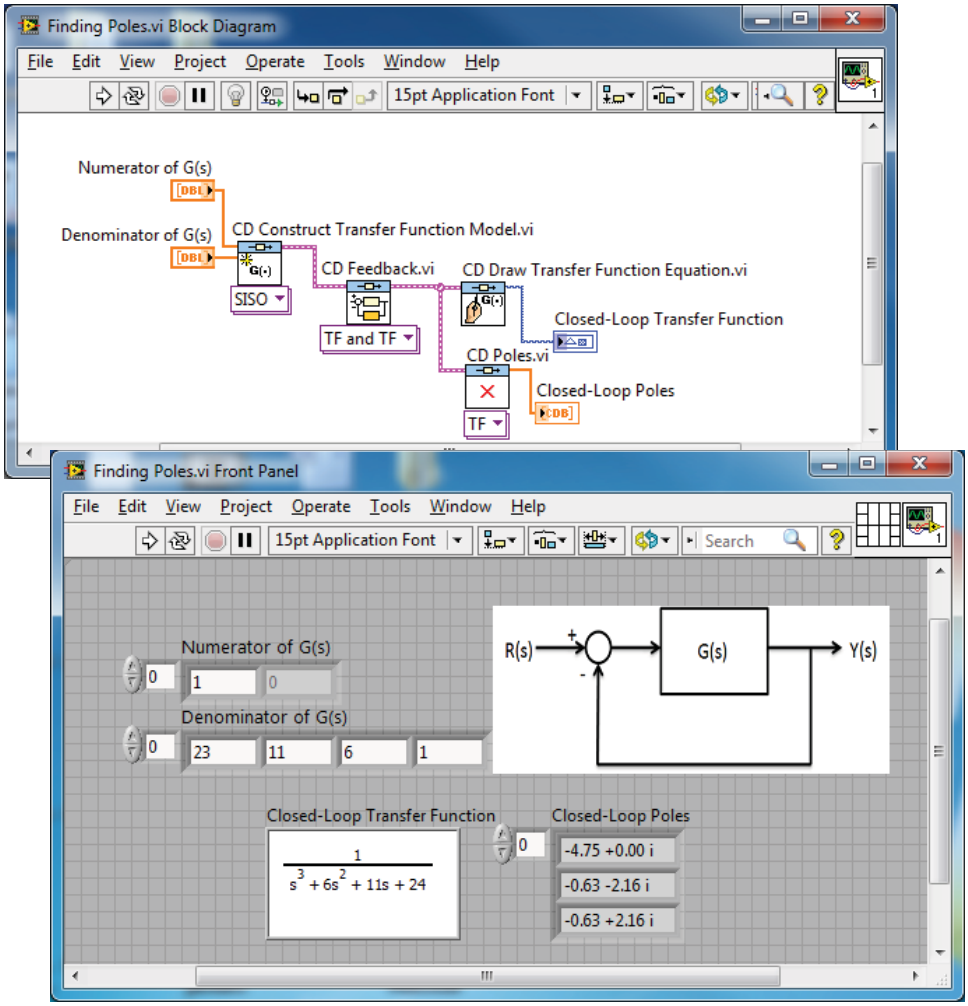


Figure 5.3: Using the CD Poles function to compute the closed-loop control system poles of the system shown in Fig. 5.1.

The associated Routh array is given by

$$\begin{array}{c|cc} s^3 & 1 & 4 \\ s^2 & 2 & K \\ s & \frac{8-K}{2} & 0 \\ 1 & K & . \end{array}$$

The Routh–Hurwitz criterion states that for closed-loop stability, all terms in the first column of the Routh array must be positive. Thus we have

$$\begin{aligned} 8 - K > 0 \quad &\Rightarrow \quad K < 8\,, \\ K > 0 \quad &\Rightarrow \quad K > 0\,. \end{aligned}$$

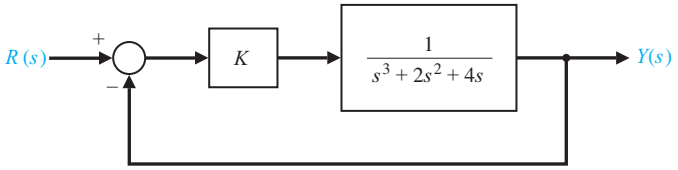


Figure 5.4: Closed-loop control system with $T(s) = Y(s)/R(s) = 1/(s^3 + 2s^2 + 4s + K)$.

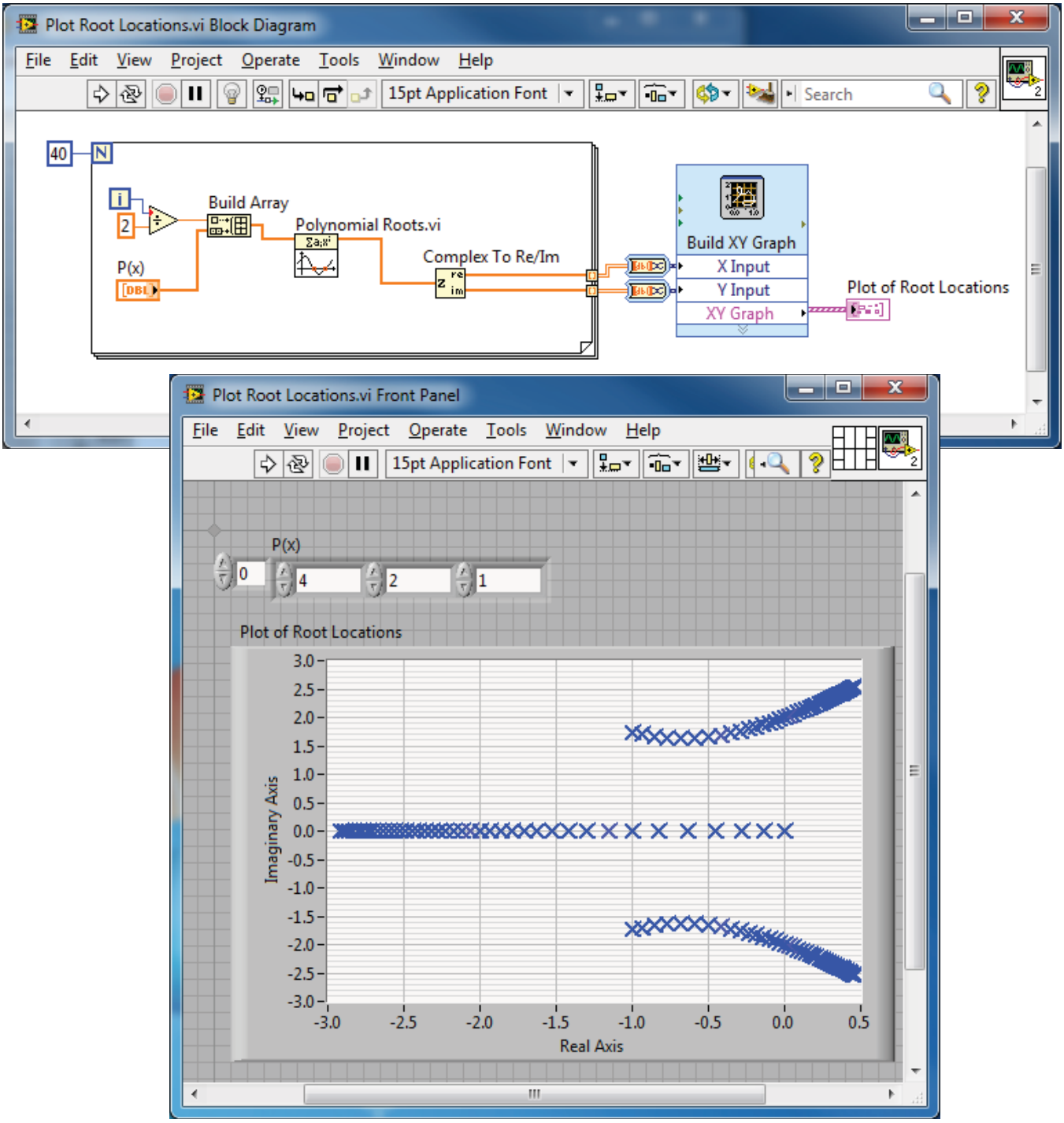


Figure 5.5: Plot of root locations of $q(s) = s^3 + 2s^2 + 4s + K$ for $0 \leq K \leq 20$.

We can use LabVIEW to verify this result graphically. As shown in Fig. 5.5, we establish a vector of values for K at which we wish to compute the roots of the characteristic equation. Then using the Polynomial Roots function, we calculate and plot the roots of the characteristic equation, as shown in Fig. 5.5. It can be seen that as K increases, the roots of the characteristic equation move toward the right half-plane as the gain tends toward $K = 8$, and eventually into the right half-plane when $K > 8$.

The VI in Fig. 5.6 uses a For Loop function. The For Loop structure is described in more detail in Fig. 5.6. This structure provides a mechanism for repeatedly executing a series of statements a given number of times. The For Loop structure sets up a repeating calculation loop. Figure 5.6 describes the For Loop structure format and provides an illustrative example of its usefulness.

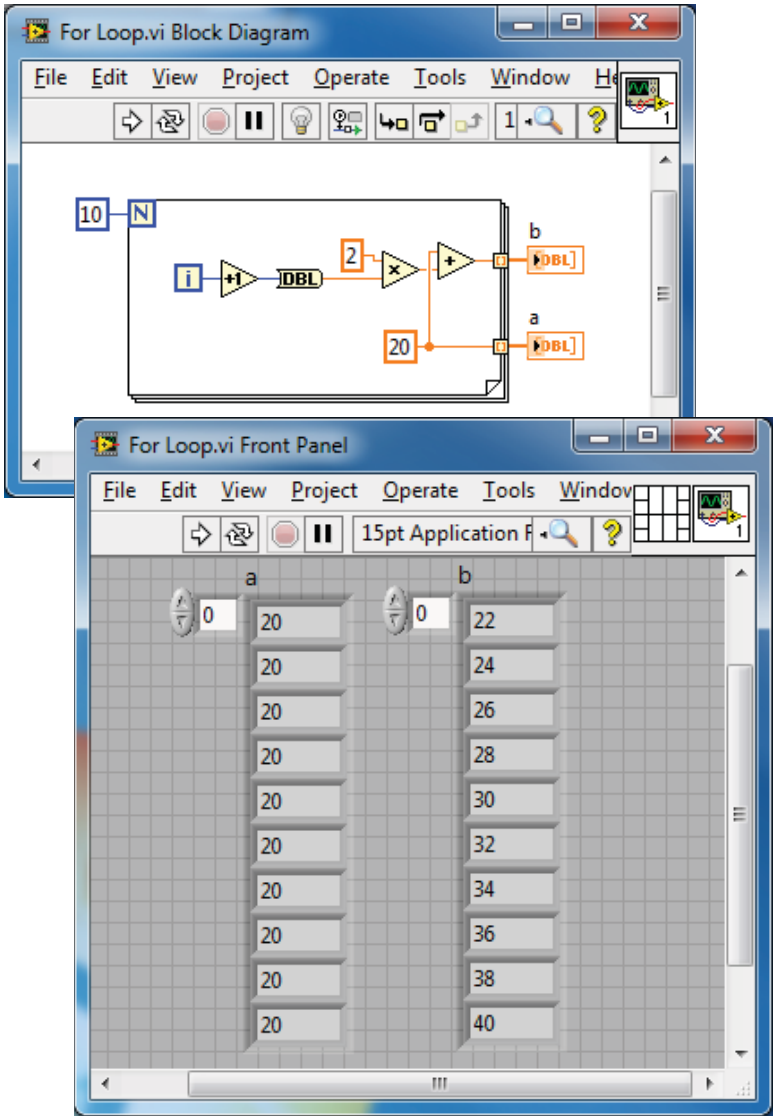


Figure 5.6: Using the For Loop structure.

The example sets up a loop that repeats ten times. During the ith iteration, where $1 \leq i \leq 10$, the ith element of the vector a is set equal to 20, and the scalar b is recomputed.

The Routh–Hurwitz method allows us to make definitive statements regarding absolute stability of a linear system. The method does not address the issue of relative stability, which is directly related to the location of the roots of the characteristic equation. Routh–Hurwitz tells us how many poles lie in the right half-plane, but not the specific location of the poles. With LabVIEW, we can easily calculate the poles explicitly, thus allowing us to investigate relative stability.

Example 5.1 Tracked Vehicle Control

The design objective is to find a and K such that the system is stable and the steady-state error for a ramp input is less than or equal to 24% of the command. We can use the Routh–Hurwitz method to aid in the search for appropriate values of a and K. The closed-loop characteristic equation is

$$q(s) = s^4 + 8s^3 + 17s^2 + (K + 10)s + aK = 0.$$

Using the Routh array, we find that for stability we require

$$K < 126, \qquad aK > 0.$$

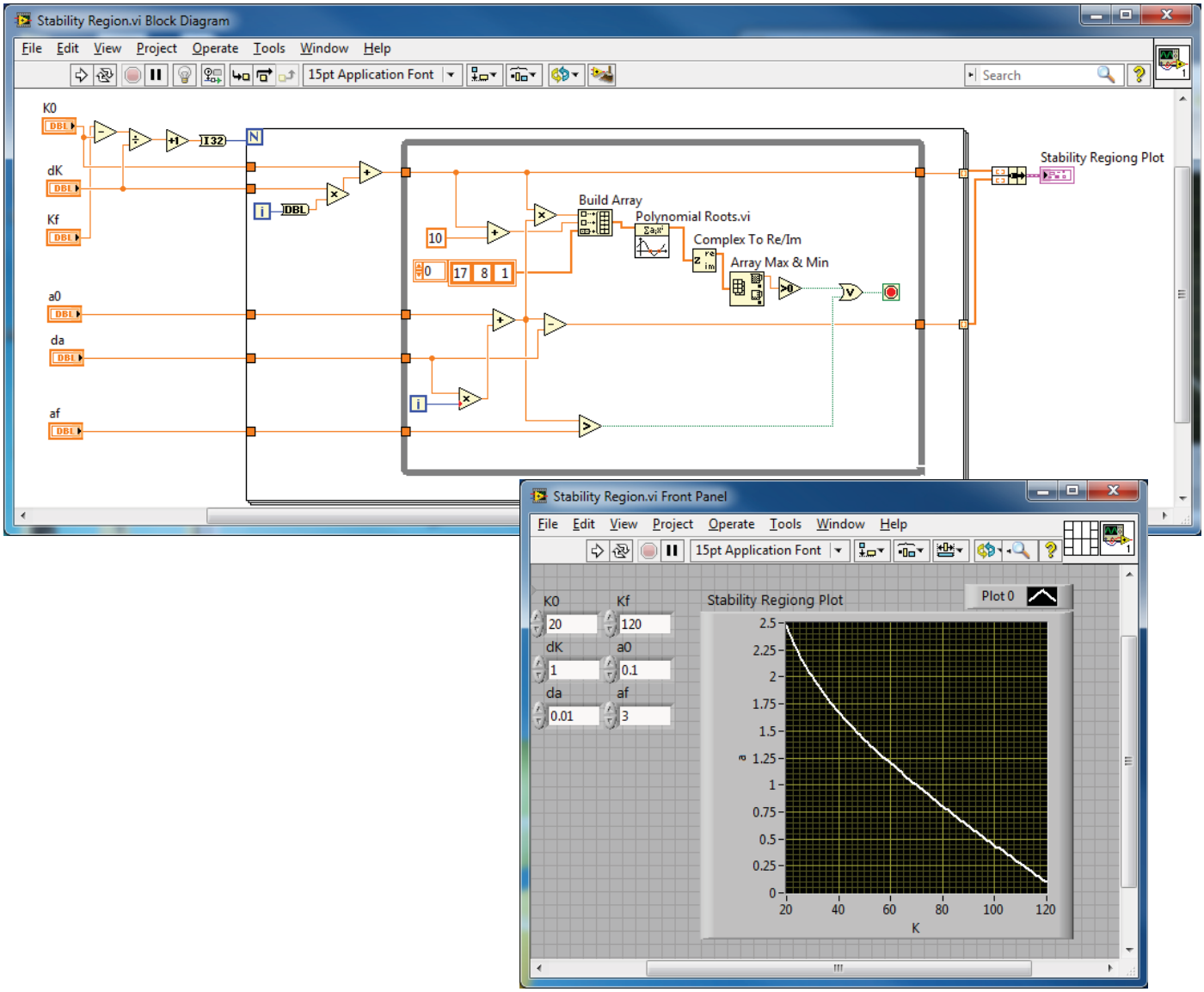


Figure 5.7: Stability region for a and K for two-track vehicle turning control.

For positive K it follows that we can restrict our search to $0 < K < 126$ and $a > 0$. Our approach will be to use LabVIEW to find a parameterized a versus K region in which stability is assured. Then we can find a set of (a, K) belonging to the stable region such that the steady-state error specification is met. This procedure, shown in Fig. 5.7, involves selecting a range of values for a and K and computing the roots of the characteristic polynomial for specific values of a and K. For each value of K, we find the first value of a that results in at least one root of the characteristic equation in the right half-plane. The process is repeated until the entire selected range of a and K is exhausted. The plot of the (a, K) pairs defines the separation between the stable and unstable regions. The region to the left of the plot of a versus K in Fig. 5.7 is the stable region. If we assume that $r(t) = At, t > 0$, then the steady-state error is

$$e_{ss} = \lim_{s\to 0} s \cdot \frac{s(s+1)(s+2)(s+5)}{s(s+1)(s+2)(s+5) + K(s+a)} \cdot \frac{A}{s^2} = \frac{10A}{aK},$$

where we have used the fact that

$$E(s) = \frac{1}{1 + G_c G(s)} R(s) = \frac{s(s+1)(s+2)(s+5)}{s(s+1)(s+2)(s+5) + K(s+a)} R(s).$$

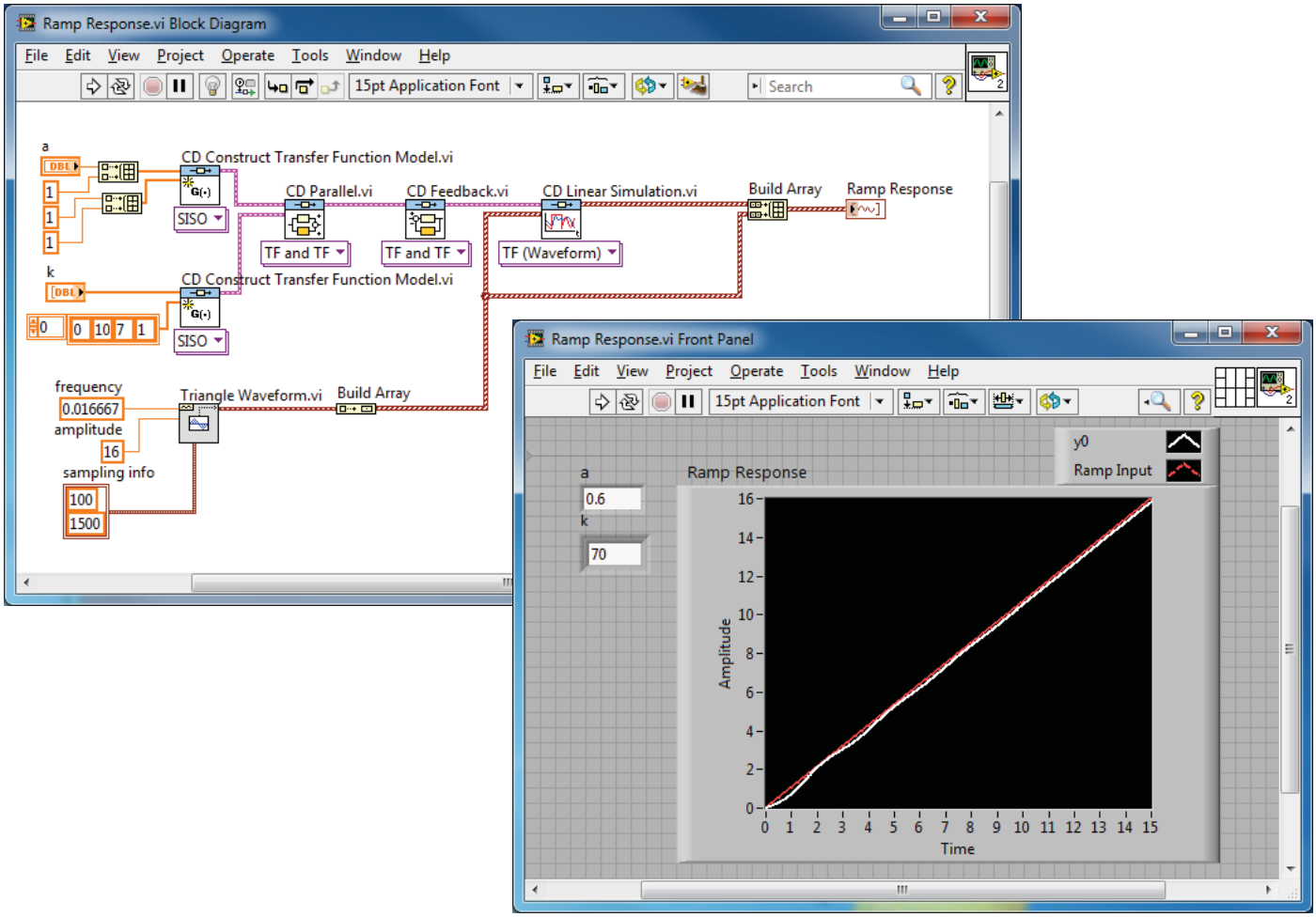


Figure 5.8: Ramp response for $a = 0.6$ and $K = 70$ for two-track vehicle turning control.

Given the steady-state specification, $e_{ss} < 0.24A$, we find that the specification is satisfied when

$$\frac{10A}{aK} < 0.24A,$$

or

$$aK > 41.67. \tag{5.1}$$

Any values of a and K that lie in the stable region in Fig. 5.7 and satisfy Eq. (5.1) will lead to an acceptable design. For example, $K = 70$ and $a = 0.6$ will satisfy all the design requirements. The closed-loop transfer function (with $a = 0.6$ and $K = 70$) is

$$T(s) = \frac{70s + 42}{s^4 + 8s^3 + 17s^2 + 80s + 42}.$$

The associated closed-loop poles are

$$\begin{aligned} s &= -7.0767, \\ s &= -0.5781, \\ s &= -0.1726 + 3.1995i, \text{ and} \\ s &= -0.1726 - 3.1995i. \end{aligned}$$

The corresponding unit ramp input response is shown in Fig. 5.8. The steady-state error is less than 0.24, as desired. ◇

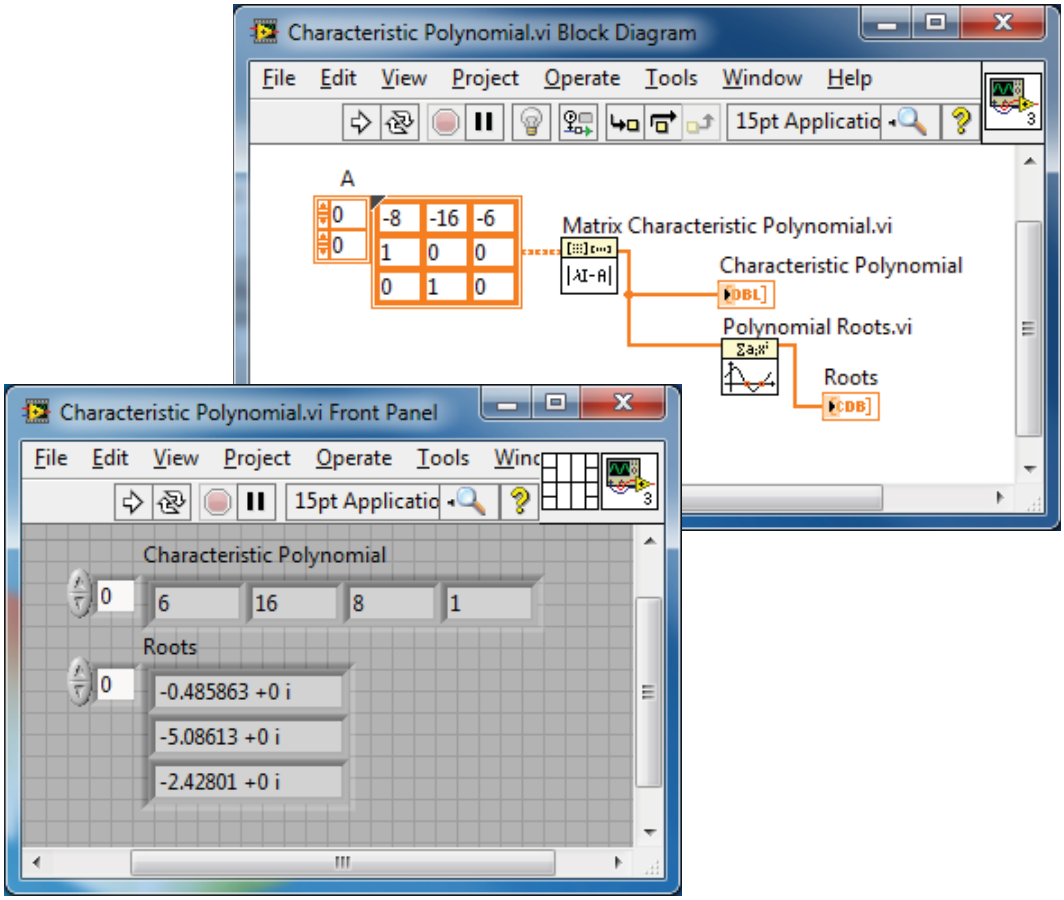


Figure 5.9: Computing the characteristic polynomial of **A** with the CD Characteristic Polynomial function.

5.2 The Stability of State Variable Systems

Now let us turn to determining the stability of systems described in state variable form. Suppose we have a system in state-space form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u.$$

The stability of the system can be evaluated with the **characteristic equation** associated with the system matrix **A**. The characteristic equation is

$$\det(s\mathbf{I} - \mathbf{A}) = 0. \tag{5.2}$$

The left-hand side of the characteristic equation is a polynomial in s. If all of the roots of the characteristic equation have negative real parts (i.e., $\mathrm{Re}(s_i) < 0$), then the system is stable.

When the system model is given in state variable form, we must calculate the characteristic polynomial associated with the **A** matrix. In this regard we have several options. We can calculate the characteristic equation directly from Eq. (5.2) by manually computing the determinant of $s\mathbf{I} - \mathbf{A}$. Then we can compute the roots using the polynomial roots function to check for stability, or alternatively, we can utilize the Routh–Hurwitz method to detect any unstable roots. Unfortunately the manual computations can become lengthy, especially if the dimension of **A** is large. We would like to avoid this manual computation if possible. As it turns out, LabVIEW can assist in this endeavor. The Matrix Characteristic Polynomial function is used to compute the characteristic equation associated with **A**, as illustrated in Fig. 5.9.

The input matrix **A** is

$$\mathbf{A} = \begin{bmatrix} -8 & -16 & -6 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

and the associated characteristic polynomial is

$$s^3 + 8s^2 + 16s + 6 = 0.$$

If **A** is an $n \times n$ matrix, Matrix Characteristic Polynomial(**A**) is the characteristic equation represented by the $n + 1$ element row vector whose elements are the coefficients of the characteristic equation.

Example 5.2 Stability Region for an Unstable Plant

A jump-jet aircraft has a control system as shown in Fig. 5.10. Assume that $z > 0$ and $p > 0$. The system is open-loop unstable (without feedback) since the characteristic equation of the plant and controller is

$$s(s-1)(s+p) = s[s^2 + (p-1)s - p] = 0.$$

Note that since one term within the bracket has a negative coefficient, the characteristic equation has at least one root in the right-hand s-plane. The characteristic equation of the closed-loop system is

$$s^3 + (p-1)s^2 + (K-p)s + Kz = 0.$$

The goal is to determine the region of stability for K, p, and z. The Routh array is

$$\begin{array}{c|cc} s^3 & 1 & K-p \\ s^2 & p-1 & Kz \\ s & b_2 & 0 \\ 1 & Kz & \end{array}$$

where

$$b_2 = \frac{(p-1)(K-p) - Kz}{p-1}.$$

From the Routh–Hurwitz criterion, we require $Kz > 0$ and $p > 1$. Setting $b_2 = 0$, we have

$$(p-1)(K-p) - Kz = K[(p-1) - z] - p(p-1) = 0.$$

Therefore we require that

$$K > \frac{p(p-1)}{(p-1) - z}. \tag{5.3}$$

Consider three cases:

1. $z > p - 1$: Since $p > 1$, any $K > 0$ will satisfy the stability conditions.

2. $z = p - 1$: There is no $0 < K < \infty$ that leads to stability.

3. $z < p - 1$: Any $0 < K < \infty$ satisfying stability condition Eq. (5.3) for a given p and z will result in stability.

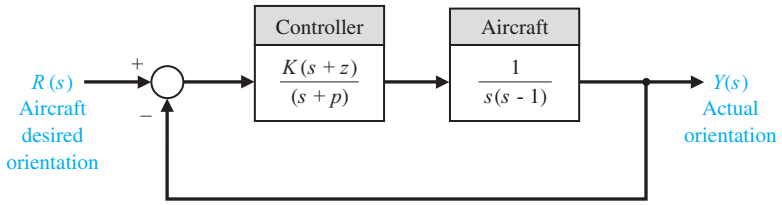


Figure 5.10: Control system for jump-jet aircraft. Assume that $z > 0$ and $p > 0$.

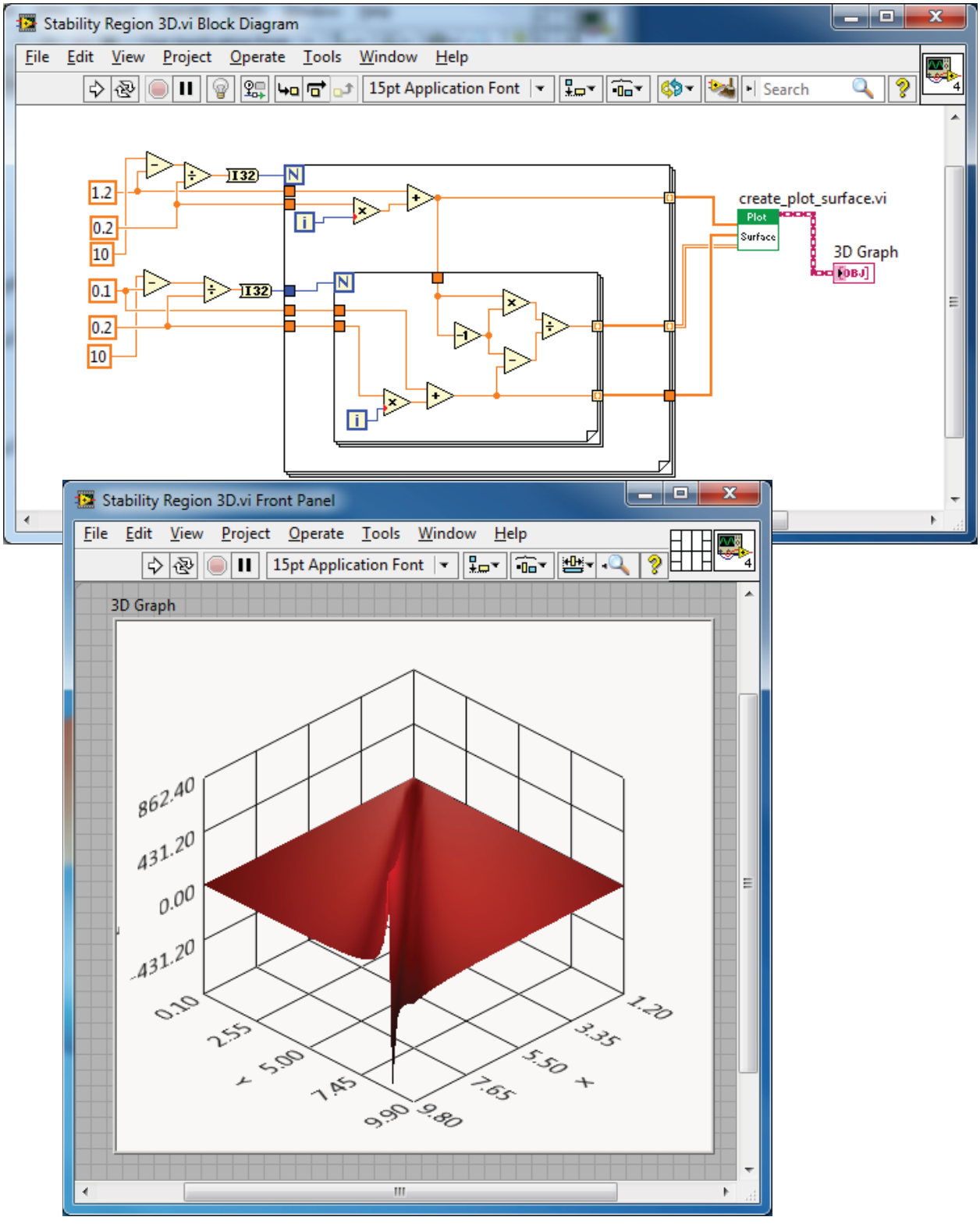


Figure 5.11: The three-dimensional region of stability lies above the surface shown.

The stability conditions can be depicted graphically. The LabVIEW script used to generate a three-dimensional stability surface is shown in Fig. 5.11. One acceptable stability point is $z = 1$, $p = 10$, and $K = 15$. ◇

CHAPTER 6

Root Locus Method

LabVIEW can readily be used to obtain accurate root locus plots. The fundamental concepts associated with *sketching* a root locus are embedded in the CD Root Locus function. If you are not familiar with how to sketch a root locus, it is recommended that you learn. It is essential for good control system design with root locus to fully understand the underlying theory. The chapter begins with a discussion on obtaining a root locus plot with LabVIEW. This is followed by a discussion of the connections between the partial fraction expansion, dominant poles, and the closed-loop system response. Root sensitivity is covered in the final paragraphs. The functions covered in this section are CD Root Locus and Partial Fraction Expansion.

6.1 Obtaining a Root Locus Plot

Consider the closed-loop control system in Fig. 6.1. The closed-loop transfer function is

$$T(s) = \frac{Y(s)}{R(s)} = \frac{K(s+1)(s+3)}{s(s+2)(s+3)+K(s+1)}.$$

The characteristic equation can be written as

$$1 + K\frac{s+1}{s(s+2)(s+3)} = 0. \tag{6.1}$$

The characteristic equation in Eq. (6.1) is in the necessary Evans form for generating root locus plots using LabVIEW. The general form of the characteristic equation necessary for application of the CD Root Locus function is

$$1 + K\frac{p(s)}{q(s)} = 0, \tag{6.2}$$

where K is the parameter of interest to be varied from $0 < K < \infty$. The CD Root Locus function is used to obtain the root locus plot associated with Eq. (6.1). For the characteristic equation in Eq. (6.1), the associated root locus plot is shown in Fig. 6.2.

The steps required to use CD Root Locus to obtain a root locus plot with LabVIEW are as follows:

1. Obtain the characteristic equation in the form given in Eq. (6.2), where K is the parameter of interest.

2. Use the CD Root Locus function to generate the plots.

Referring to Fig. 6.2, we see that as K increases, two branches of the root locus break away from the real axis. This means that for some values of K the closed-loop system characteristic equation will have complex roots. For example, we find that when

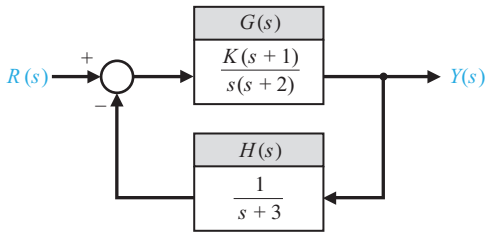


Figure 6.1: Closed-loop system.

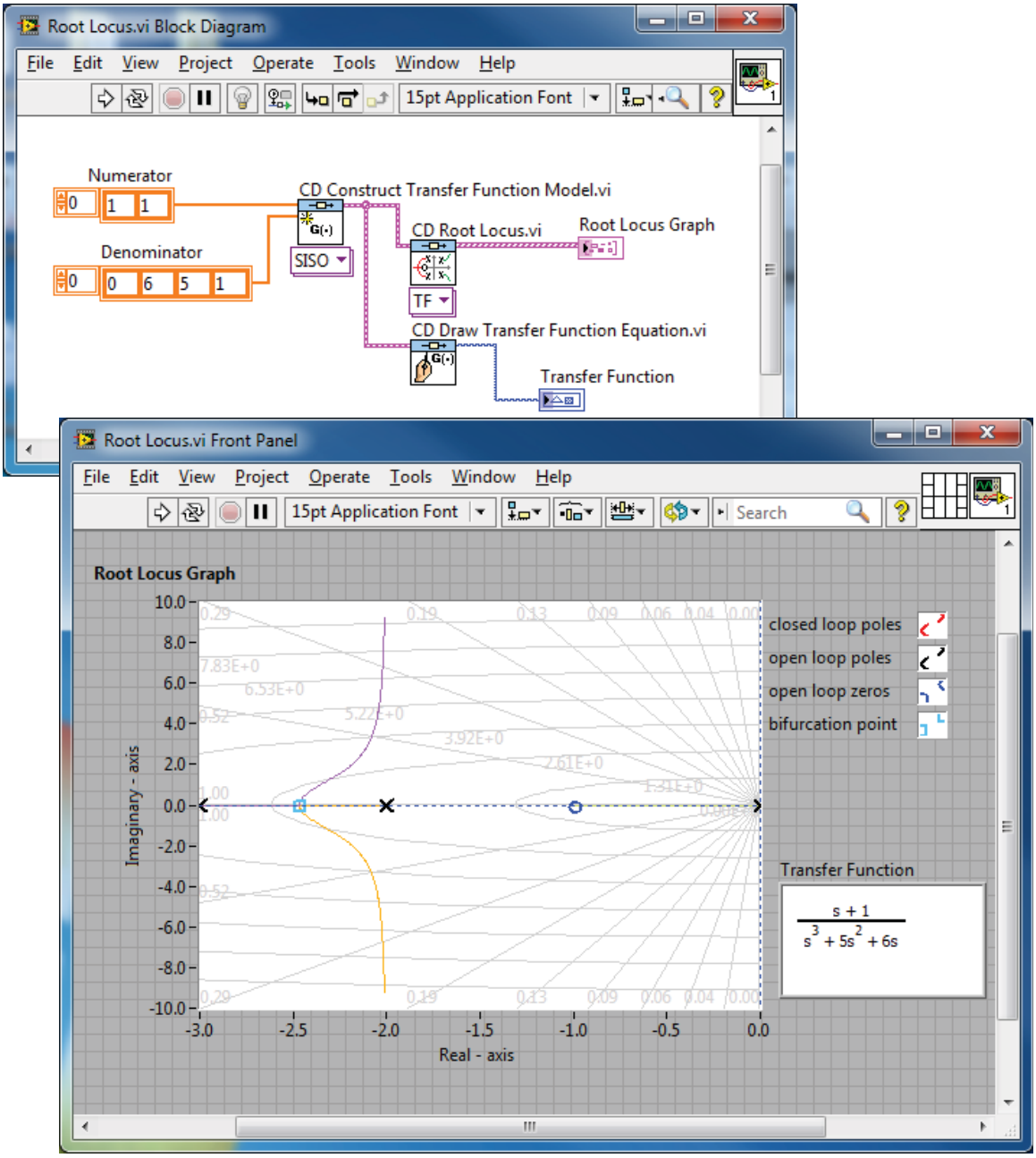


Figure 6.2: The **CD Root Locus** function.

$K = 20.5775$, the closed-loop transfer function has three poles and two zeros at poles:

$$\text{poles} : s = \begin{pmatrix} -2.0505 + 4.3227i \\ -2.0505 - 4.3227i \\ -0.8989 \end{pmatrix}, \quad \text{zeros} : s = \begin{pmatrix} -1 \\ -3 \end{pmatrix}.$$

Considering the closed-loop pole locations only, we would expect that the real pole at $s = -0.8989$ would be the **dominant pole**. To verify this, we can study the closed-loop system response to a step input, $R(s) = 1/s$. For a step input we have

$$Y(s) = \frac{20.5775(s+1)(s+3)}{s(s+2)(s+3) + 20.5775(s+1)} \cdot \frac{1}{s}. \tag{6.3}$$

Generally the first step in computing $y(t)$ is to expand Eq. (6.3) in a partial fraction expansion. The Partial Fraction Expansion function can be used to expand Eq. (6.3) in a partial fraction expansion, as shown in Fig. 6.3. The partial fraction expansion of

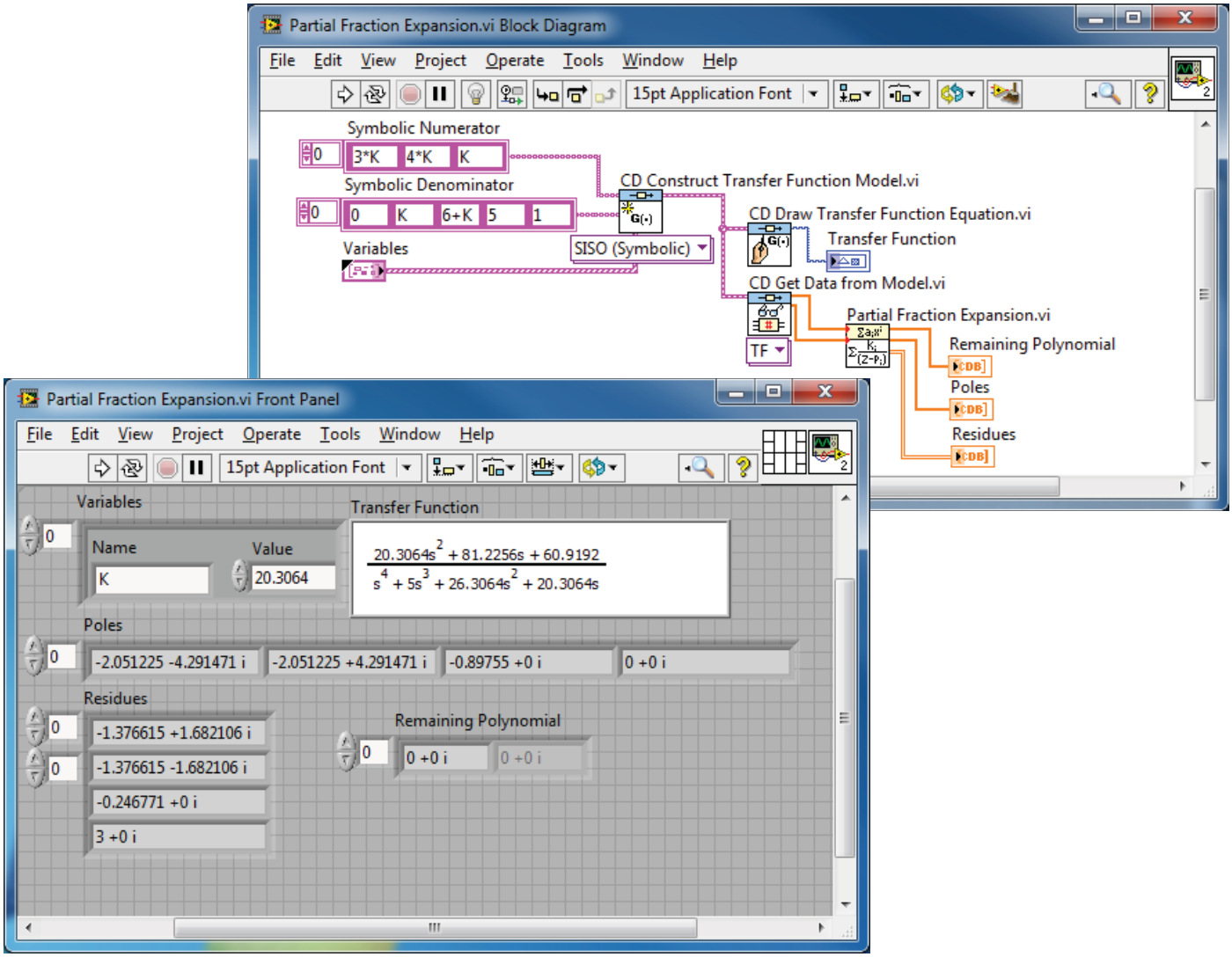


Figure 6.3: Partial fraction expansion of Eq. (6.3).

Eq. (6.3) is

$$Y(s) = \frac{-1.3786 + 1.7010i}{s + 2.0505 + 4.3228i} + \frac{-1.3786 - 1.7010i}{s + 2.0505 - 4.3228i} + \frac{-0.2429}{s + 0.8989} + \frac{3}{s}.$$

Comparing the terms of the partial fraction expansion, we see that the coefficient of the term corresponding to the pole at $s = -0.8989$ is considerably smaller than the coefficient of the terms corresponding to the complex-conjugate poles at $s = -2.0505 \pm j4.3227$. From this we expect that the influence of the pole at $s = -0.8989$ on the output response $y(t)$ does not dominate. The settling time to within 2% of the final value is then predicted by considering the complex-conjugate poles. The poles at $s = -2.0505 \pm j4.3227$ correspond to a damping of $\zeta = 0.4286$ and a natural frequency of $\omega_n = 4.7844$. Thus the settling time is predicted to be

$$T_s \approx \frac{4}{\zeta \omega_n} = 1.95 \text{ seconds}.$$

Using the CD Paraetric Time Response function, as shown in Fig. 6.4, we find that $T_s \approx 1.6$ seconds. Hence our approximation of settling time $T_s \approx 1.95$ is a fairly good approximation.

In this example the role of the system zeros on the transient response is illustrated. The proximity of the zero at $s = -1$ to the pole at $s = -0.8989$ reduces the impact of that pole on the transient response. The main contributors to the transient response are the complex-conjugate poles at $s = -2.0505 \pm j4.3228$ and the zero at $s = -3$.

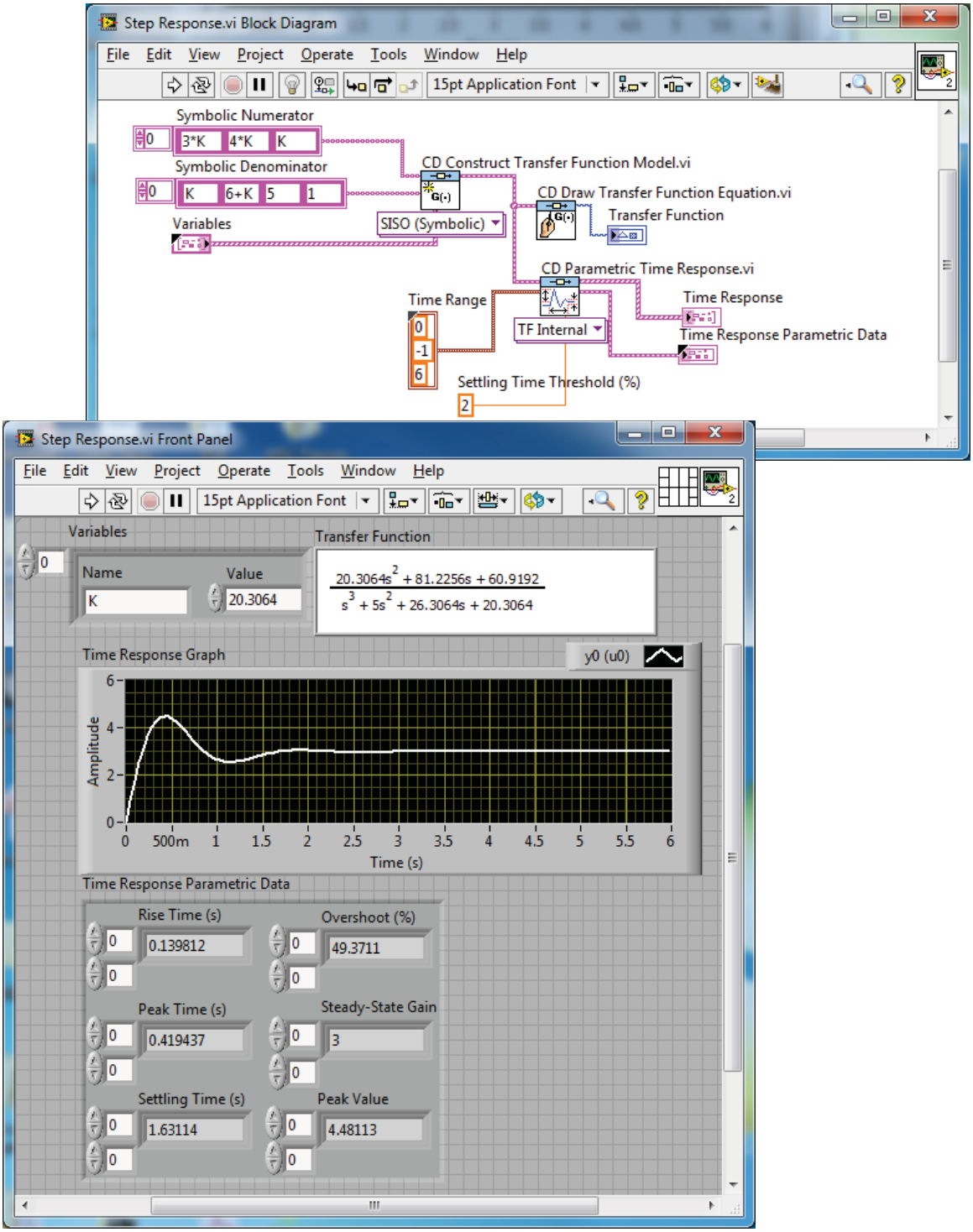


Figure 6.4: CD Step Response function with the closed-loop system in Fig. 6.1 with $K = 20.5775$.

6.1.1 Sensitivity and the Root Locus

The roots of the characteristic equation play an important role in defining the closed-loop system transient response. The effect of parameter variations on the roots of the characteristic equation is a useful measure of sensitivity. The root sensitivity can be defined to be

$$S_K^{r_i} = \frac{\partial r_i}{\partial K / K}. \tag{6.4}$$

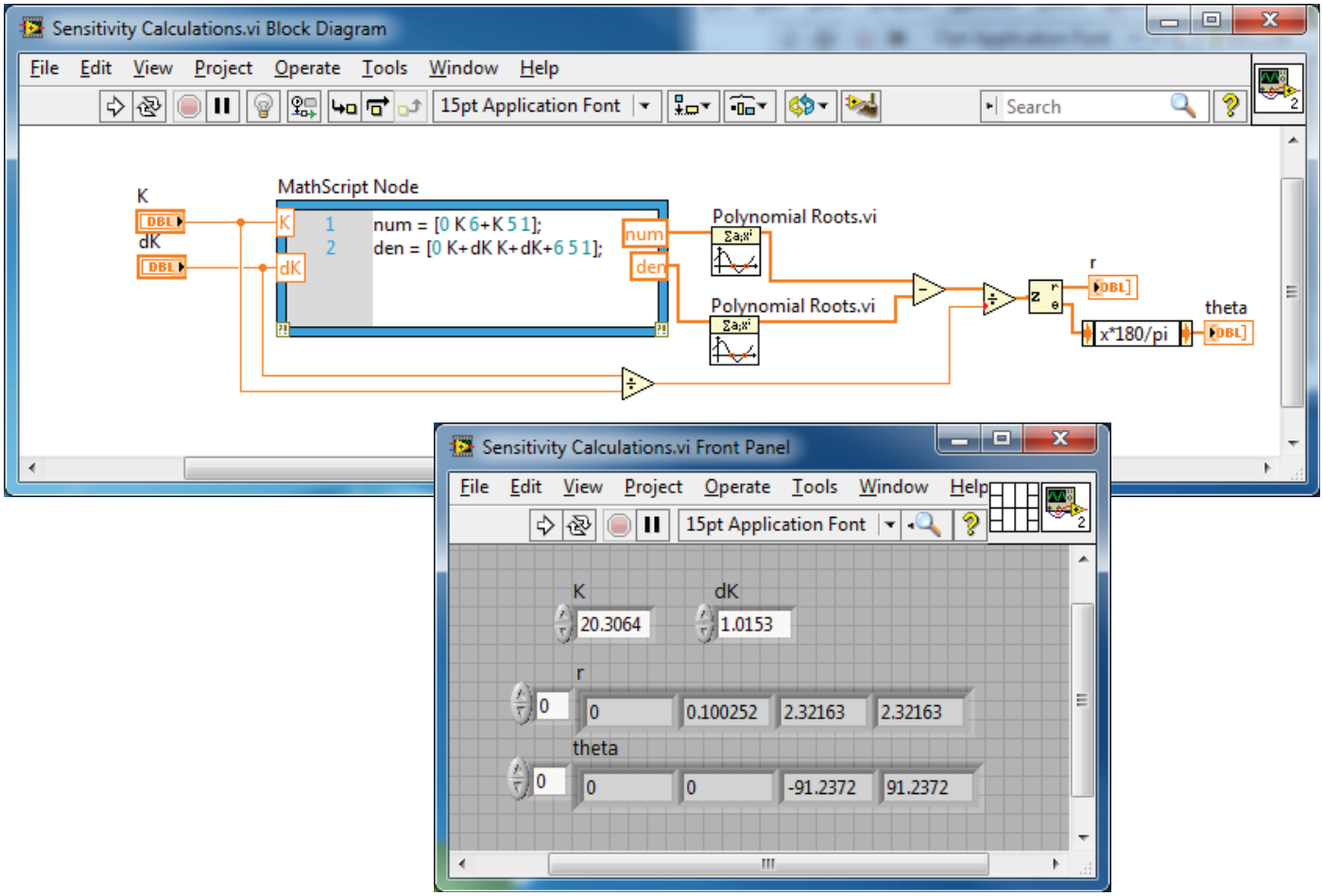


Figure 6.5: Sensitivity calculations for the root locus for a 5% change in K.

We can utilize Eq. (6.4) to investigate the sensitivity of the roots of the characteristic equation to variations in the parameter K. If we change K by a small finite amount ΔK, and evaluate the modified root $r_i + \Delta r_i$, it follows that

$$S_K^{r_i} \approx \frac{\Delta r_i}{\Delta K / K}. \tag{6.5}$$

In general, the quantity $S_K^{r_i}$ is a complex number. Referring back to the third-order example of Fig. 6.1, if we change K by a factor of 5%, we find that the dominant complex-conjugate pole at $s = -2.0505 + j4.3228$ changes by

$$\Delta r_i = -0.0025 - 0.1168i$$

when K changes from $K = 20.5775$ to $K = 21.6064$. From Eq. (6.5), it follows that

$$S_K^{r_i} = \frac{-0.0025 - 0.1168i}{1.0289/20.5775} = -0.0494 - 2.3355i.$$

The sensitivity $S_K^{r_i}$ can also be written in the form

$$S_K^{r_i} = 2.3360 \quad \angle 268.7872°.$$

The magnitude and direction of $S_K^{r_i}$ provides a measure of the root sensitivity. The script used to perform these sensitivity calculations is shown in Fig. 6.5. The root sensitivity measure is useful for comparing the sensitivity for various system parameters at different root locations.

CHAPTER 7

Frequency Response Methods

This chapter begins with an introduction to the Bode plot and then discusses the connection between the frequency response and performance specifications in the time domain. We conclude with an illustrative example of designing a control system in the frequency domain. The LabVIEW function covered is CD Bode. The CD Bode function is used to generate a Bode plot. We will use transfer function models in this chapter, but the use of the CD Bode function is exactly the same for state variable models as with transfer functions, except that the input is a state-space object instead of a transfer function object.

7.1 Bode Plots

Consider the transfer function

$$G(s) = \frac{5(1+0.1s)}{s(1+0.5s)(1+\frac{0.6}{50}s+\frac{1}{50^2}s^2)}. \tag{7.1}$$

The Bode plot corresponding to Eq. (7.1) is shown in Fig. 7.1. The plot consists of the logarithmic gain in dB versus ω in one plot and the phase $\phi(\omega)$ versus ω in a second plot. As with the root locus plots, it will be tempting to rely exclusively on LabVIEW to obtain your Bode plots. Treat LabVIEW as one tool in a tool kit that can be used to design and analyze control systems. It is essential to develop the capability to sketch Bode plots. There is no substitute for a clear understanding of the underlying theory.

Keeping in mind the goal of designing control systems that satisfy certain performance specifications given in the time domain, we must establish a connection between the frequency response and the transient time response of a system. The relationship between specifications given in the time domain to those given in the frequency domain depends upon approximation of the system by a second-order system with the poles being the system **dominant poles**.

Consider the second-order system shown in Fig. 7.2. The closed-loop transfer function is

$$T(s) = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}. \tag{7.2}$$

The Bode plot magnitude associated with the closed-loop transfer function in Eq. (7.2) is shown in Fig. 7.3. The relationship between the resonant frequency, ω_r, the maximum of the frequency response, M_{p_ω}, and the damping ratio, ζ, and the natural frequency, ω_n, is shown in Fig. 7.4. The information in Fig. 7.4 will be quite helpful in designing control systems in the frequency domain while satisfying time-domain specifications. The relationship between the resonant frequency, ω_r, and the natural frequency, ω_n, is

$$\omega_r = \sqrt{1-2\zeta^2}\,\omega_n\,, \quad \zeta < 0.707 \tag{7.3}$$

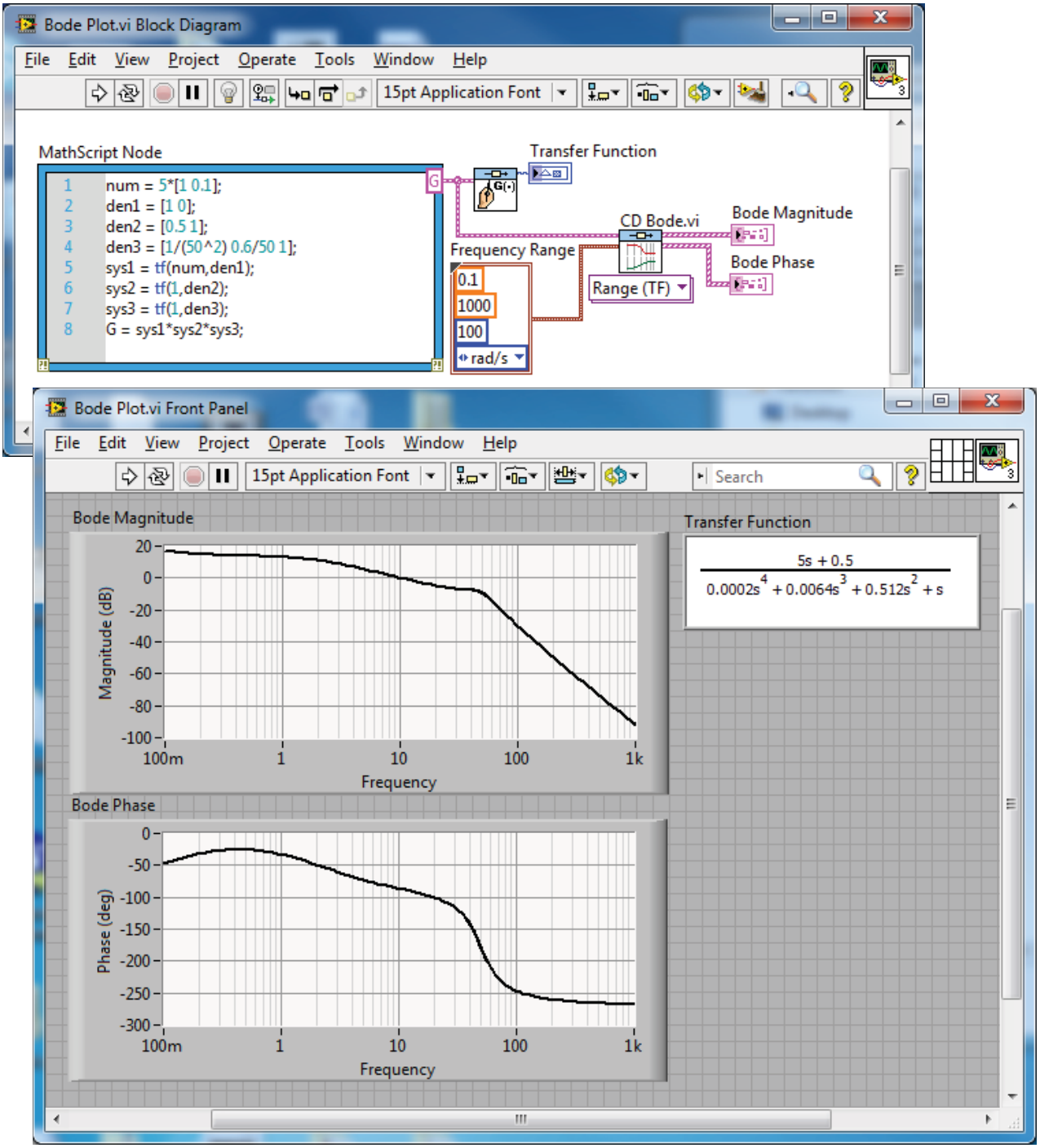


Figure 7.1: The Bode plot associated with Eq. (7.1).

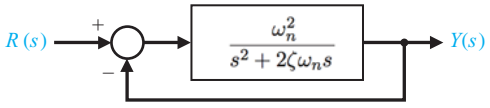


Figure 7.2: A second-order closed-loop system.

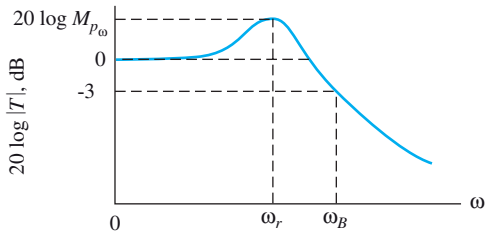


Figure 7.3: Magnitude characteristic of the second-order system.

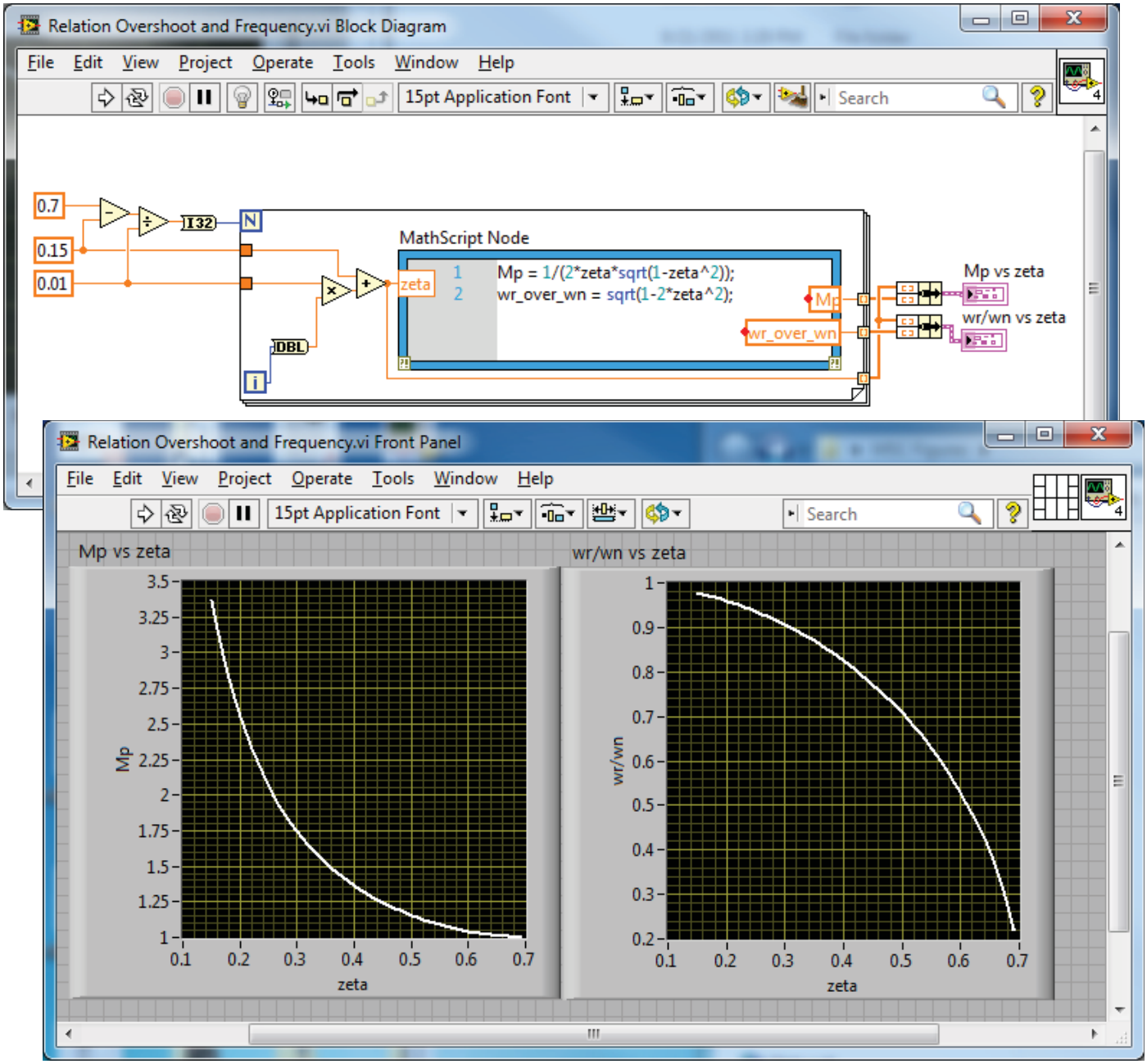


Figure 7.4: The relationship between (M_{p_ω}, ζ) and $(\omega_r/\omega_n, \zeta)$ for a second-order system.

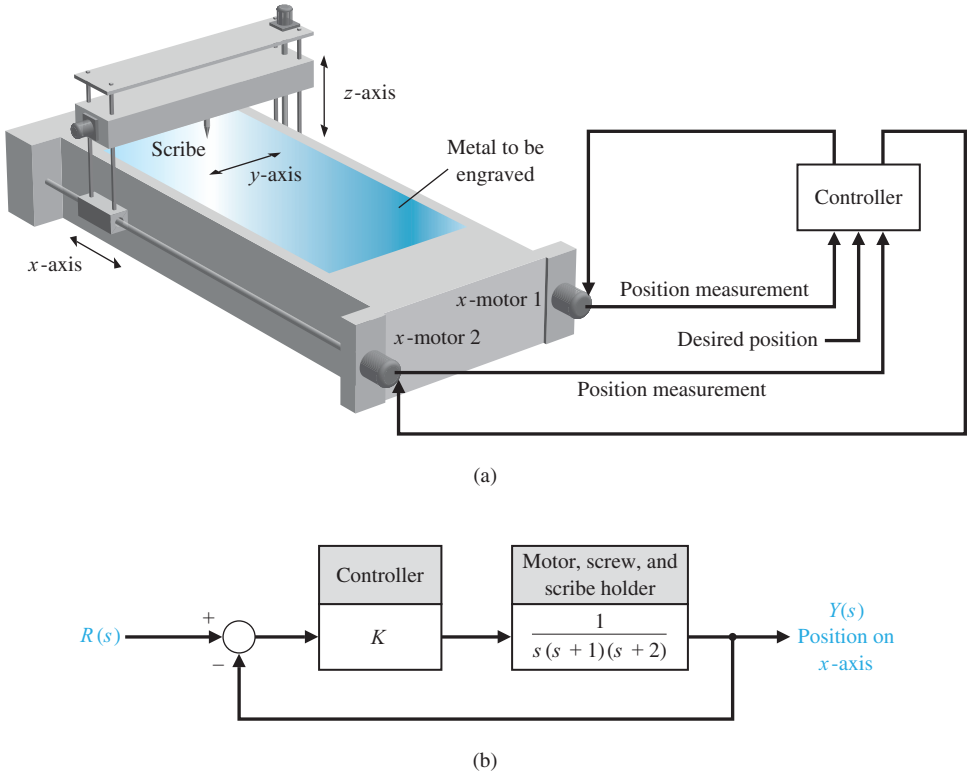


Figure 7.5: (a) Engraving machine control system. (b) Block diagram model.

for second-order systems. The relationship between the maximum frequency response, M_{p_ω}, and the damping ratio, ζ, is

$$M_{p_\omega} = \left(2\zeta\sqrt{1-\zeta^2}\right)^{-1}, \quad \zeta < 0.707 \tag{7.4}$$

Example 7.1 Engraving Machine System

Consider the block diagram model in Fig. 7.5. Our objective is to select K so that the closed-loop system has an acceptable time response to a step command. A functional block diagram describing the frequency-domain design process is shown in Fig. 7.6. First we choose $K = 2$ and then iterate on K if the performance is unacceptable. The script shown in Fig. 7.7 is used in the design. The value of K is defined at the command level. Then the script is executed and the closed-loop Bode diagram is generated. The values of M_{p_ω} and ω_r are determined by inspection from the Bode plot. Those values are used in conjunction with Fig. 7.4 to determine the corresponding values of ζ and ω_n. Given the damping ratio, ζ, and the natural frequency, ω_n, the settling time and percent overshoot are estimated using the formulas

$$T_s \approx \frac{4}{\zeta\omega_n}, \quad P.O. \approx 100\exp\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}.$$

If the time-domain specifications are not satisfied, then we adjust K and iterate. The values for ζ and ω_n corresponding to $K = 2$ are $\zeta = 0.29$ and $\omega_n = 0.88$. This leads to a prediction of $P.O. = 37\%$ and $T_s = 15.7$ seconds. The step response, shown in Fig. 7.8, is a verification that the performance predictions are quite accurate and that the closed-loop system performs adequately. ◇

In this example, the second-order system approximation is reasonable and leads to an acceptable design. However, the second-order approximation may not always lead directly to a good design. Fortunately with LabVIEW we can construct an interactive design facility to assist in the design process by reducing the manual computational loads while providing easy access to a host of classical and modern control tools.

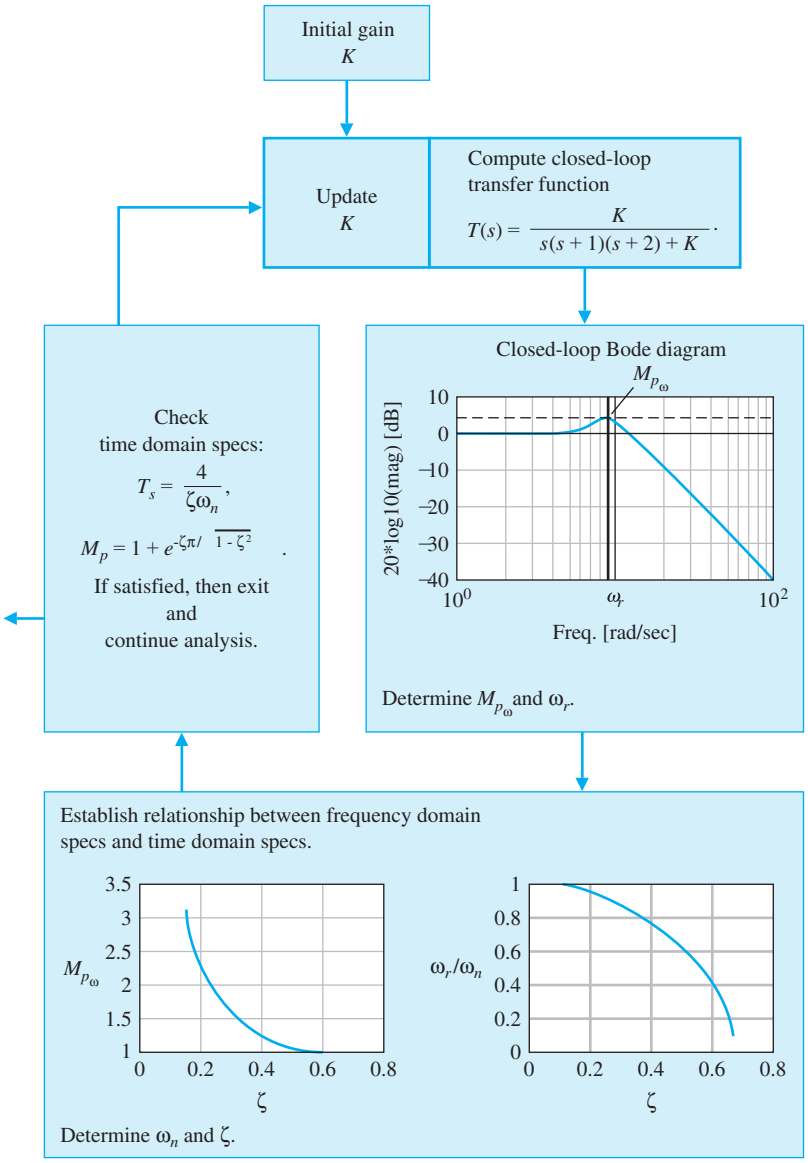


Figure 7.6: Frequency design functional block diagram for the engraving machine.

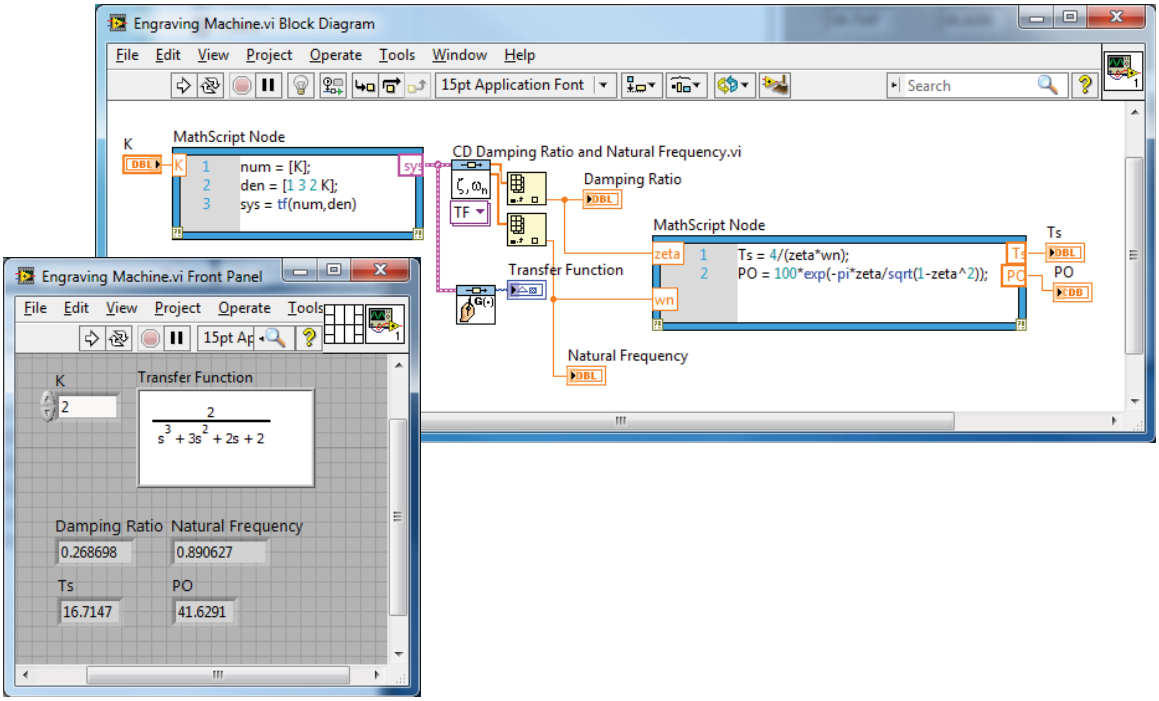


Figure 7.7: Script for the design of an engraving machine.

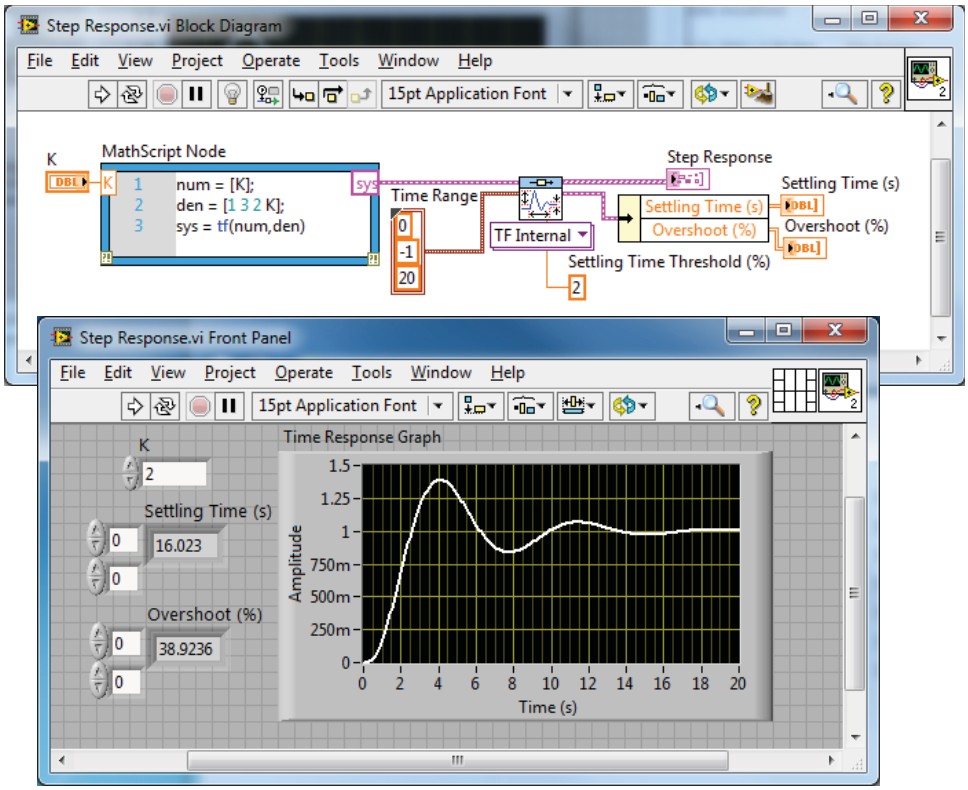


Figure 7.8: Engraving machine step response for $K = 2$.

CHAPTER 8

Stability in the Frequency Domain

This chapter considers the Nyquist diagram and the Nichols chart. Two examples are used illustrate the frequency-domain design approach. We also present an illustrative example that shows how to deal with a time delay in the system utilizing a Padé approximation. The LabVIEW functions covered in this chapter are CD Nyquist, CD Nichols, CD Gain and Phase Margin, and CD Construct Special TF Model.

8.1 Nyquist Plots

It is generally more difficult to generate the Nyquist plot manually than the Bode plot. However, we can use LabVIEW to obtain the Nyquist plot. The Nyquist plot is generated with the CD Nyquist function, as shown in Fig. 8.1.

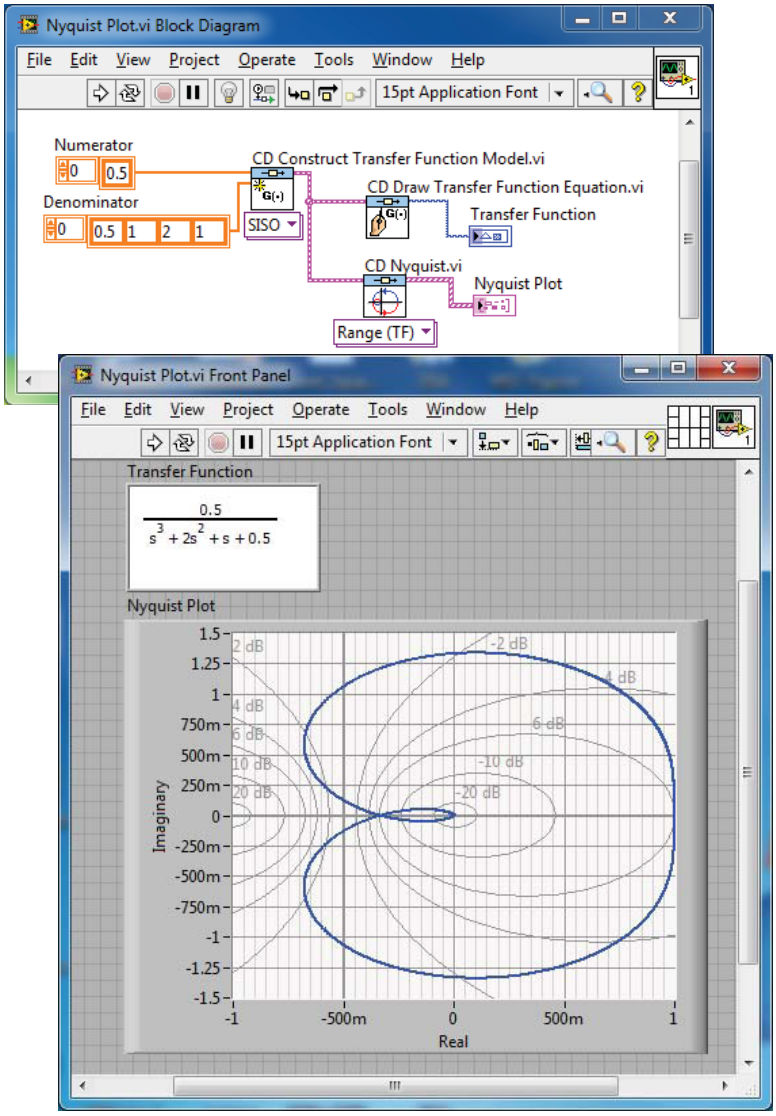


Figure 8.1: The CD Nyquist function.

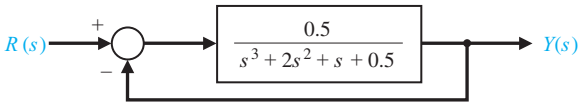


Figure 8.2: A closed-loop control system example for relative stability analysis.

Relative stability measures of **gain margin** and **phase margin** can be determined from both the Nyquist plot and the Bode plot. The gain margin is a measure of how much the system gain would have to be increased for the loop transfer function $L(j\omega)$ locus to pass through the $(-1, 0)$ point, thus resulting in an unstable system. The phase margin is a measure of the additional phase lag required before the system becomes unstable. Gain and phase margins can be determined from both the Nyquist plot and the Bode plot.

Consider the system shown in Fig. 8.2. Relative stability can be determined from the Bode diagram using the CD Gain and Phase Margin function, which is shown in Fig. 8.3. The VI used to generate the Nyquist plot for the system in Fig. 8.2 is shown in Fig. 8.4. In this case, the number of poles of the loop transfer function $L(s)$ with positive real parts is zero, and the number of counterclockwise encirclements of -1 is zero; hence the closed-loop system is stable. We can also determine the gain margin and phase margin, as indicated in Fig. 8.4.

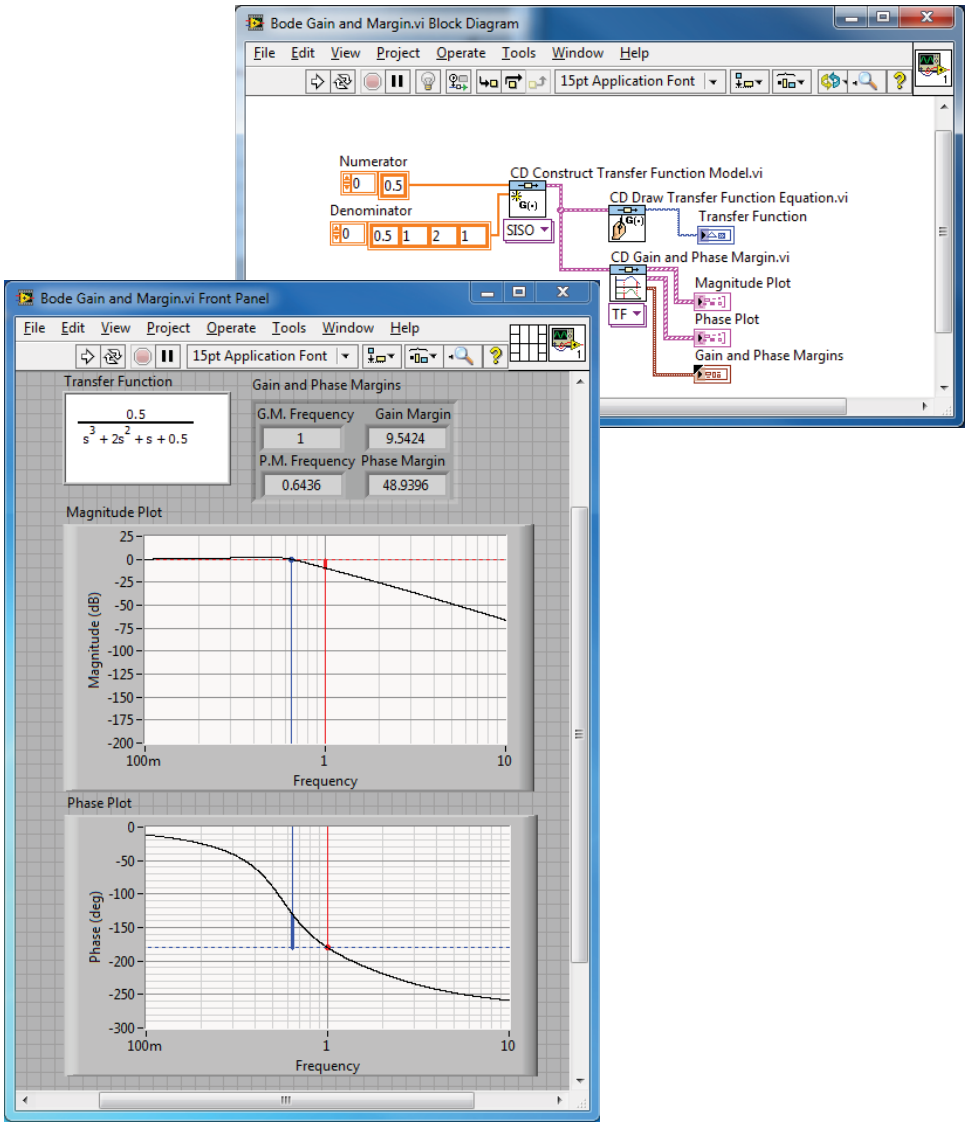


Figure 8.3: The Bode diagram for the system in Fig. 8.2 with the gain margin and phase margin.

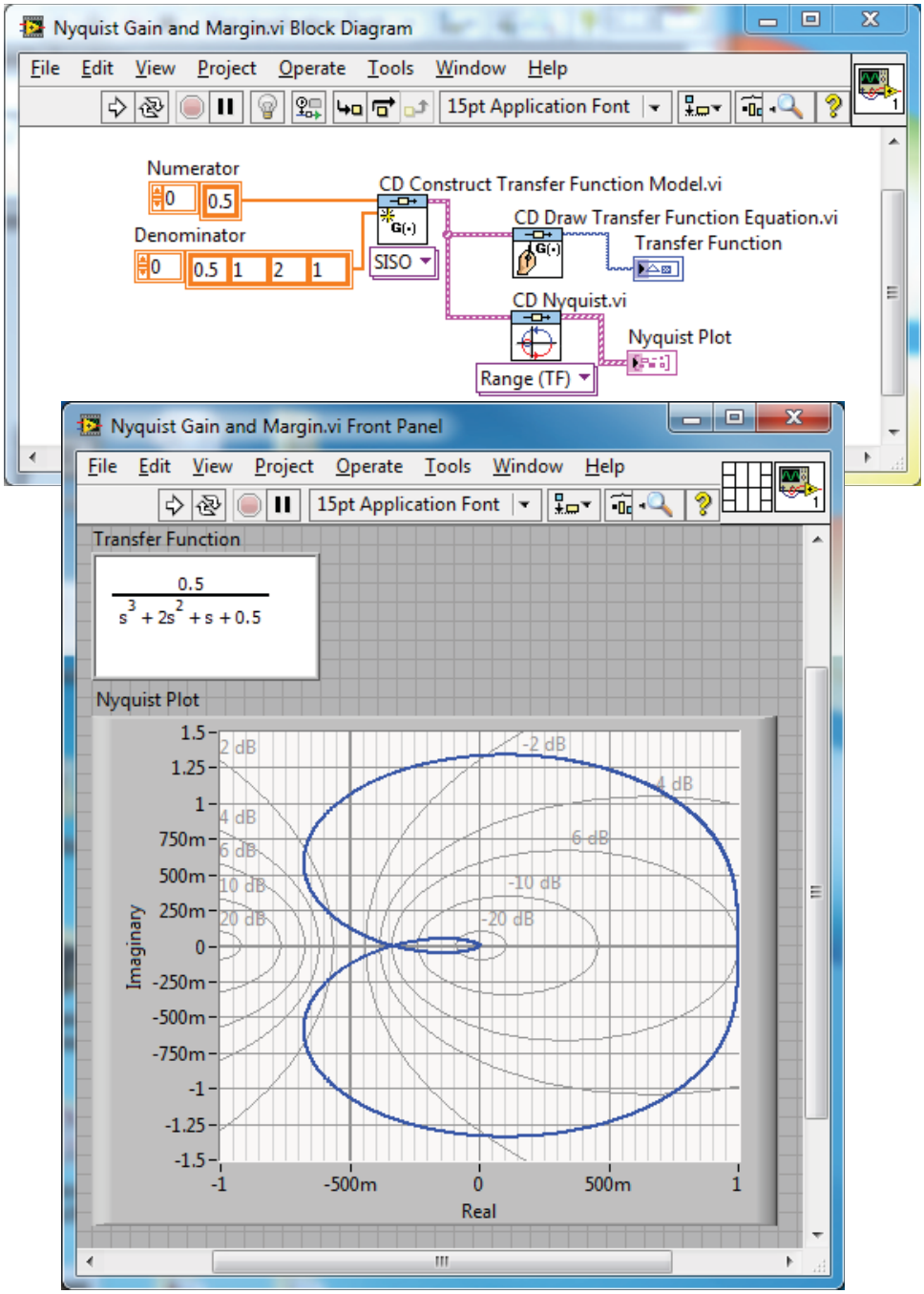


Figure 8.4: The Nyquist plot for the system in Fig. 8.2 with gain and phase margins.

8.2 Nichols Charts

Nichols charts can be generated using the CD Nichols function, shown in Fig. 8.5. The Nichols chart shown in Fig. 8.6 is for the system

$$G(s) = \frac{1}{0.2s^3 + 1.2s^2 + s}. \tag{8.1}$$

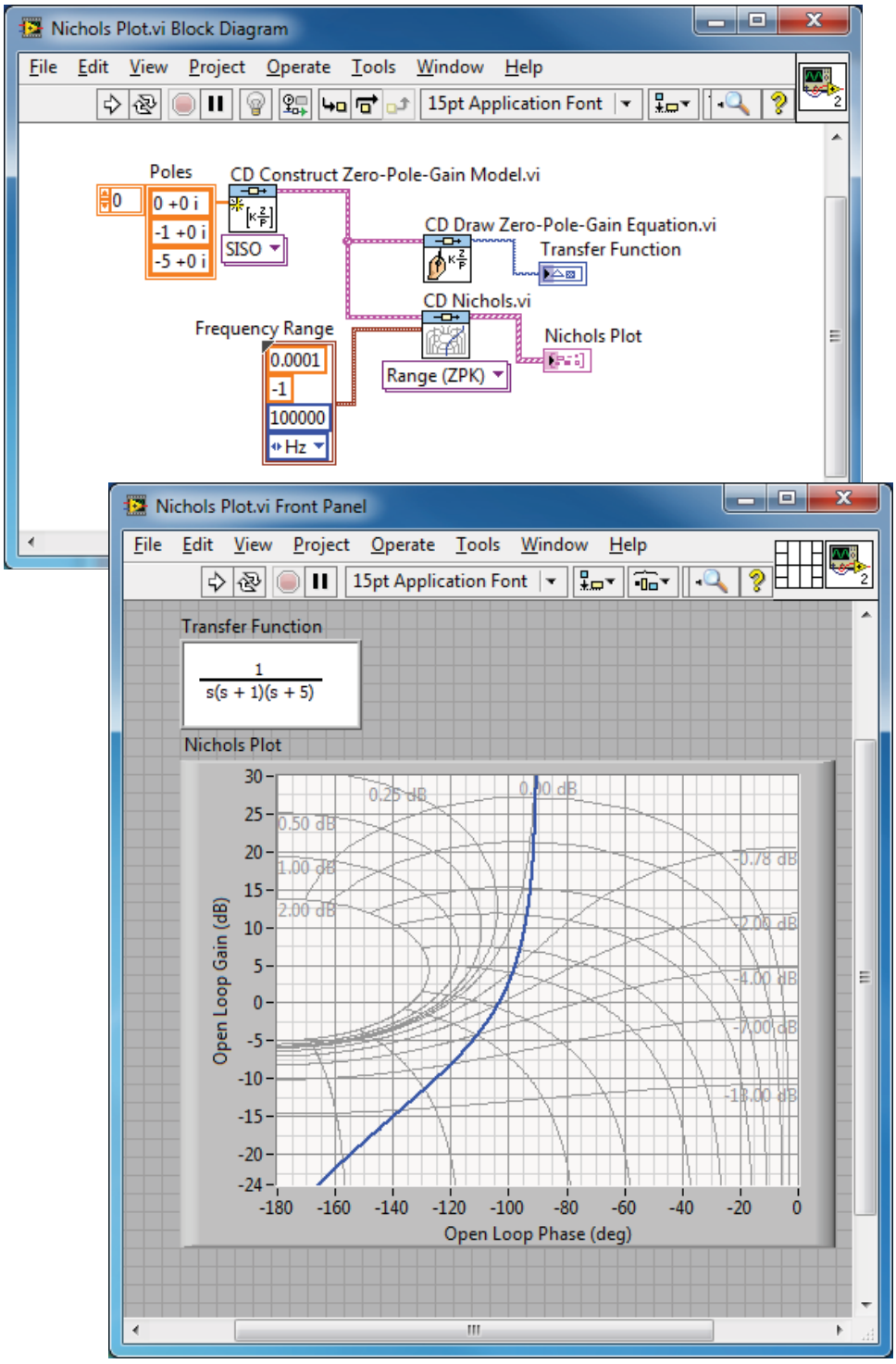


Figure 8.5: The CD Nichols function.

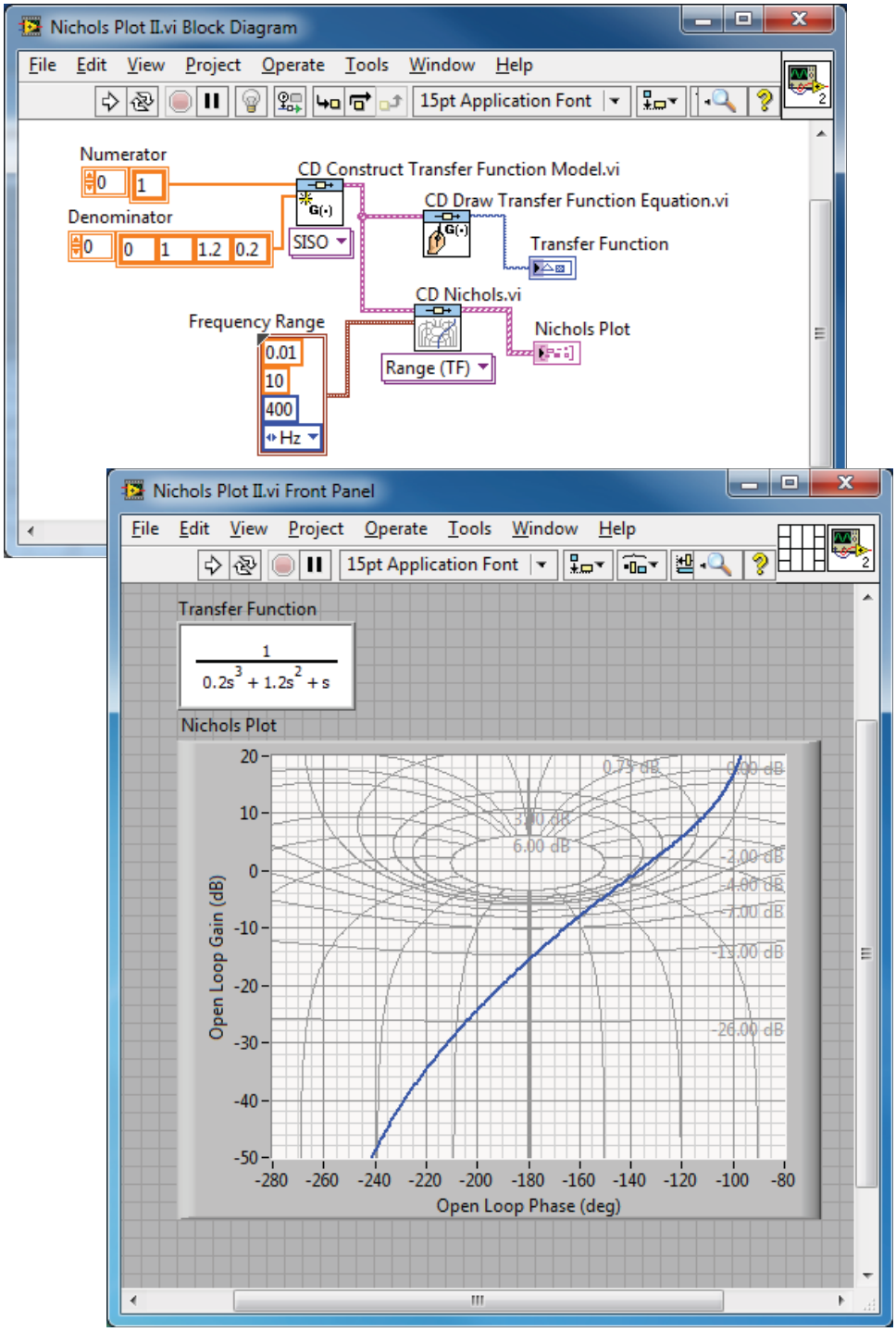


Figure 8.6: Nichols chart for the system in Eq. (8.1).

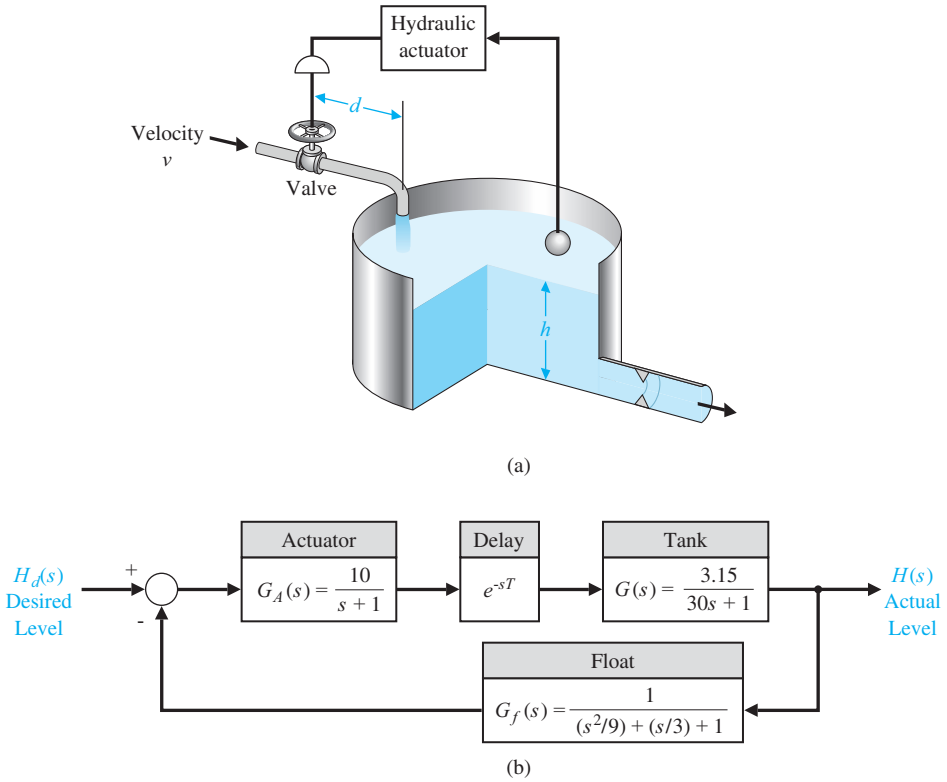


Figure 8.7: Liquid level control system block diagram.

Example 8.1 Liquid Level Control System

Consider a liquid level control system described by the block diagram shown in Fig. 8.7. Notice that this system has a time delay. The loop transfer function is given by

$$L(s) = \frac{31.5}{(s+1)(30s+1)(\frac{s^2}{9} + \frac{s}{3} + 1)} \exp^{-sT}. \tag{8.2}$$

We first rearrange Eq. (8.2) in such a way that $L(s)$ has a transfer function form with polynomials in the numerator and denominator. To do this we can make an approximation to e^{-sT} with the CD Construct Special TF Model function. The CD Construct Special TF Model function is shown in Fig. 8.8 with the time delay as an input. For example, suppose our time delay is $T = 1$ second and we want a second-order approximation $n = 2$. Then, using the CD Construct Special TF Model function we find that

$$e^{-s} \approx \frac{0.0743s^2 - 0.4460s + 0.8920}{0.0743s^2 + 0.4460s + 0.8920}. \tag{8.3}$$

Substituting Eq. (8.3) into Eq. (8.2), we have

$$L(s) = \frac{31.5(0.0743s^2 - 0.4460s + 0.8920)}{(s+1)(30s+1)(\frac{s^2}{9} + \frac{s}{3} + 1)(0.0743s^2 + 0.4460s + 0.8920)}.$$

Now we can build a script to investigate the relative stability of the system using the Bode diagram. Our goal is to have a phase margin of 30°. The associated script is shown in Fig. 8.8. To make the VI interactive, we let the gain K (now set at $K = 31.5$) be adjustable. We set K and run the VI to check the phase margin and iterate, if necessary. The final selected gain is $K = 16$. Remember that we have utilized a second-order Padé approximation of the time delay in our analysis. ◇

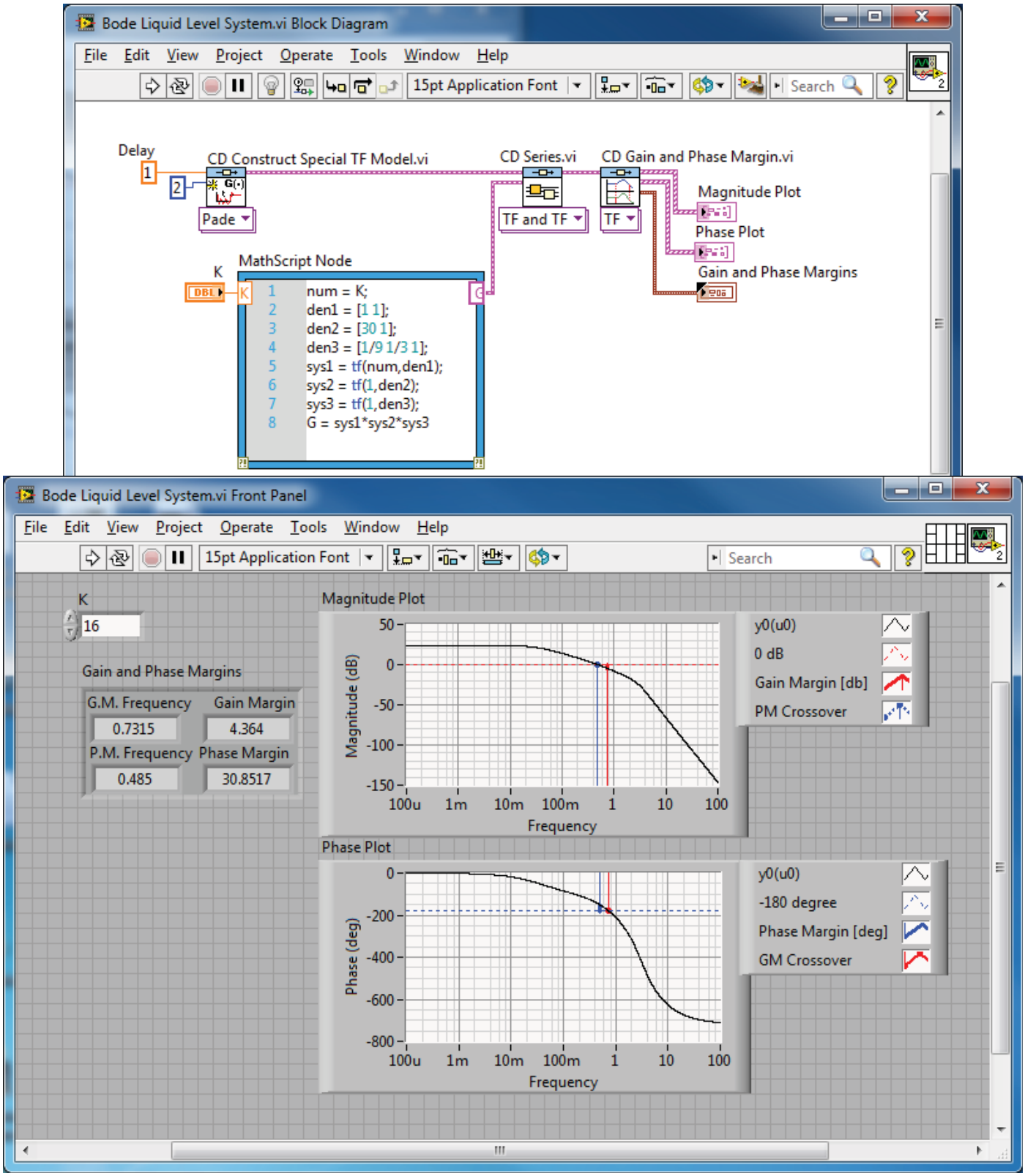
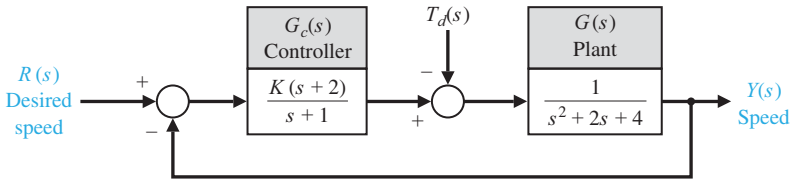


Figure 8.8: Bode diagram for the liquid level control system.

(a)



(b)

Figure 8.9: Remotely controlled reconnaissance vehicle's speed control system.

Example 8.2 Remotely Controlled Reconnaissance Vehicle

Consider the speed control system for a remotely controlled reconnaissance vehicle shown in Fig. 8.9. The design objective is to achieve good control with low steady-state error and low overshoot to a step command. Building a VI will allow us to perform many design iterations quickly and efficiently. First we investigate the steady-state error specification. The steady-state error, e_{ss}, to a unit step command is

$$e_{ss} = \frac{1}{1 + K/2}. \tag{8.4}$$

The effect of the gain K on the steady-state error is clear from Eq. (8.4): If $K = 20$, the error is 9% of the input magnitude; if $K = 10$, the error is 17% of the input magnitude. Now we can investigate the overshoot specification in the frequency domain. Suppose we require that the percent overshoot be less than 50%. Solving

$$P.O. \approx 100 \exp^{-\zeta\pi/\sqrt{1-\zeta^2}} \leq 50$$

for ζ yields $\zeta \geq 0.215$. For second-order systems we have the relationship between M_{p_ω} and ζ given by

$$M_{p_\omega} = \left(2\zeta\sqrt{1-\zeta^2}\right)^{-1}. \tag{8.5}$$

With $\zeta \geq 0.215$, we find that $M_{p_\omega} \leq 2.45$. We must keep in mind that the formula in Eq. (8.5) is exact for second-order systems only and can be used here only as a guideline. We now compute the closed-loop Bode diagram and check the values of M_{p_ω}. Any gain K for which $M_{p_\omega} \leq 2.45$ may be a valid gain for our design, but we will have to investigate further to include step responses to check the actual overshoot. The script in Fig. 8.10 aids us in this task. We can investigate further the gains $K = 20$, 10, and 4.44 (even though $M_{p_\omega} > 2.45$ for $K = 20$). We plot the step responses to quantify the overshoot as shown in Fig. 8.11. Additionally, we could have used a Nichols chart to aid the design process as shown in Fig. 8.12. The results of the analysis are summarized in Table 8.1 for $K = 20$, 10, and 4.44.

We choose $K = 10$ as our design gain. The gain margin is $GM = 49.56$ dB and the phase margin is $PM = 26.11°$. ◇

Table 8.1: Compensator Design Results

K	4.44	10	20
Percent overshoot	32.4	48.4	61.4
Settling time (seconds)	4.94	5.46	6.58
Peak time (seconds)	1.19	0.88	0.67
e_{ss}	31%	16.7%	9.1%

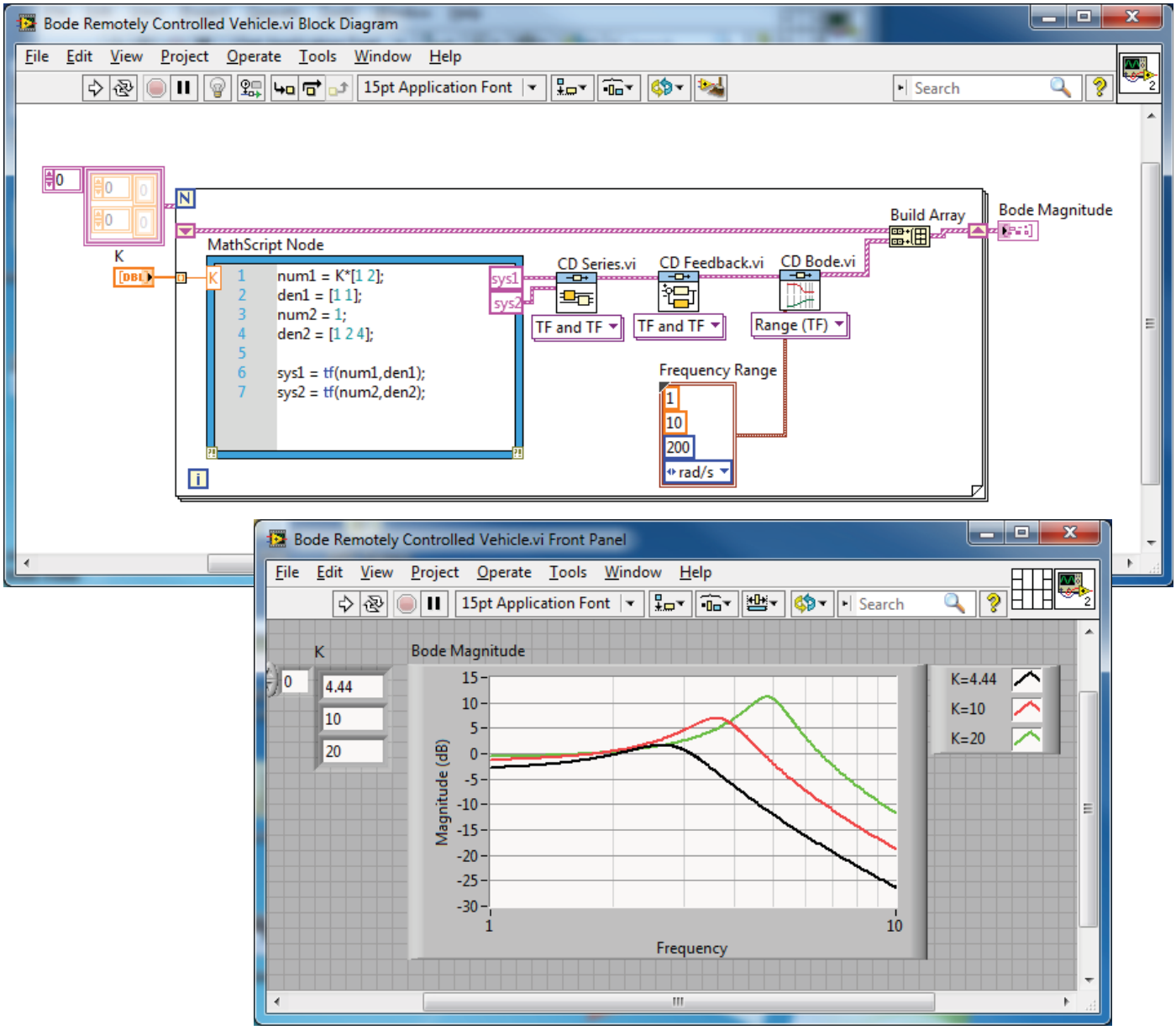


Figure 8.10: Remotely controlled vehicle closed-loop system Bode diagram

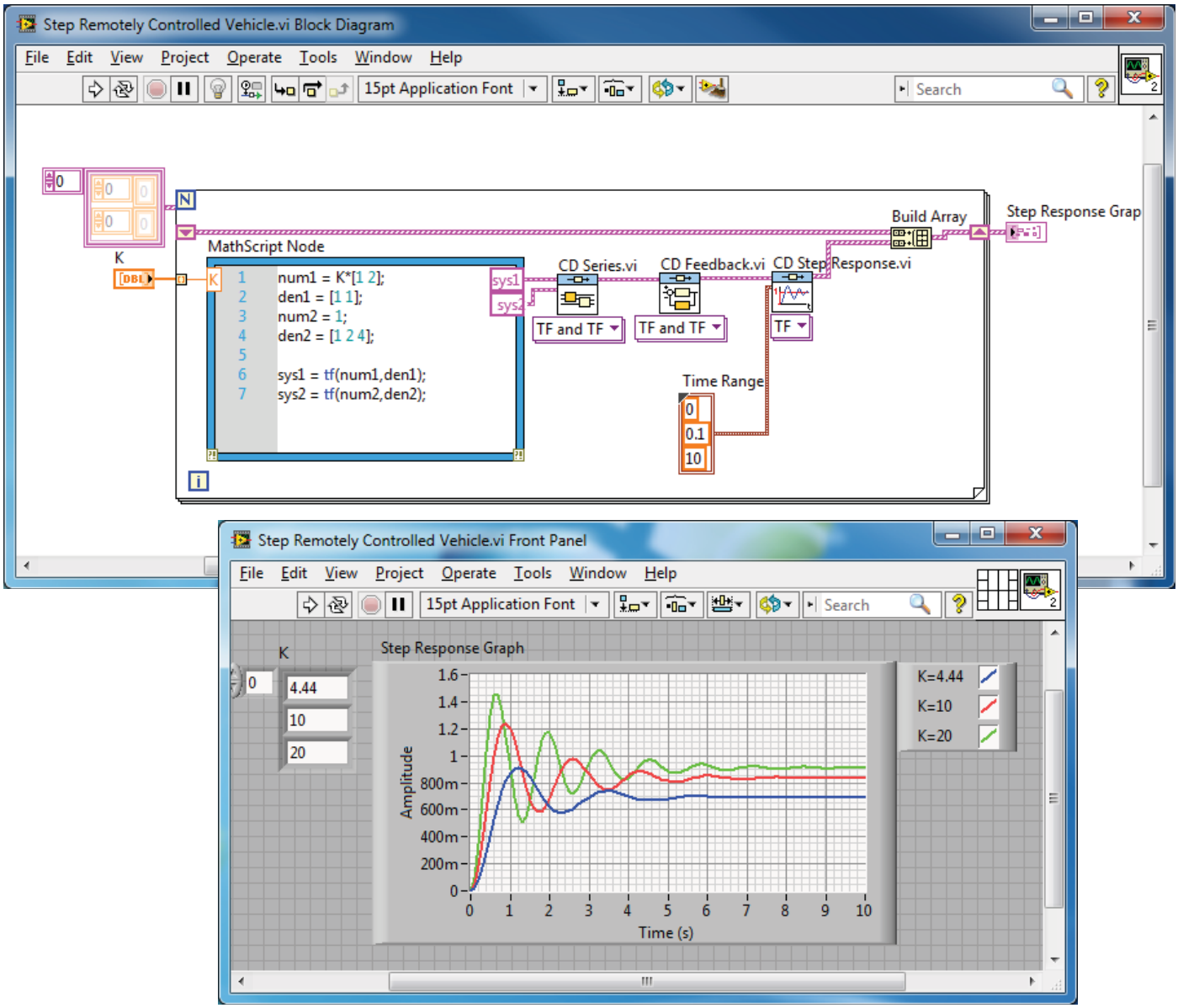


Figure 8.11: Remotely controlled vehicle step response.

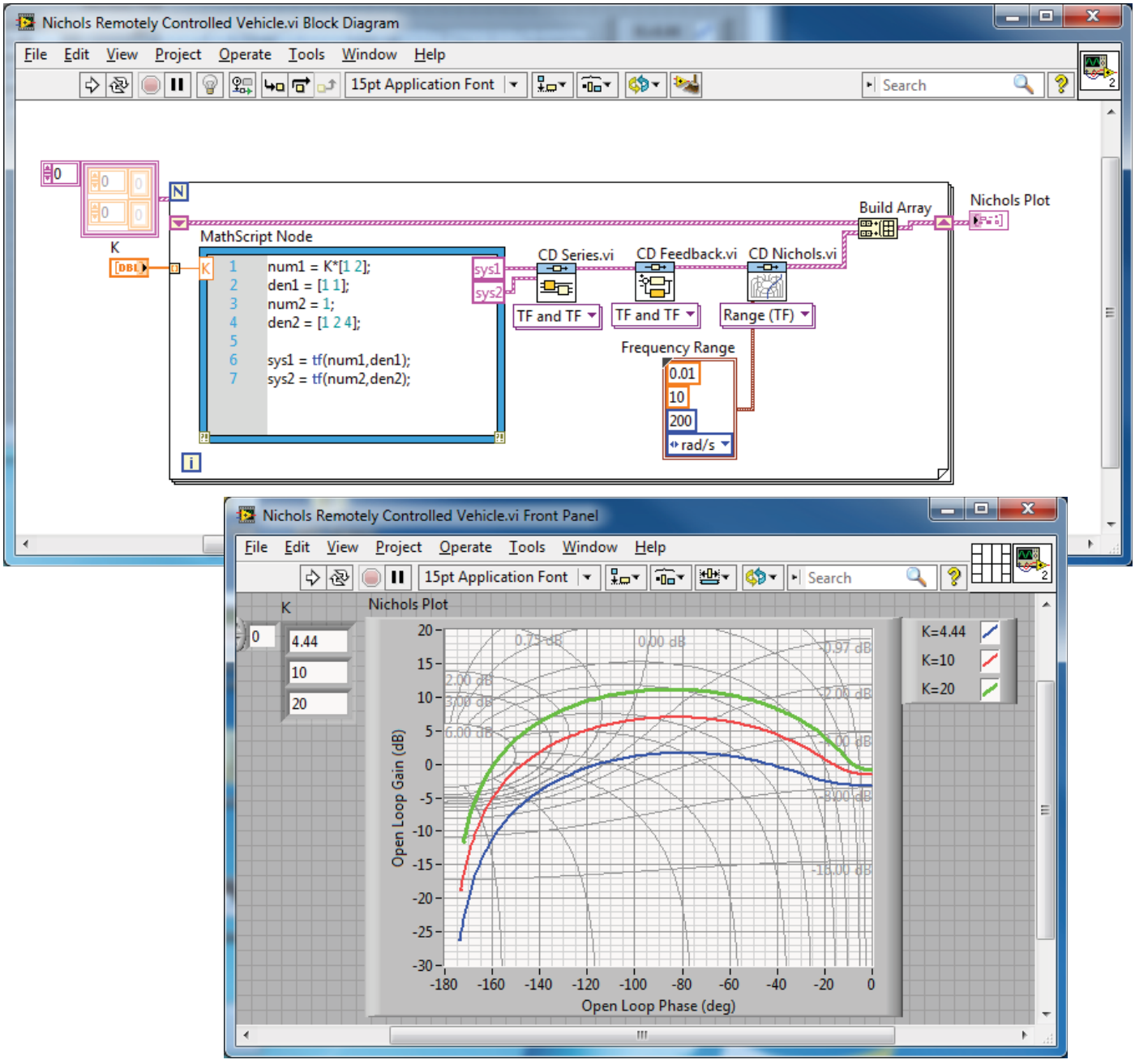


Figure 8.12: Remotely controlled vehicle Nichols chart.

CHAPTER 9

Design of Feedback Control Systems

In this chapter the compensation of control systems is illustrated using frequency response and root locus methods. We reconsider a rotor winder design example to illustrate the use of LabVIEW scripts in designing and developing control systems with good performance characteristics. Lead and lag compensators are examined for this design example and the system response is obtained using LabVIEW.

Example 9.1 Rotor Winder Control System

Consider the rotor winder control system shown in Fig. 9.1. The design objective is to achieve high steady-state accuracy to a ramp input. The steady-state error to a unit ramp input, $R(s) = 1/s^2$, is

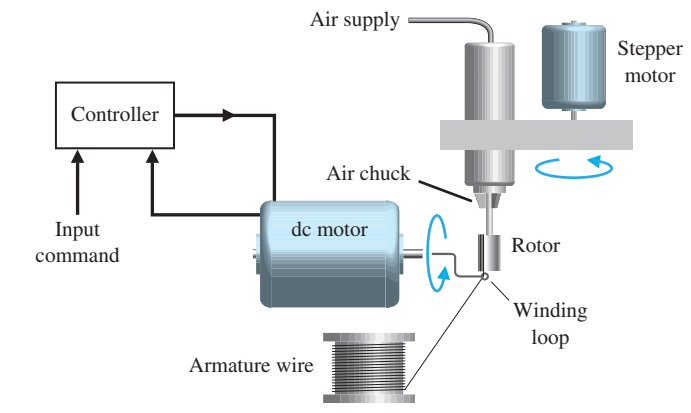
$$e_{ss} = \frac{1}{K_v},$$

where

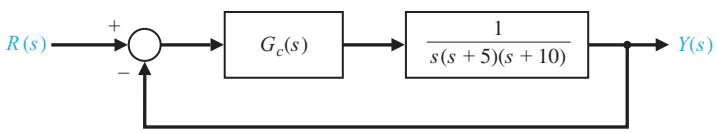
$$K_v = \lim_{s\to 0} s\frac{G_c(s)}{s(s+5)(s+10)} = \lim_{s\to 0}\frac{G_c(s)}{50}.$$

The performance specification of overshoot and settling time must be considered as well as steady-state tracking error. In all likelihood, a simple gain will not be satisfactory, so we will also consider controllers utilizing lead and lag compensators using both Bode diagram and root locus plot design methods. Our approach is to develop a series of LabVIEW scripts to aid in the compensator designs. Consider first a simple gain controller, $G_c(s)$, where

$$G_c(s) = K.$$



(a)



(b)

Figure 9.1: (a) Rotor winder control system. (b) Block diagram.

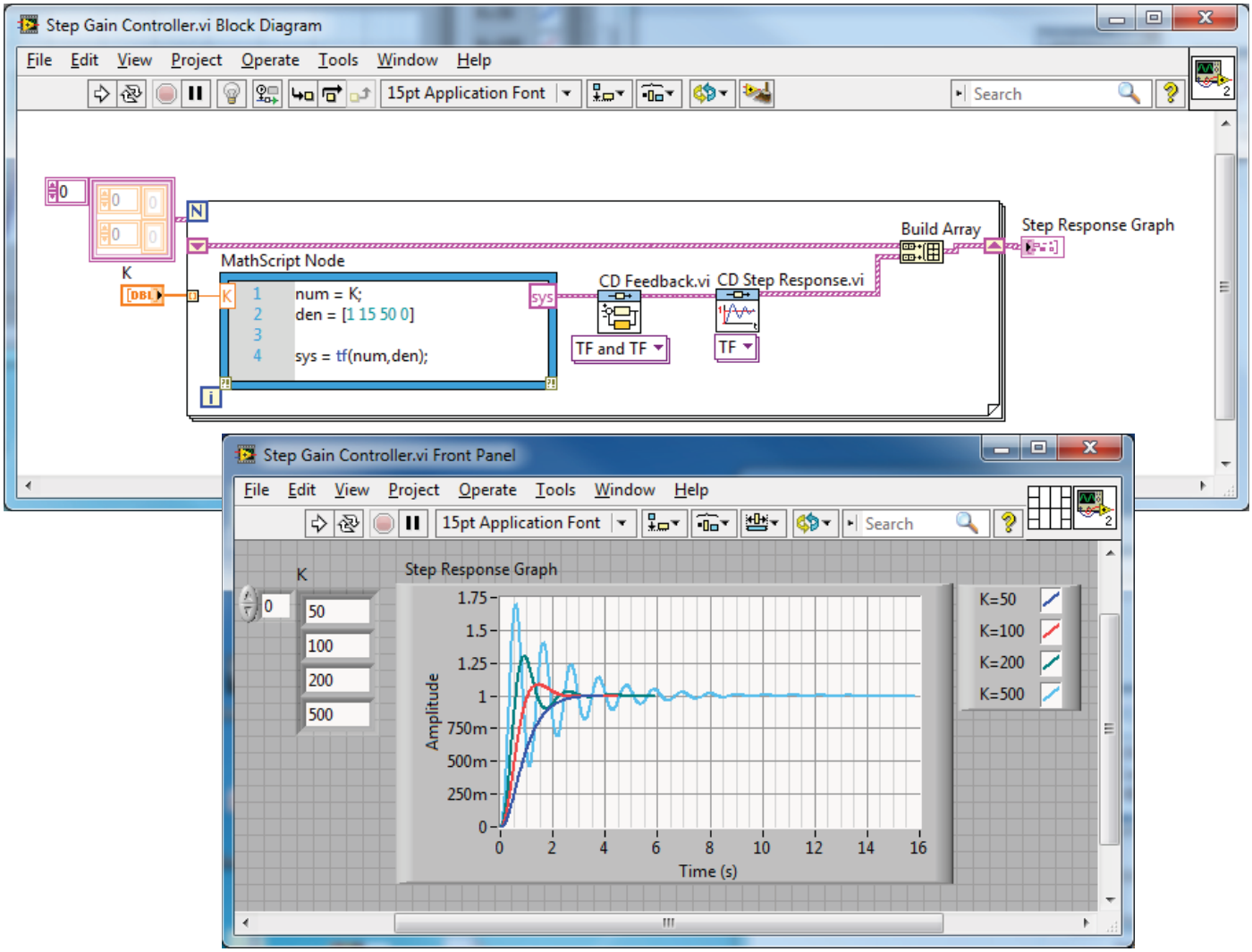


Figure 9.2: Transient response for simple gain controller.

Then, the steady-state error is

$$e_{ss} = \frac{50}{K}.$$

The larger we make K, the smaller is the steady-state error e_{ss}. However, we must also consider the effect that increasing K has on the transient response, as shown in Fig. 9.2. When $K = 500$, our steady-state error for a ramp is 10%, but the overshoot is 70% and the settling time is approximately 8 seconds for a step input. This is unacceptable performance and thus we turn to compensation. The two important compensator types considered are lead and lag compensators. First, consider the lead compensator

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

where $|z| < |p|$. The lead compensator provides the capability to improve the transient response. We will use a frequency-domain approach to design the lead compensator. We desire a steady-state error of less than 10% to a ramp input, or $K_v = 10$. In addition to the steady-state specifications, we desire to meet certain performance specifications: (1) settling time (2% criterion) $T_s \leq 3$ seconds, and (2) percent overshoot for a step input $P.O. \leq 10\%$. Solving for ζ and ω_n using

$$P.O. = 100\exp^{-\zeta\pi/\sqrt{1-\zeta^2}} = 10 \qquad \text{and} \qquad T_s = \frac{4}{\zeta\omega_n} = 3$$

yields $\zeta = 0.59$ and $\omega_n = 2.26$. We thus obtain the phase margin requirement:

$$\phi_{pm} \approx \frac{\zeta}{0.01} = 60^\circ.$$

The steps leading to the final design are as follows:

1. Obtain the uncompensated system Bode diagram with $K = 500$, and compute the phase margin.
2. Determine the amount of necessary phase lead.
3. Evaluate α from $\sin\phi_m = (\alpha - 1)/(\alpha + 1)$.
4. Compute $10\log\alpha$ and find the frequency ω_n on the uncompensated Bode diagram where the magnitude curve is equal to -10 $\log\alpha$.
5. In the neighborhood of ω_n on the uncompensated Bode, draw a line through the 0-dB point at ω_n with slope equal to the current slope plus 20 dB/decade. Locate the intersection of the line with the uncompensated Bode to determine the lead compensation zero location. Then calculate the lead compensator pole location as $p = \alpha z$.
6. Obtain the compensated Bode and check the phase margin. Repeat any steps if necessary.
7. Raise the gain to account for attenuation ($1/\alpha$).
8. Verify the final design with simulation using step functions, and repeat any steps if necessary.

Three scripts are utilized in the design. The design scripts are shown in Figs. 9.3–9.5. The first script is for the Bode plot of the uncompensated system, as shown in Fig. 9.3. The detailed Bode plot of the compensated system is shown in Fig. 9.4. Figure 9.5

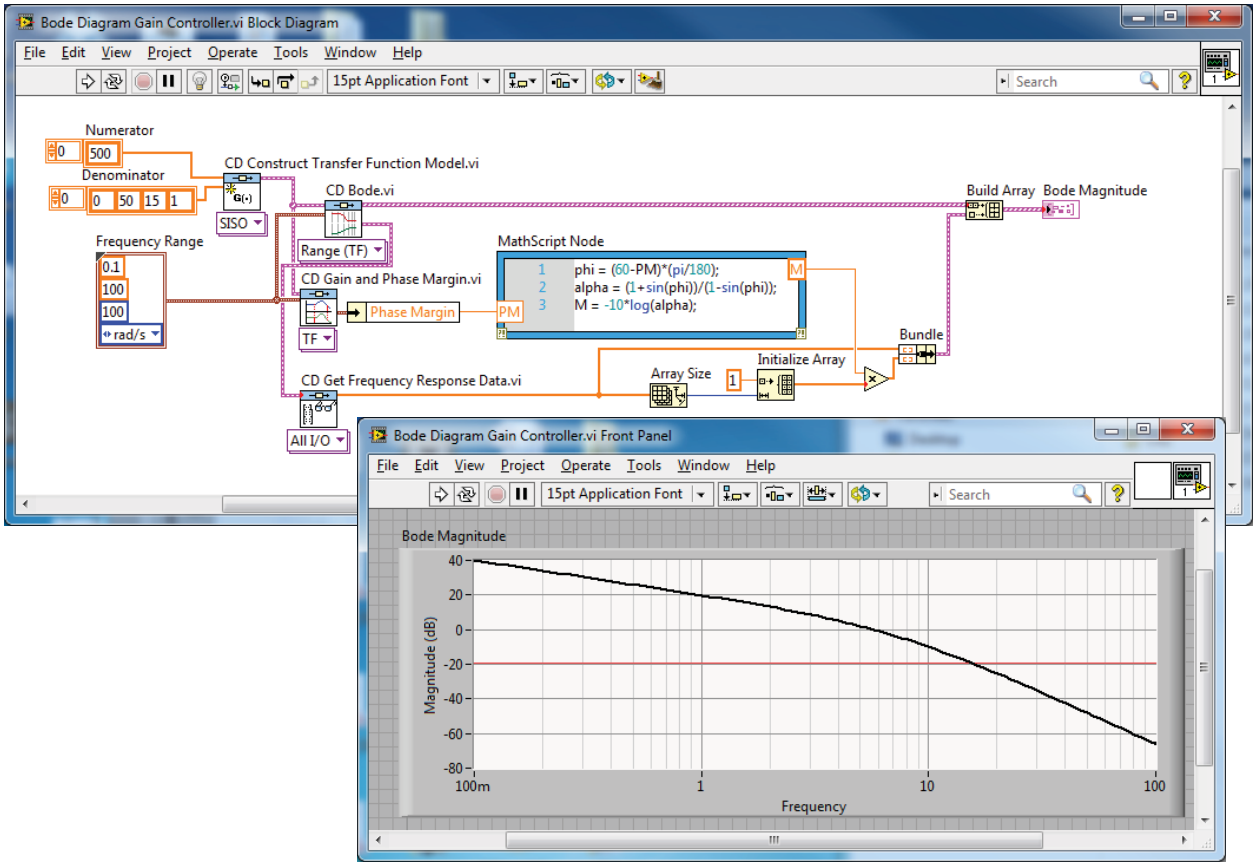


Figure 9.3: Bode diagram.

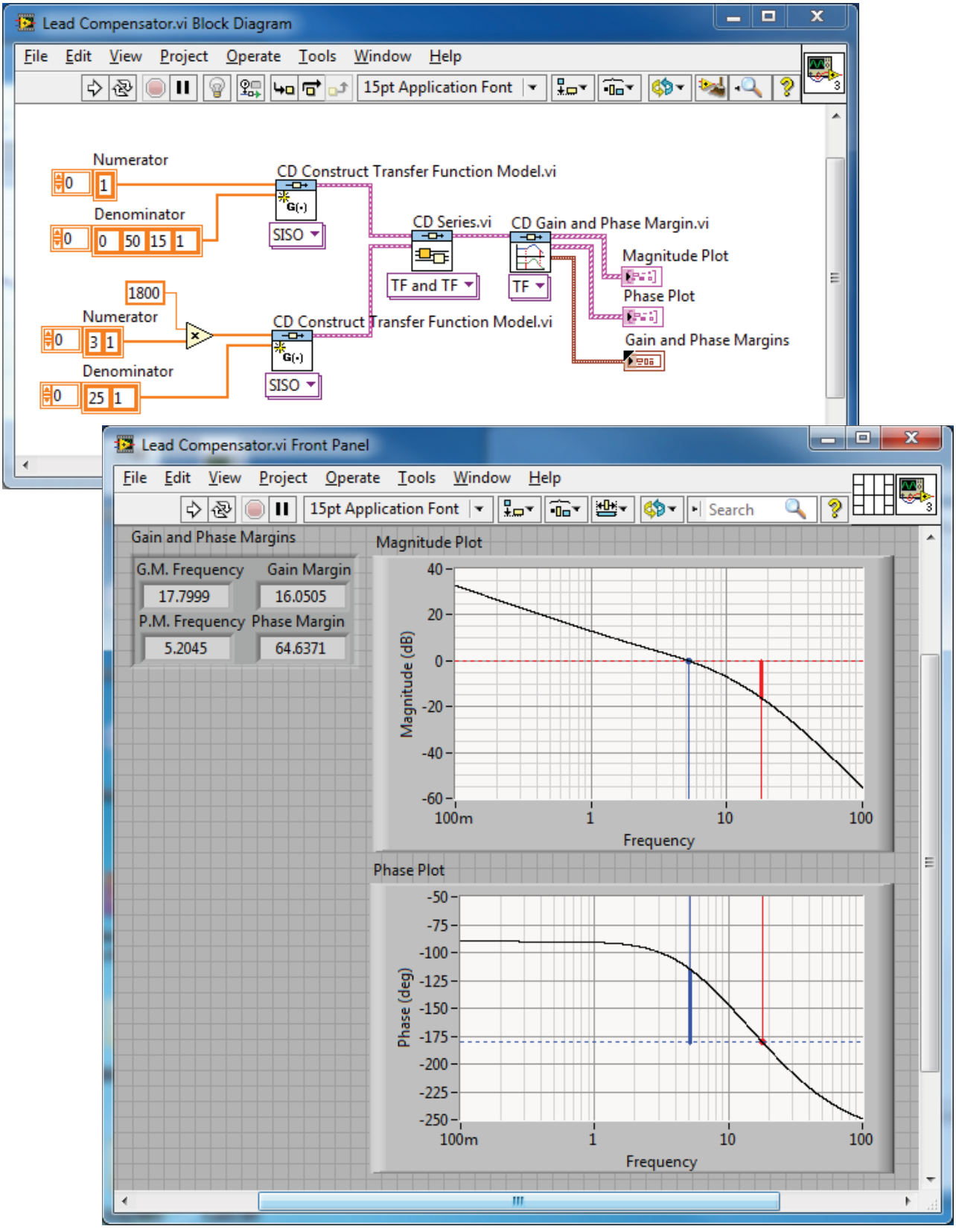


Figure 9.4: Lead compensator Bode diagram.

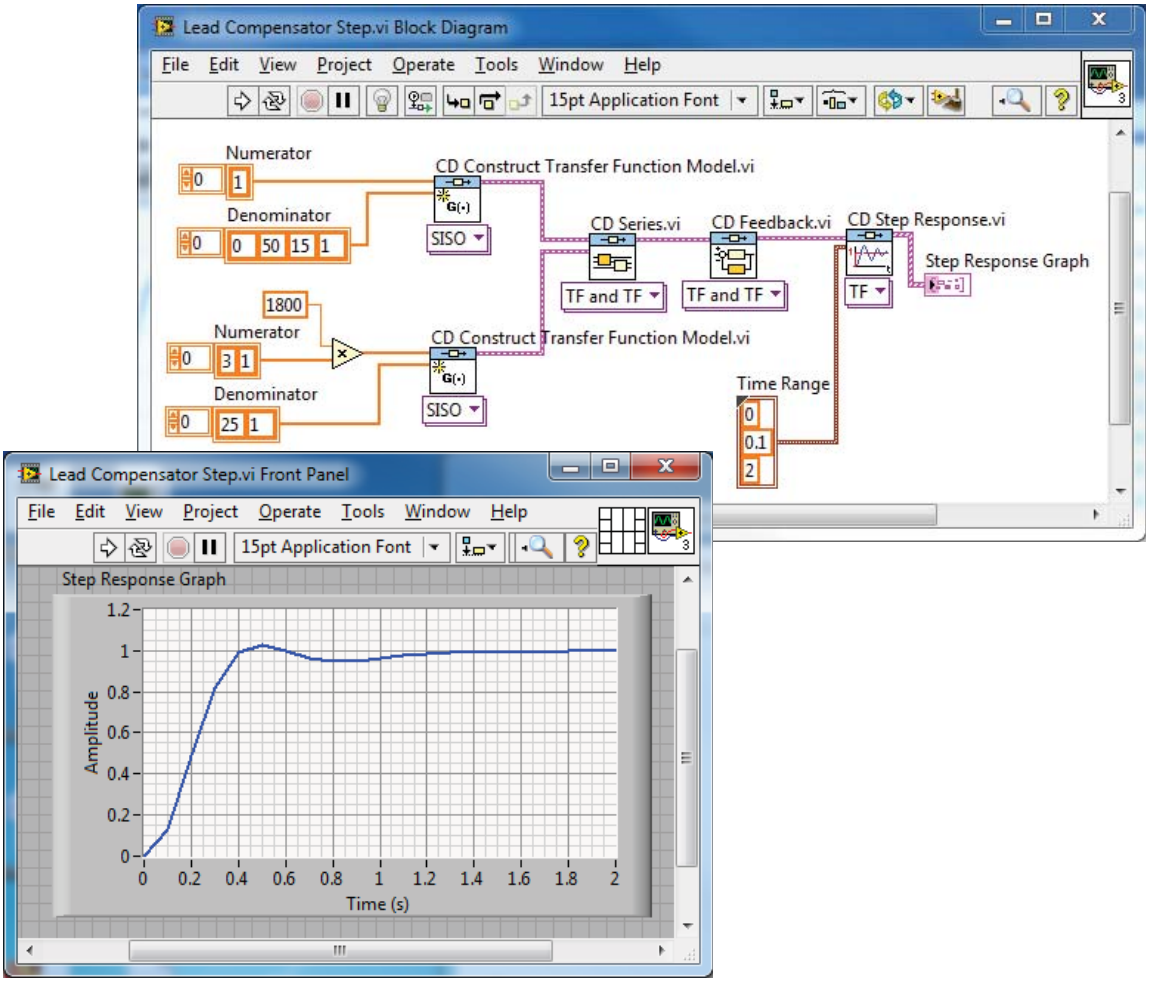


Figure 9.5: Lead compensator step response.

displays the LabVIEW script for the step response analysis. The final lead compensator design is

$$G_c(s) = \frac{1800(s+3.5)}{(s+25)},$$

where $K = 1800$ was selected after iteratively using the LabVIEW script. The settling time and overshoot specifications are satisfied, but $K_v = 5$, resulting in a 20% steady-state error to a ramp input. It is possible to continue the design iteration and refine the compensator somewhat, although it should be clear that the lead compensator has added phase margin and improved the transient response as anticipated. To reduce the steady-state error, we can consider the lag compensator, which has the form

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

where $|p| < |z|$. We will use a root locus approach to design the lag compensator, although it can be done using Bode as well. The desired root location region of the dominant roots is specified by $\zeta = 0.59$ and $\omega_n = 2.26$. The steps in the design are as follows:

1. Obtain the root locus of the uncompensated system.
2. Locate suitable root locations on the uncompensated system that lie in the region defined by $\zeta = 0.59$ and $\omega_n = 2.26$.
3. Calculate the loop gain at the desired root location and the system error constant, $K_{v_{uncomp}}$.
4. Compute $\alpha = K_{v_{comp}}/K_{v_{uncomp}}$ where $K_{v_{comp}} = 10$.

5. With a known, determine suitable locations of the compensator pole and zero so that the compensated root locus still passes through desired location.

6. Verify with simulation and repeat any steps if necessary.

The design methodology is illustrated in Figs. 9.6–9.8. We can compute the gain K associated with the roots of our choice on the uncompensated root locus that lie in the performance region. We then compute α to ensure that we achieve the desired K_v. We place the lag compensator pole and zero to avoid impacting the uncompensated root locus. In Fig. 9.7, the lag compensator pole and zero are very near the origin, at $z = -0.1$ and $p = -0.01$. The settling time and overshoot specifications are nearly satisfied, and $K_v = 10$, as desired. It is possible to continue the design iteration and refine the compensator somewhat, although it should be clear that the lag compensator has improved the steady-state errors to a ramp input relative to the lead compensator design. The final lag compensator design is

$$G_c(s) = \frac{100(s + 0.1)}{(s + 0.01)}.$$

◇

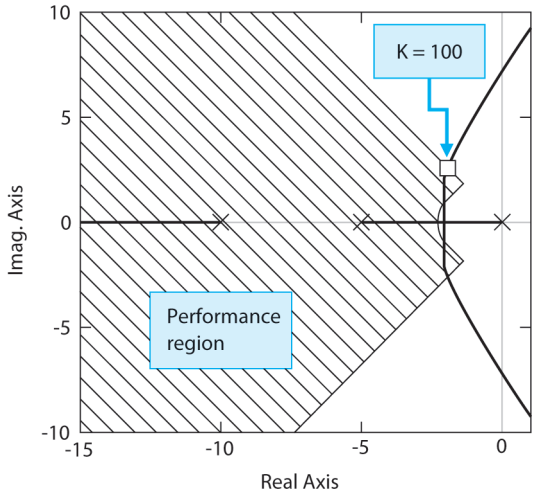


Figure 9.6: Lag compensator: uncompensated root locus.

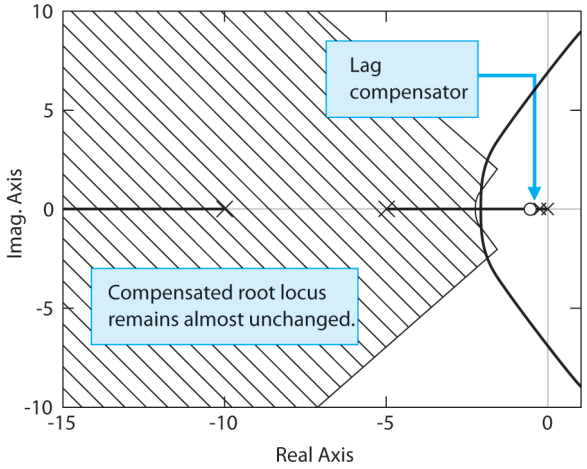


Figure 9.7: Lag compensator: compensated root locus.

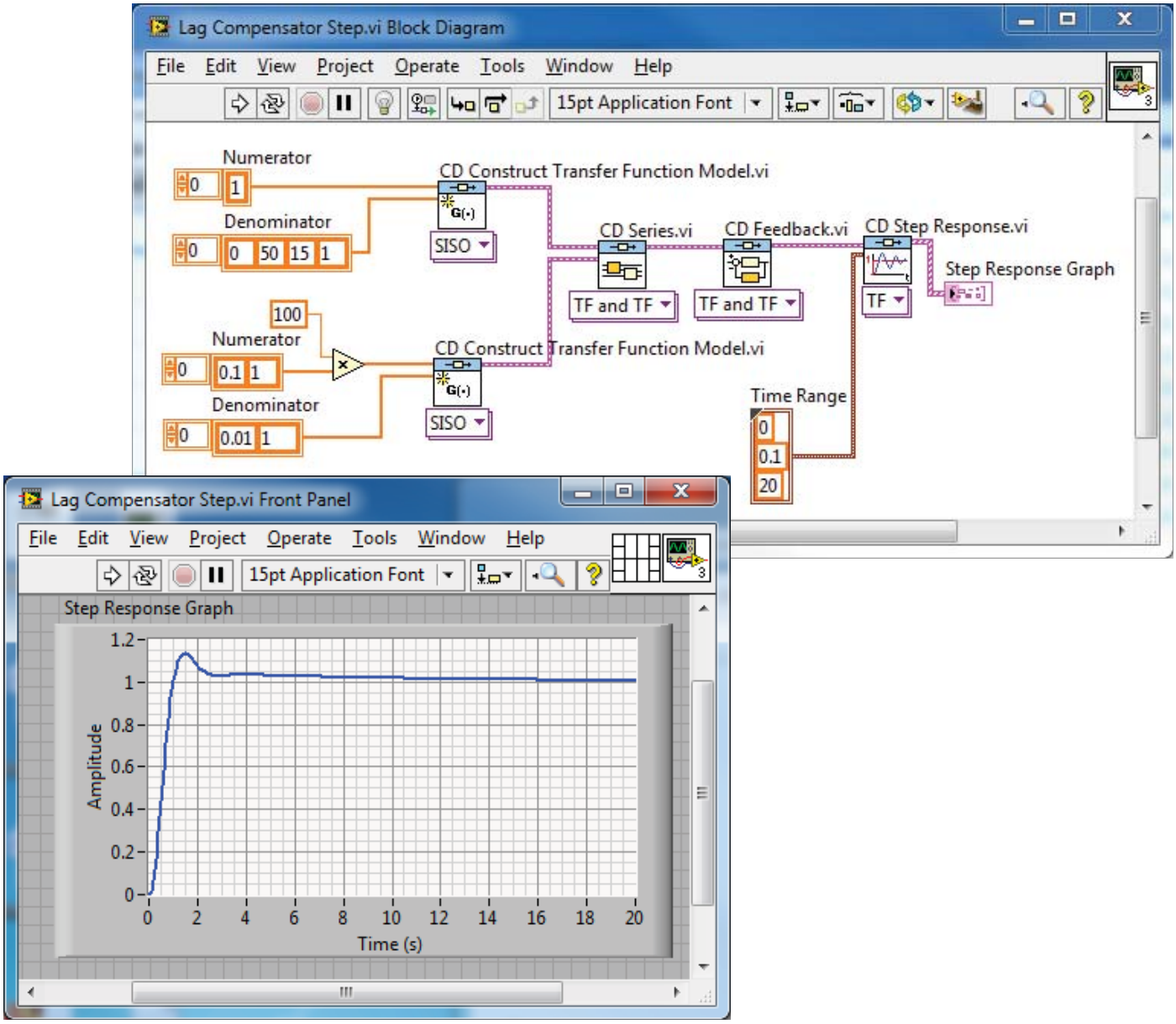


Figure 9.8: Lag compensator: Step response.

CHAPTER 10

Design of State Variable Feedback Systems

Controllability and observability of a system in state variable feedback form can be checked using the LabVIEW functions CD Controllability Matrix and CD Observability Matrix, respectively. The inputs to the CD Controllability Matrix function are the system matrix **A** and the input matrix **B**; the output of CD Controllability Matrix is the controllability matrix $\mathbf{P}_c$. Similarly, the input to the CD Observability Matrix function is the system matrix **A** and the output matrix **C** and the output of CD Observability Matrix is the observability matrix $\mathbf{P}_o$. Note that the controllability matrix, $\mathbf{P}_c$, is a function only of **A** and **B**, while the observability matrix, $\mathbf{P}_o$, is a function only of **A** and **C**.

Example 10.1 Satellite Trajectory Control

Consider a satellite in a circular, equatorial orbit at an altitude of 250 nautical miles above the earth, as illustrated in Fig. 10.1. The satellite motion (in the orbit plane) is described by the normalized state variable model

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 3\omega^2 & 0 & 0 & 2\omega \\ 0 & 0 & 0 & 1 \\ 0 & -2\omega & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} u_r + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_t, \tag{10.1}$$

where the state vector **x** represents normalized perturbations from the circular, equatorial orbit, u_r is the input from a radial thruster, u_t is the input from a tangential thruster, and $\omega = 0.0011$ rad/s (approximately one orbit of 90 minutes) is the orbital rate for the

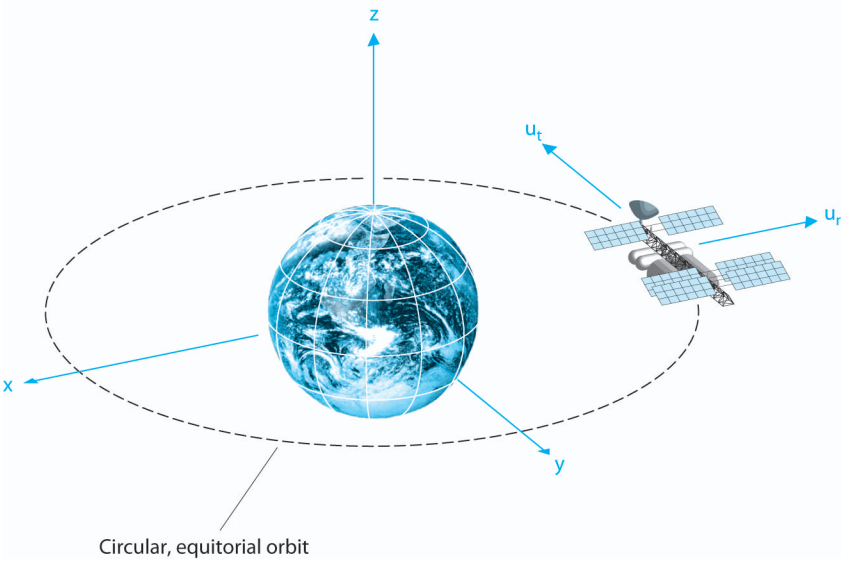


Figure 10.1: The satellite in an equatorial, circular orbit.

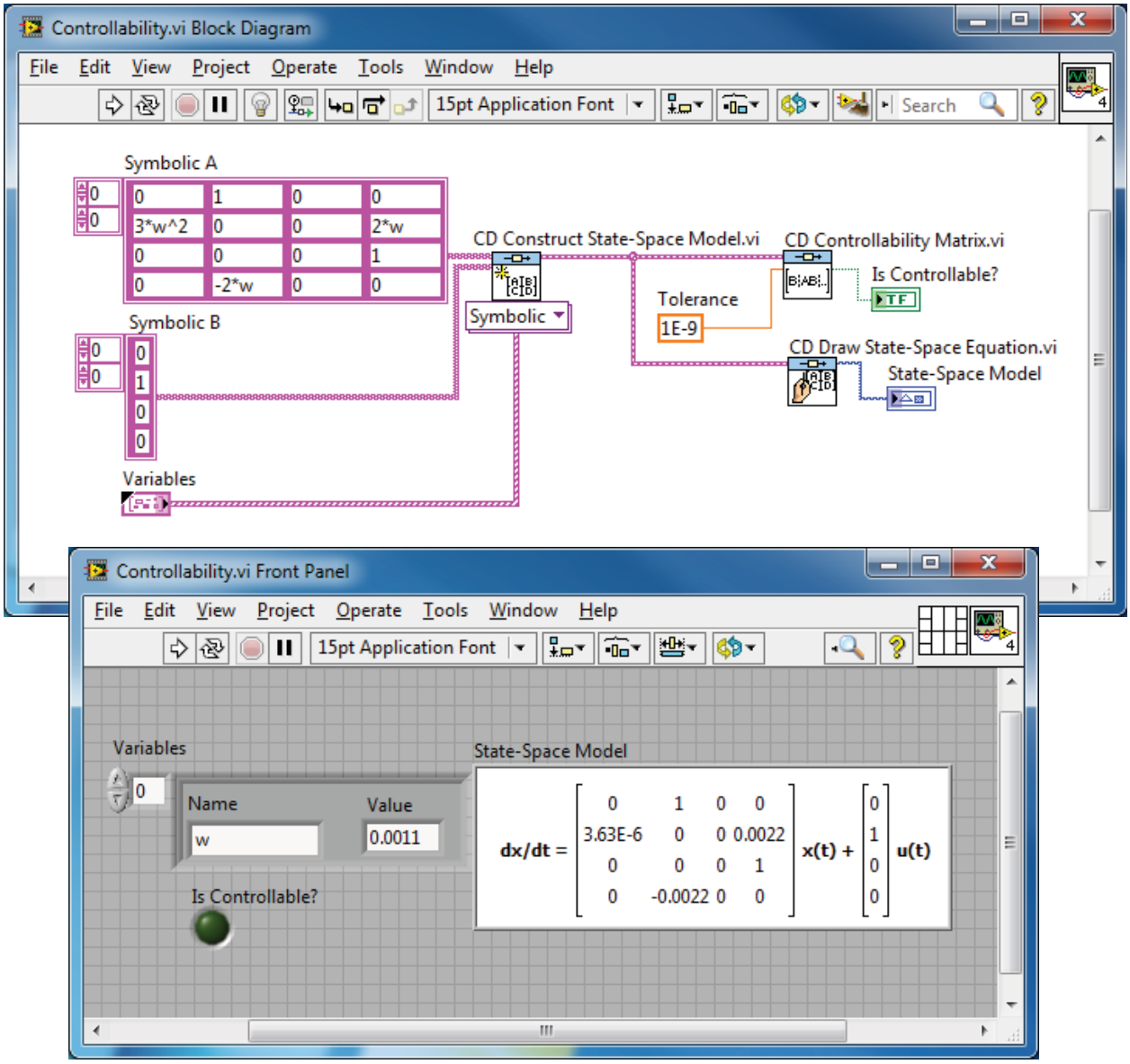


Figure 10.2: Controllability with radial thrusters only.

satellite at the specific altitude. In the absence of perturbations, the satellite will remain in the nominal circular, equatorial orbit. However, disturbances such as aerodynamic drag can cause the satellite to deviate from its nominal path. The problem is to design a controller that commands the satellite thrusters in such a manner that the actual orbit remains near the desired circular orbit. Before commencing with the design, we check controllability. In this case we investigate controllability using the radial and tangential thrusters independently. Suppose the tangential thruster fails (i.e., $u_t = 0$), and only the radial thruster is operational. Is the satellite controllable from u_r only? We answer this question by using LabVIEW to determine the controllability. Using the script shown in Fig. 10.2, we find that the determinant $\mathbf{P}_c$ is zero; thus the satellite is not completely controllable when the tangential thruster fails.

Suppose now that the radial thruster fails (i.e., $u_r = 0$) and that the tangential thruster is functioning properly. Is the satellite controllable from u_t only? Using the script in Fig. 10.3, we find that the satellite is completely controllable using the tangential thruster only. ◇

We conclude this section with a controller design for an automatic test system using state variable models. The design approach utilizes root locus methods and incorporates LabVIEW scripts to assist in the procedure.

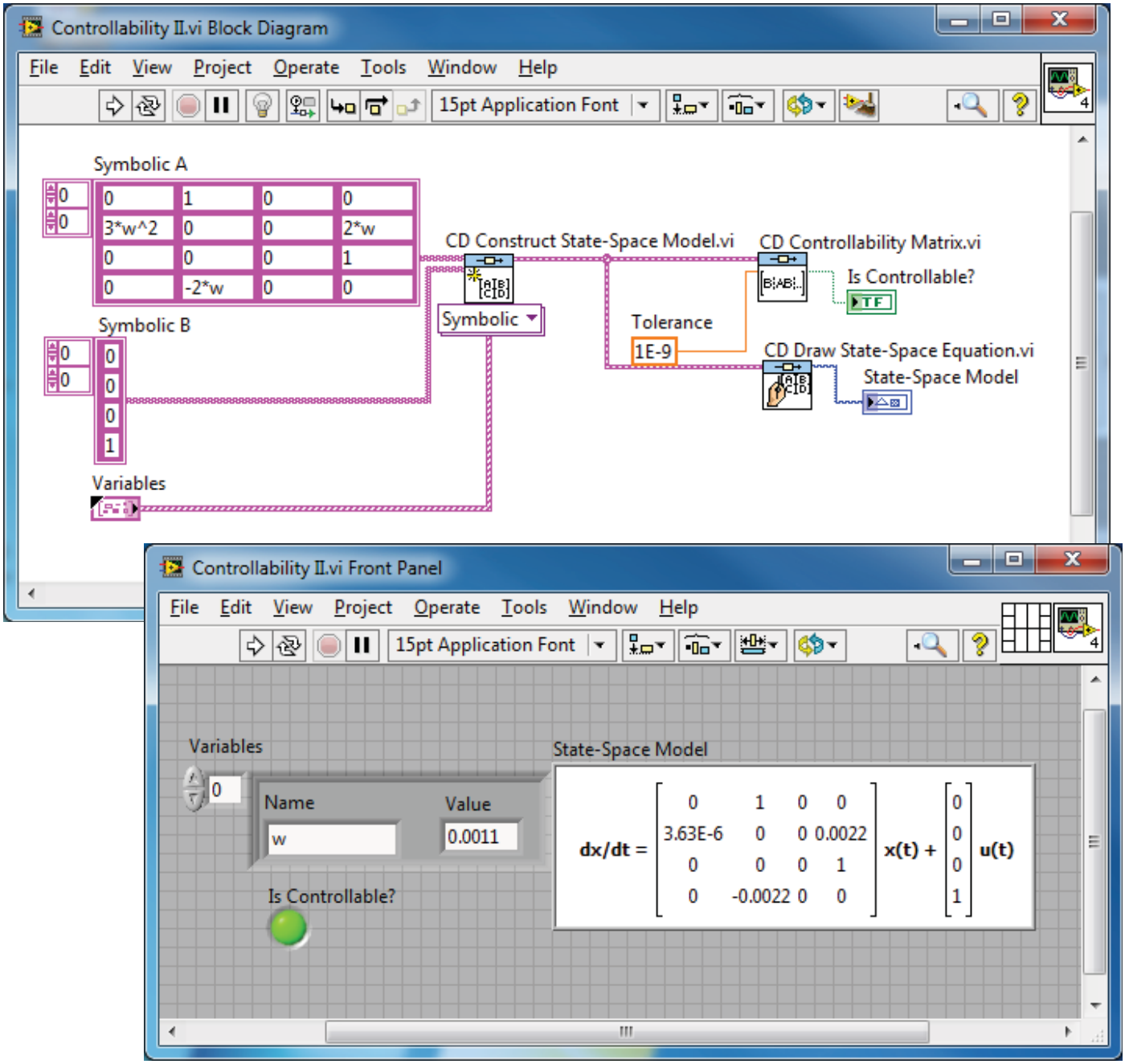


Figure 10.3: Controllability with tangential thrusters only.

Example 10.2 Automatic Test System

The state-space representation for an automatic test system is

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u, \tag{10.2}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -5 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ K \end{bmatrix}.$$

Our design specifications are a step response with (1) a settling time (2% criterion) less than $T_s \leq 2$ seconds and (2) an overshoot

less than $P.O. \leq 4\%$. We assume that the state variables are available for feedback, so that the control is given by

$$u = (-K_1 \ -K_2 \ -K_3)\,\mathbf{x} = \mathbf{K}\mathbf{x}. \tag{10.3}$$

We must select the gains K, K_1, K_2, and K_3 to meet the performance specifications. Using the design approximations

$$P.O. = 100\exp^{-\zeta\pi/\sqrt{1-\zeta^2}} < 4 \qquad \text{and} \qquad T_s = \frac{4}{\zeta\omega_n} < 2$$

we find that

$$\zeta > 0.72 \qquad \text{and} \qquad \omega_n > 2.8.$$

This defines a region in the complex plane in which our dominant roots must lie, so that we expect to meet the design specifications, as shown in Fig. 10.4. Substituting Eq. (10.3) into Eq. (10.2) yields

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 1 \\ -KK_1 & -KK_2 & -(5+KK_3) \end{bmatrix} \mathbf{x} = \mathbf{H}\mathbf{x}, \tag{10.4}$$

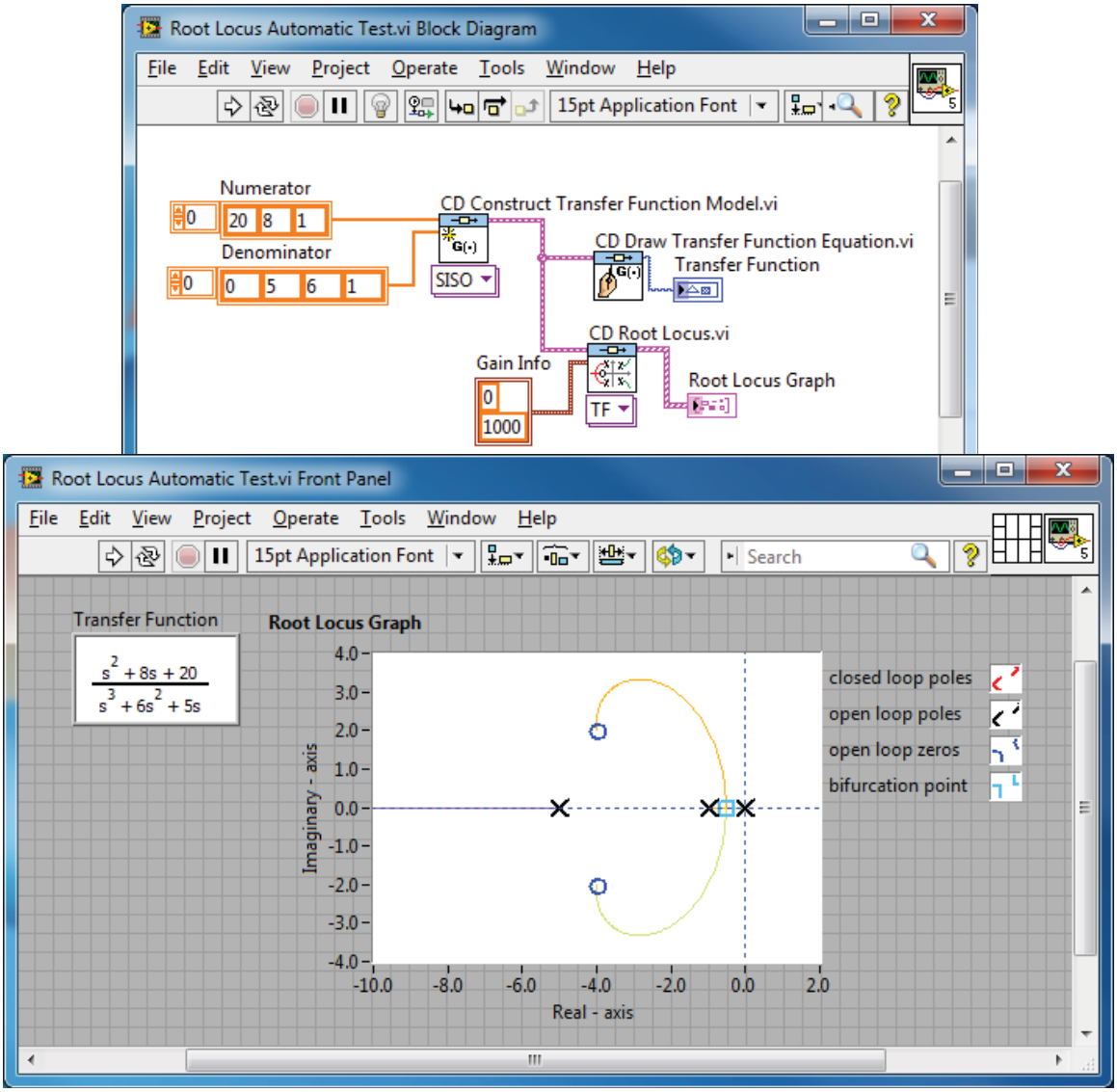


Figure 10.4: Root locus for the automatic test system.

where $\mathbf{H} = \mathbf{A} - \mathbf{BK}$. The characteristic equation associated with Eq. (10.4) can be obtained by evaluating $\det(s\mathbf{I} - \mathbf{H}) = 0$, resulting in

$$s(s+1)(s+5) + KK_3\left(s^2 + \frac{K_3 + K_2}{K_3}s + \frac{K_1}{K_3}\right) = 0. \tag{10.5}$$

If we view KK_3 as a parameter and let $K_1 = 1$, then we can write Eq. (10.5) as

$$1 + KK_3\left(\frac{s^2 + \frac{K_3+K_2}{K_3}s + \frac{1}{K_3}}{s(s+1)(s+5)}\right) = 0.$$

We place the zeros at $s = -4 \pm 2j$ in order to pull the locus to the left in the s-plane. Comparing the coefficients of

$$s^2 + 8s + 20 = s^2 + \frac{K_3 + K_2}{K_3}s + \frac{1}{K_3}$$

leads to

$$\frac{K_3 + K_2}{K_3} = 8$$

and

$$\frac{1}{K_3} = 20.$$

Therefore $K_2 = 0.35$ and $K_3 = 0.05$. We can now plot a root locus with KK_3 as the parameter, as shown in Fig. 10.4. The characteristic equation, Eq. (10.5), is

$$1 + KK_3\left(\frac{s^2 + 8s + 20}{s(s+1)(s+5)}\right) = 0.$$

The roots for the selected gain, $KK_3 = 12$, lie in the performance region, as shown in Fig. 10.4. The final gains are

$$\begin{aligned} K &= 240, \\ K_1 &= 1, \\ K_2 &= 0.35, \\ K_3 &= 0.05. \end{aligned}$$

The controller design results in a settling time of about 1.8 seconds and an overshoot of 3%, as shown in Fig. 10.5. ◇

Example 10.3 State-Space System Simulation using LabVIEW

Consider the system given in state variable form

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{B}u, \tag{10.6}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \text{and} \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

In the LabVIEW simulation, you will want to utilize full-state feedback, which means that the entire state vector (in this case, x_1 and x_2) needs to be available for feedback. One way to accomplish this is to choose the output matrices $\mathbf{C}$ and $\mathbf{D}$ such that $\mathbf{y} = \mathbf{x}$, thus

$$\mathbf{y} = \mathbf{Cx} + \mathbf{D}u.$$

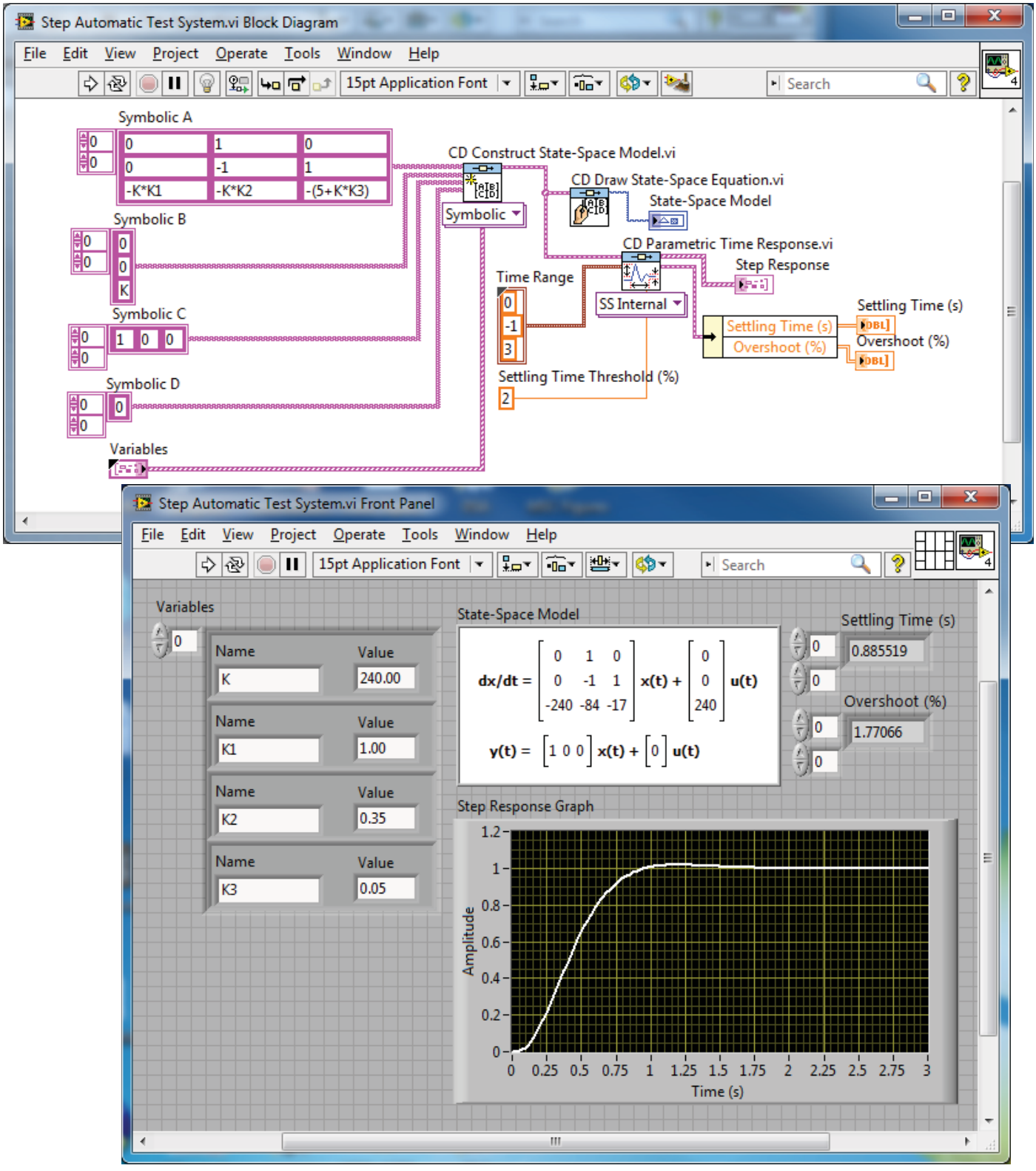


Figure 10.5: Step response for the automatic test system.

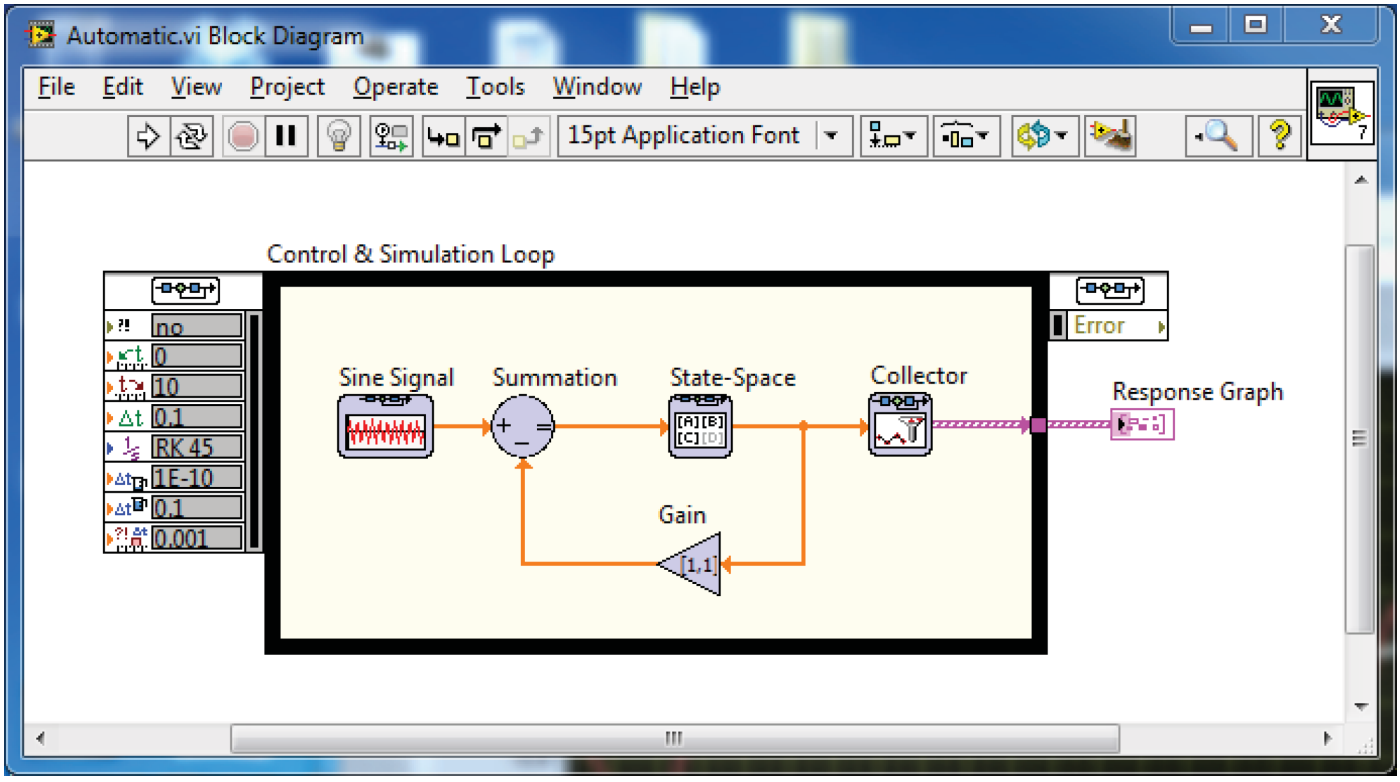


Figure 10.6: LabVIEW block diagram for a state-space system with full-state feedback.

where

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The full-state feedback control law is

$$u = -\mathbf{K}\mathbf{x} + v,$$

where v is a reference input. The LabVIEW block diagram of the closed-loop system is shown in Fig. 10.6.

The reference input is supplied by the Sine Wave function. The outputs of the state-space block are the two states x_1 and x_2. These states are fed back to the feedback gain matrix block to obtain the feedback signal $\mathbf{K}\mathbf{x}$. The reference input signal v is then summed with the (negative) feedback signal forming the plant input signal $u = -\mathbf{K}\mathbf{x} + v$. You can adjust the sine wave signal, as shown in Fig. 10.7. The gain matrix $\mathbf{K}$ and the system matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are set as illustrated in Figs. 10.8 and 10.9, respectively. The feedback gain matrix is chosen as

$$\mathbf{K} = [1\ 1].$$

This places the closed-loop poles (that is, the eigenvalues of $\mathbf{A} - \mathbf{B}\mathbf{K}$ at $s_1 = -1$ and $s_2 = -3$. Using LabVIEW allows you easy access to the feedback gain matrix so that simulation studies using varying feedback gain sets can be readily performed. Notice in Fig. 10.9 that the initial conditions are inputs. In the simulation presented here, the initial conditions are chosen to be

$$\mathbf{x}(0) = \begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Once you have defined all the parameters, you can initiate the simulation. The resulting state variable time histories are shown in Fig. 10.10. You can use the LabVIEW simulation to conduct numerical experiments easily and to analyze the closed-loop system performance under a variety of conditions. ◇

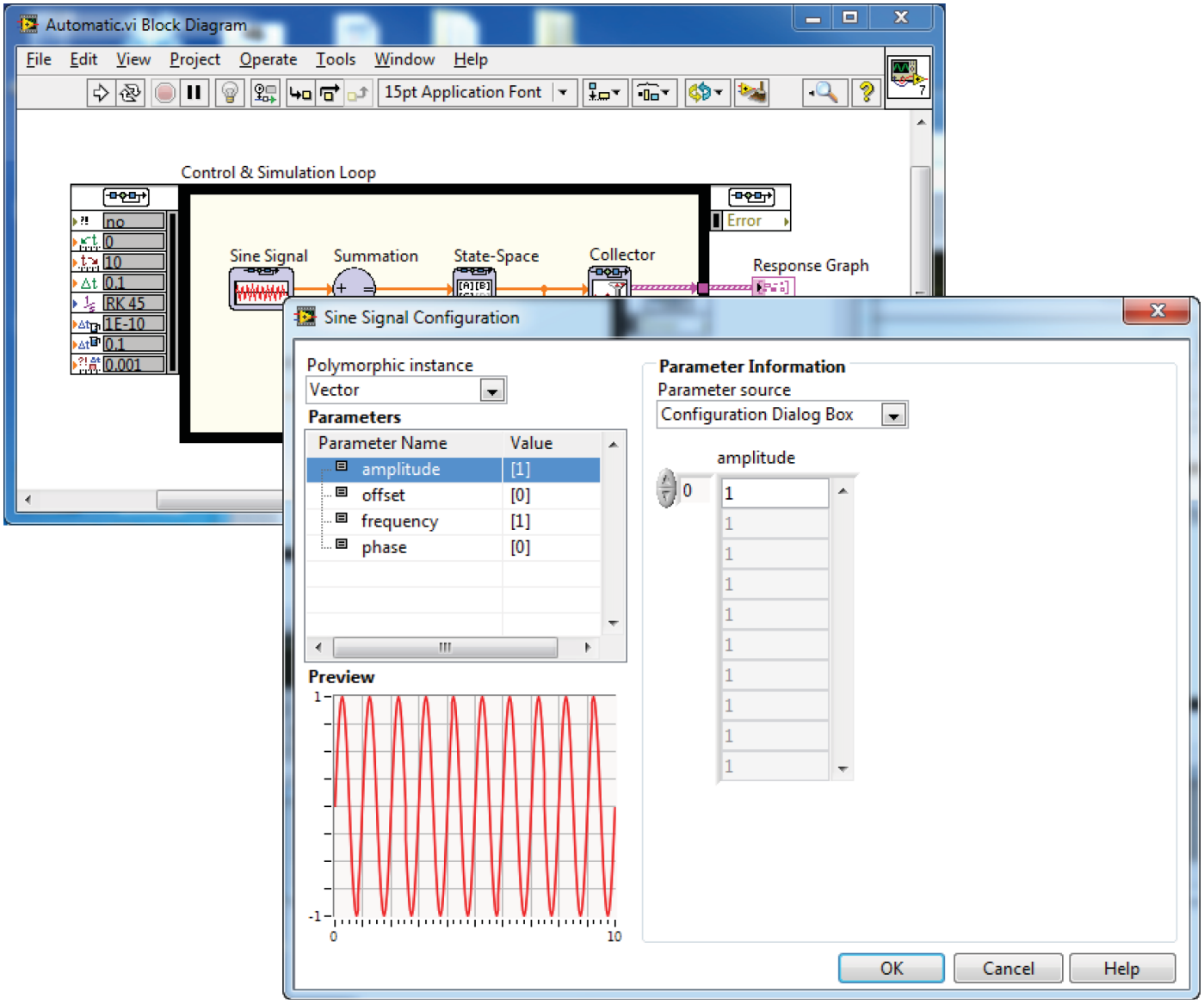


Figure 10.7: Selecting the sine wave parameters on the Sine Wave function.

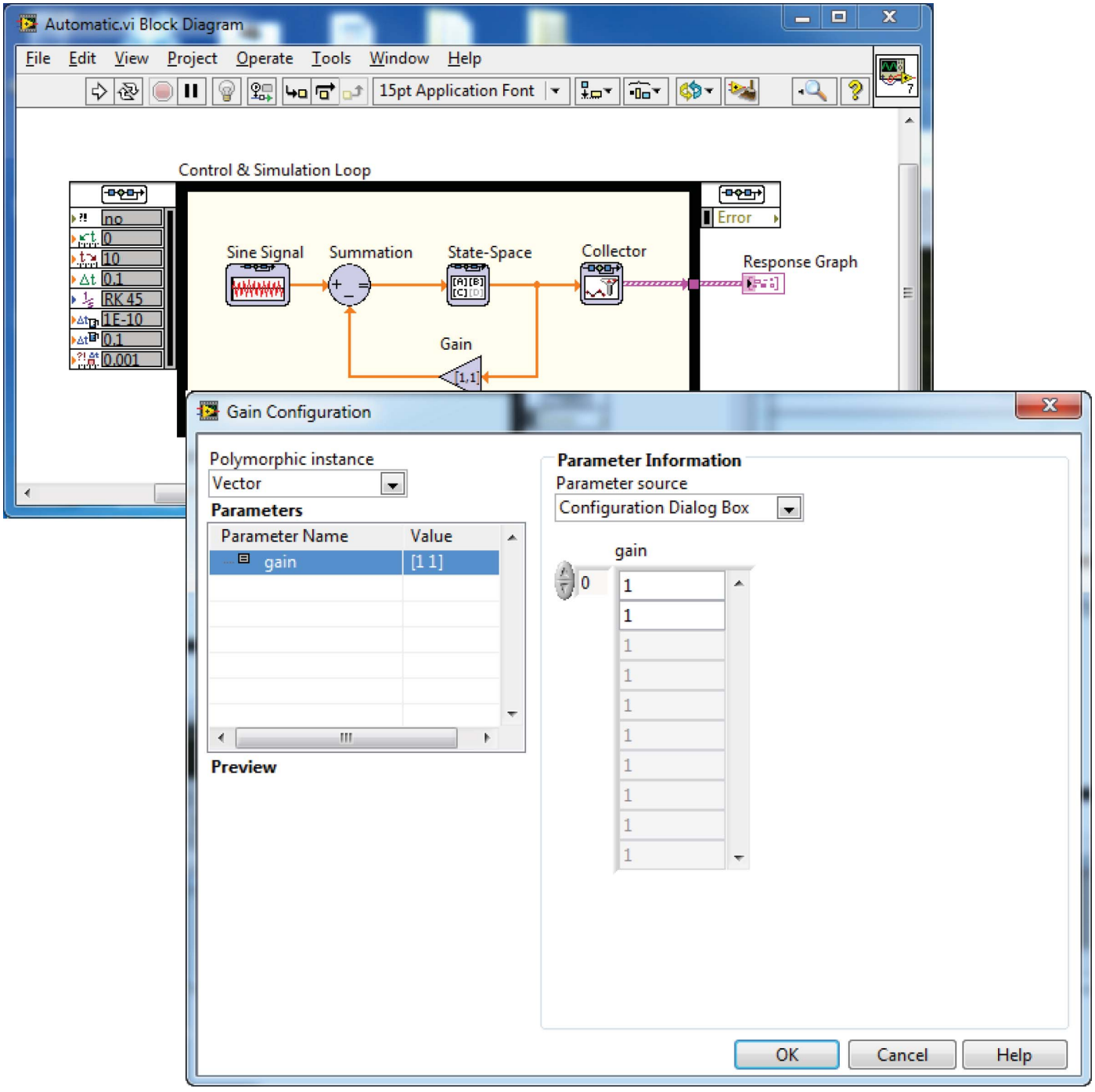


Figure 10.8: Incorporating the feedback gain into the simulation block diagram.

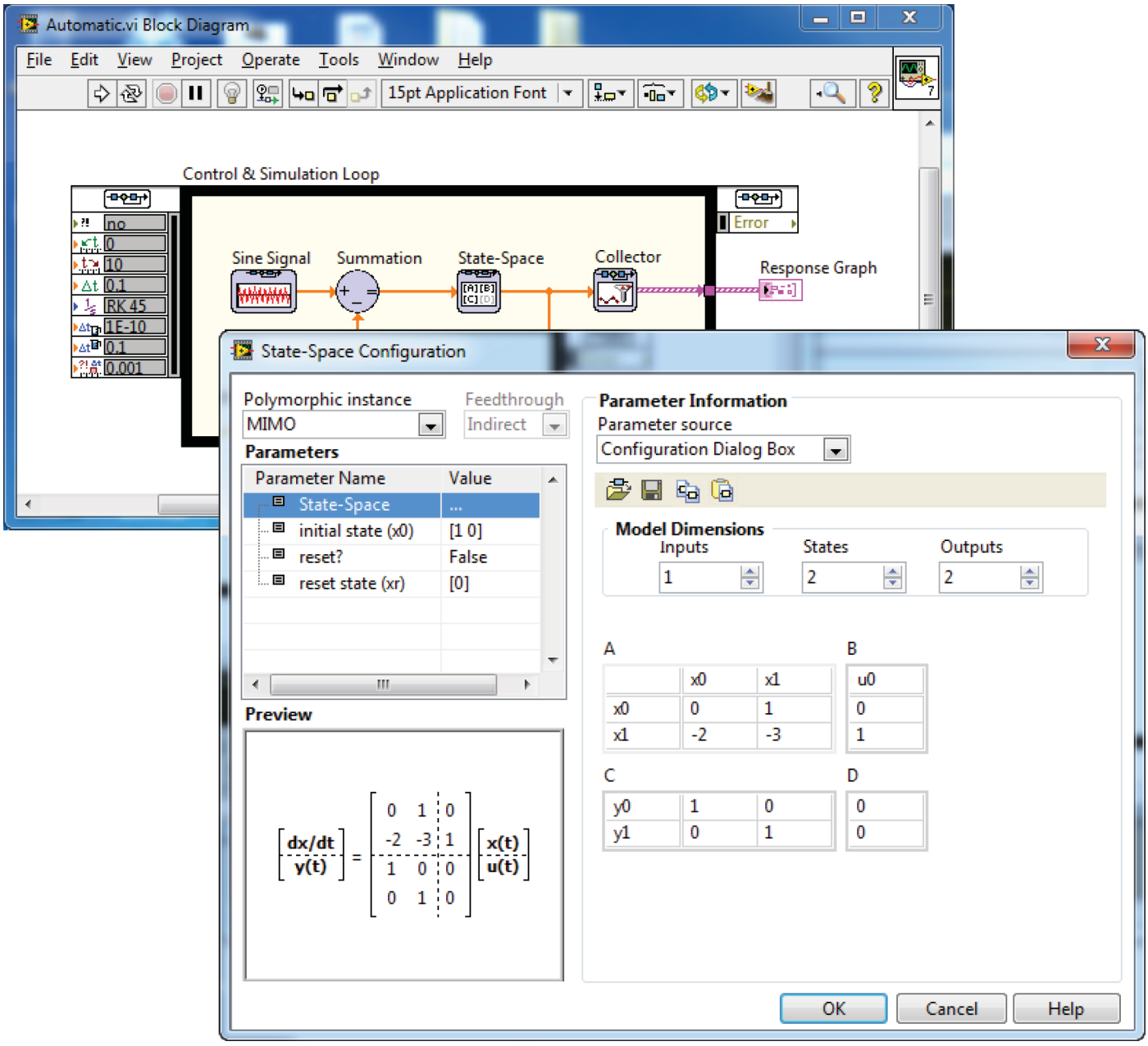


Figure 10.9: Defining the state variable model via the matrices **A**, **B**, **C**, and **D**.

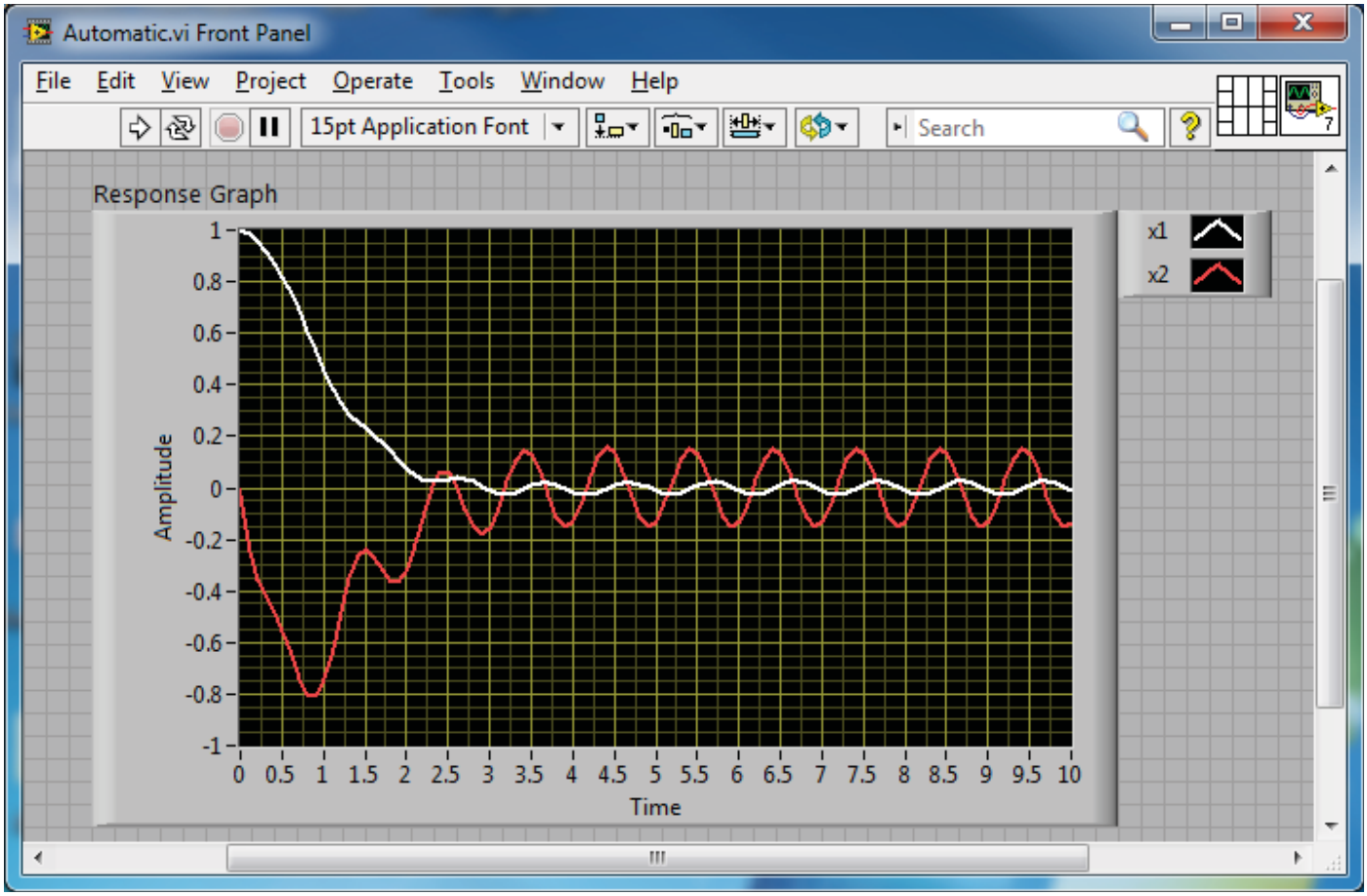


Figure 10.10: The result of the simulation showing the state variable history.

CHAPTER 11

Robust Control Systems

In this chapter, we investigate robust control systems using LabVIEW. In particular we will consider the commonly used PID controller in the feedback control system shown in Fig. 11.1. Notice that the system has a **prefilter** $G_p(s)$. The PID controller has the form

$$G_c(s) = \frac{K_D s^2 + K_P s + K_I}{s}.$$

The PID controller is not a proper rational function (i.e., the degree of the numerator polynomial is greater than the degree of the denominator polynomial). The objective is to choose the parameters K_P, K_D, and K_I to meet the performance specifications and have desirable robustness properties. Unfortunately it is not immediately clear how to choose the parameters in the PID controller to obtain certain robustness characteristics. An illustrative example will show that it is possible to choose the parameters iteratively and verify the robustness by simulation. Using LabVIEW helps in this process, since the entire design and simulation can be mechanized utilizing scripts and can easily be executed repeatedly.

Example 11.1 Robust Control of Temperature

Consider the feedback control system in Fig. 11.1, where

$$G(s) = \frac{1}{(s + c_0)^2},$$

the nominal value is $c_0 = 1$, and $G_p(s) = 1$. We will design a compensator based on $c_0 = 1$ and check robustness by simulation. Our design specifications are as follows:

1. settling time (2% criterion) $T_s \leq 0.5$ second, and
2. minimize the overshoot for a step input.

For this design we will not utilize a prefilter to meet specification (2) but will instead show that acceptable performance (i.e., low overshoot) can be obtained by increasing a cascade gain. The closed-loop transfer function is

$$T(s) = \frac{K_D s^2 + K_P s + K_I}{s^3 + (2 + K_D)s^2 + (1 + K_P)s + K_I}. \tag{11.1}$$

The associated root locus equation is

$$1 + \hat{K}\left(\frac{s^2 + as + b}{s^3}\right) = 0,$$

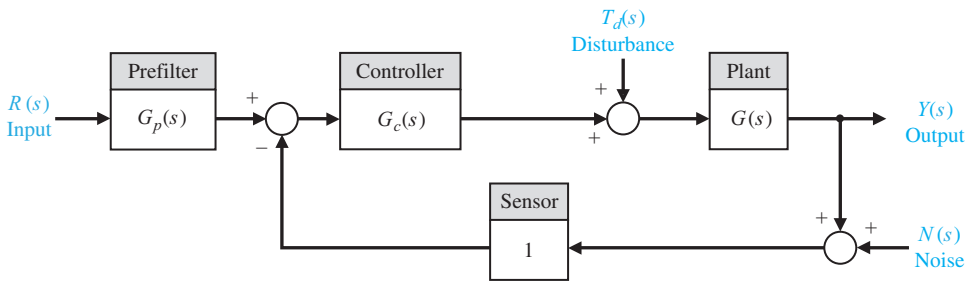


Figure 11.1: Closed-loop system.

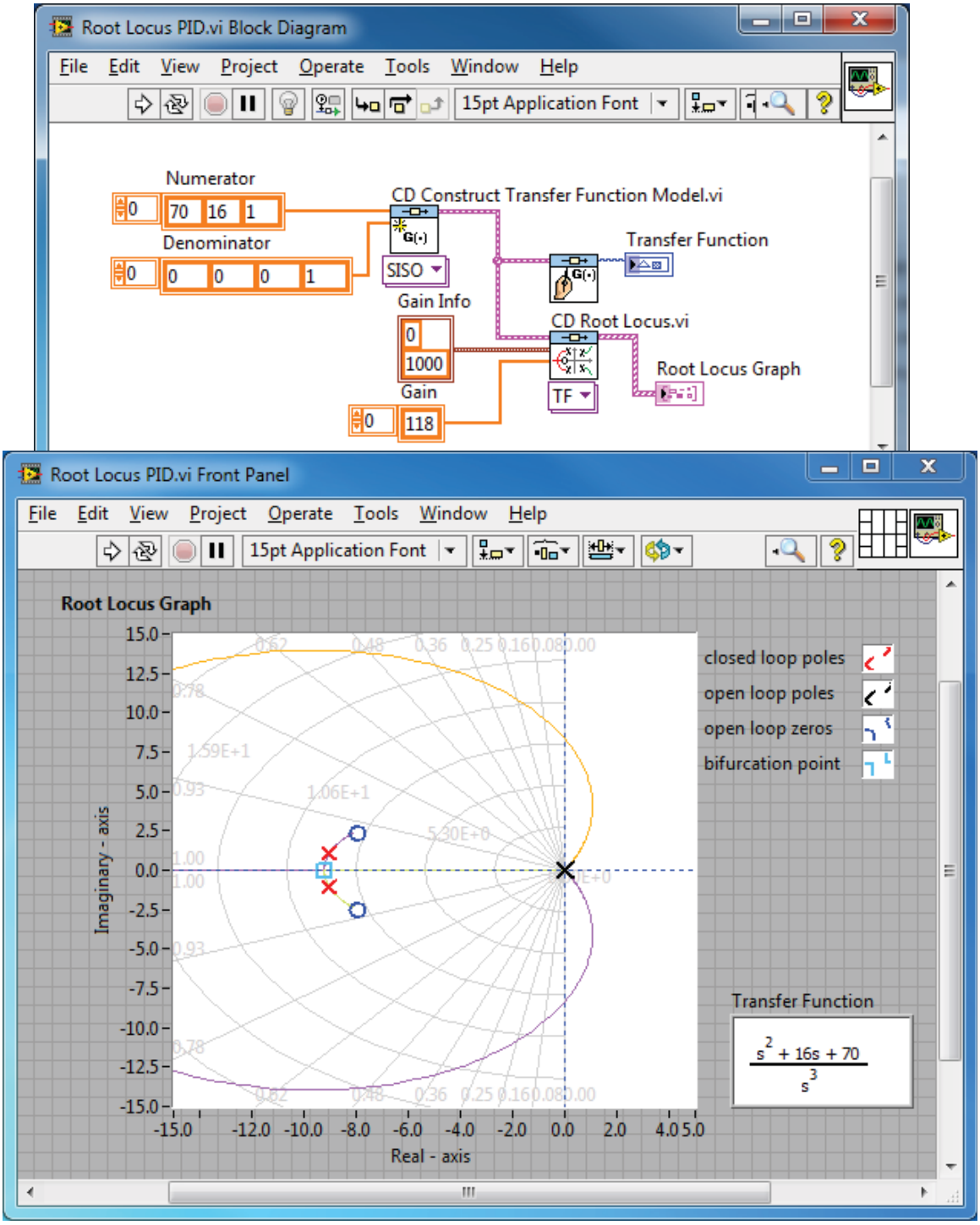


Figure 11.2: Root locus for the PID-compensated temperature controller as $\hat{K}$ varies.

where

$$\hat{K} = K_D + 2, \qquad a = \frac{1 + K_P}{2 + K_D}, \qquad b = \frac{K_I}{2 + K_D}.$$

The settling time requirement $T_s \leq 0.5$ second leads us to choose the roots of $(s^2 + as + b)$ to the left of the $s = -\zeta\omega_n = -8$ line in the s-plane, as shown in Fig. 11.2. To ensure that the locus travels into the required s-plane region, we can choose $a = 16$ and

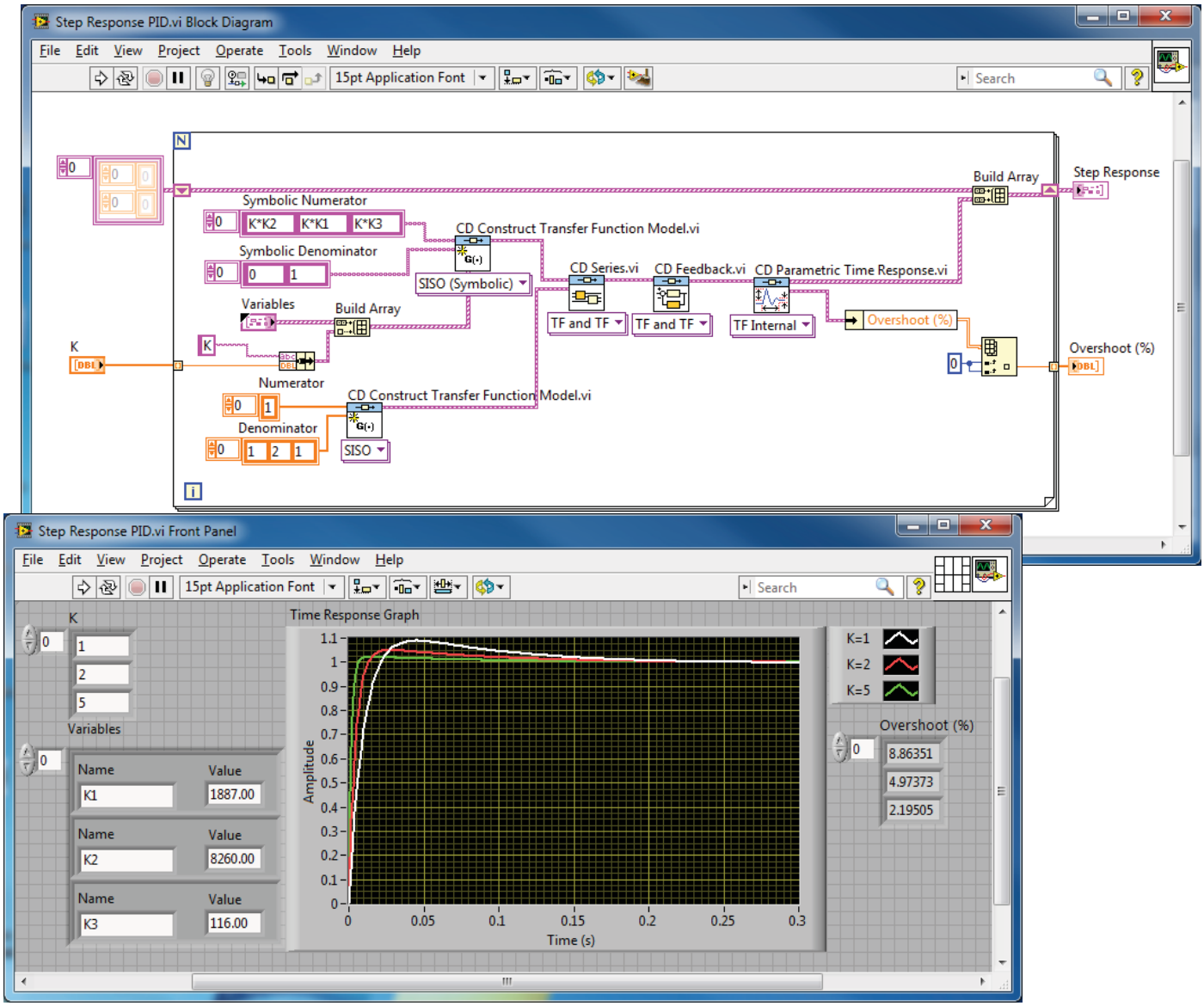


Figure 11.3: Step response of the PID temperature controller.

$b = 70$. We select a point on the root locus in the performance region and we find the associated gain and the associated value of ω_n. For our chosen point we find that $\hat{K} = 118$. Then with $\hat{K}$, a, and b, we can solve for the PID coefficients as follows:

$$\begin{aligned} K_D &= \hat{K} - 2 = 116, \\ K_P &= a(2 + K_3) - 1 = 1187, \\ K_I &= b(2 + K3) = 8260. \end{aligned}$$

To meet the overshoot performance requirements for a step input, we utilize a cascade gain K that will be chosen by iterative methods using the CD Step function, as illustrated in Fig. 11.3. The step response corresponding to $K = 5$ has an acceptable overshoot of 2%. With the addition of the gain $K = 5$, the final PID controller is

$$G_c(s) = K\,\frac{K_D s^2 + K_P s + K_I}{s} = 5\,\frac{116s^2 + 1187s + 8260}{s}. \tag{11.2}$$

Now we can consider the question of robustness to changes in the plant parameter c_0. The investigation into the robustness of the design consists of a step response analysis using the PID controller given in Eq. (11.2) for a range of plant parameter variations of

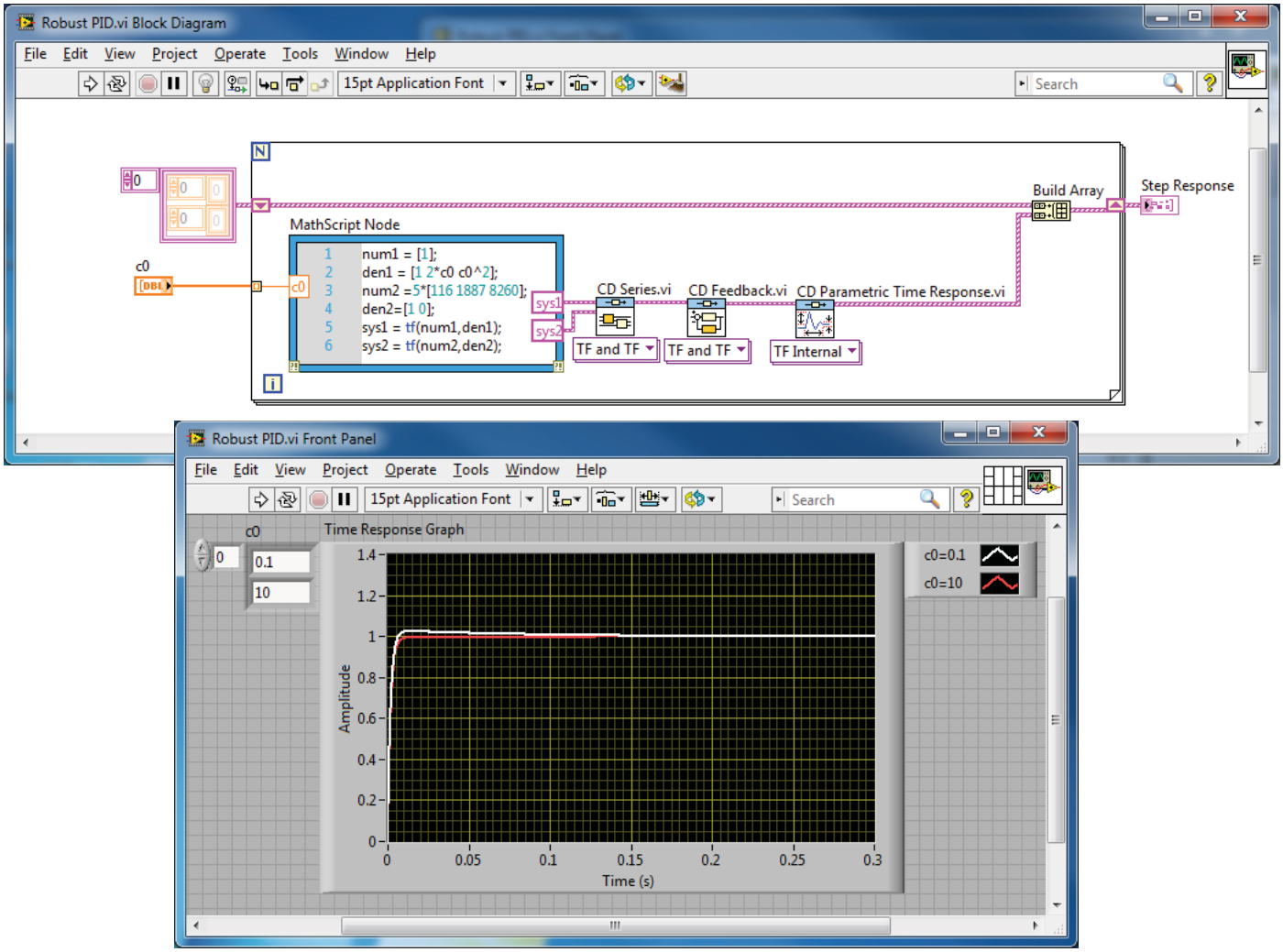


Figure 11.4: Robust PID controller analysis with variations in c_0.

$0.1 \leq c_0 \leq 10$. The results are displayed in Fig. 11.4. The simulation results indicate that the PID design is robust with respect to changes in c_0. The differences in the step responses for $0.1 \leq c_0 \leq 10$ are barely discernible on the plot. If the results showed otherwise, it would be possible to iterate on the design until an acceptable performance was achieved.The interactive capability of LabVIEW allows us to check the robustness by simulation. ◇

CHAPTER 12

Digital Control Systems

Many of the LabVIEW functions for continuous-time control design have equivalent counterparts for sampled-data systems. Model conversion can be accomplished with the functions CD Convert Continuous to Discrete and CD Convert Discrete to Continuous. The function CD Convert Continuous to Discrete converts continuous-time systems to discrete-time systems. Similarly, the function CD Convert Discrete to Continuous converts discrete-time systems to continuous-time systems. For example, consider the plant transfer function

$$G_p(s) = \frac{1}{s(s+1)},$$

as shown in Fig. 12.1. For a sampling period of $T = 1$ second, we determine that

$$G(z) = \frac{0.3678(z+0.7189)}{(z-1)(z-0.3680)} = \frac{0.3678z + 0.2644}{z^2 - 1.368z + 0.3680}. \tag{12.1}$$

We can use LabVIEW to obtain the $G(z)$, as shown in Fig. 12.2.

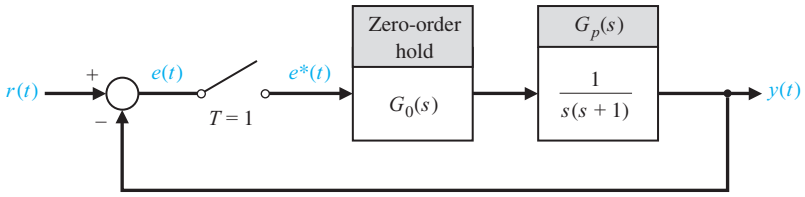


Figure 12.1: A closed-loop sampled data system.

Example 12.1 Discrete System Unit Step Response

We can use LabVIEW to compute the response $y(kT)$ of a closed-loop sampled data system using the CD Step Response function, shown in Fig. 12.3. Consider again the system shown in Fig. 12.1. The closed-loop transfer function is given by

$$G(z) = \frac{0.3678z + 0.2644}{z^2 - z + 0.6322},$$

and the associated closed-loop step response is shown in Fig. 12.3. The zero-order hold is modeled by the transfer function $G_0(s)$, where

$$G_0(s) = \frac{1 - e^{-sT}}{s}.$$

◇

In the next example, we consider the root locus of a sampled-data system.

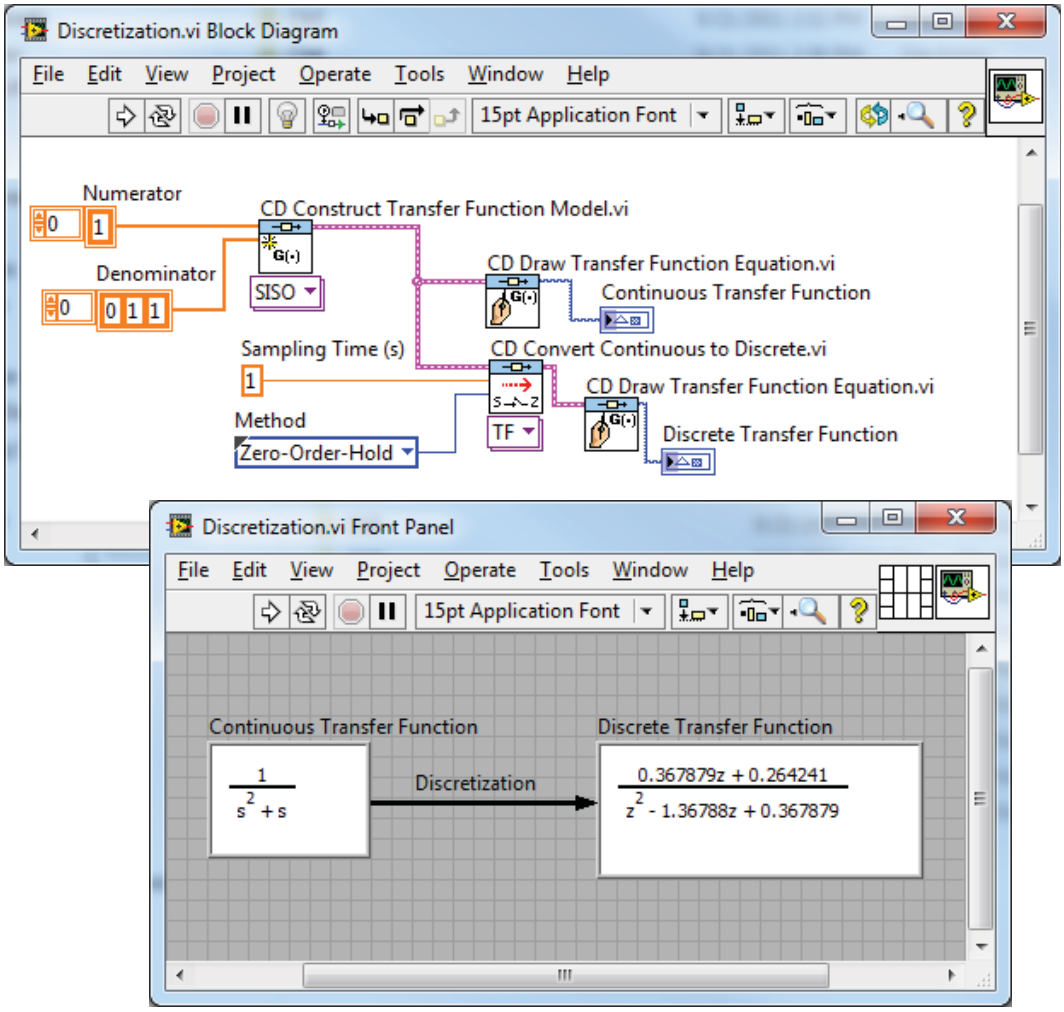


Figure 12.2: Using the CD Convert Continuous to Discrete function to convert $G(s) = G_0(s)G_p(s)$ to $G(z)$.

Example 12.2 Root Locus of a Digital Control System

Consider the system shown in Fig. 12.4 where

$$G(z) = \frac{0.3678(z + 0.7189)}{(z - 1)(z - 0.3680)} ,$$

and the compensator is

$$D(z) = \frac{K(z - 0.3678)}{z + 0.2400} .$$

The parameter K is a variable yet determined. When

$$G(z)D(z) = K\frac{0.3678(z + 0.7189)}{(z - 1)(z + 0.2400)} , \tag{12.2}$$

we have the problem in a form for which the root locus method is directly applicable. The rlocus function works for discrete-time systems in the same way as for continuous- time systems. Using a LabVIEW script, the root locus associated with Eq. (12.2) is easily generated, as shown in Fig. 12.5.

Remember that the stability region is defined by the unit circle in the complex plane. The LabVIEW function rlocfind can be used with the discrete-time system root locus in exactly the same way as for continuous-time systems to determine the value of the system gain associated with any point on the locus. Using rlocfind we determine that $K = 4.639$ places the roots on the unit circle. ◇

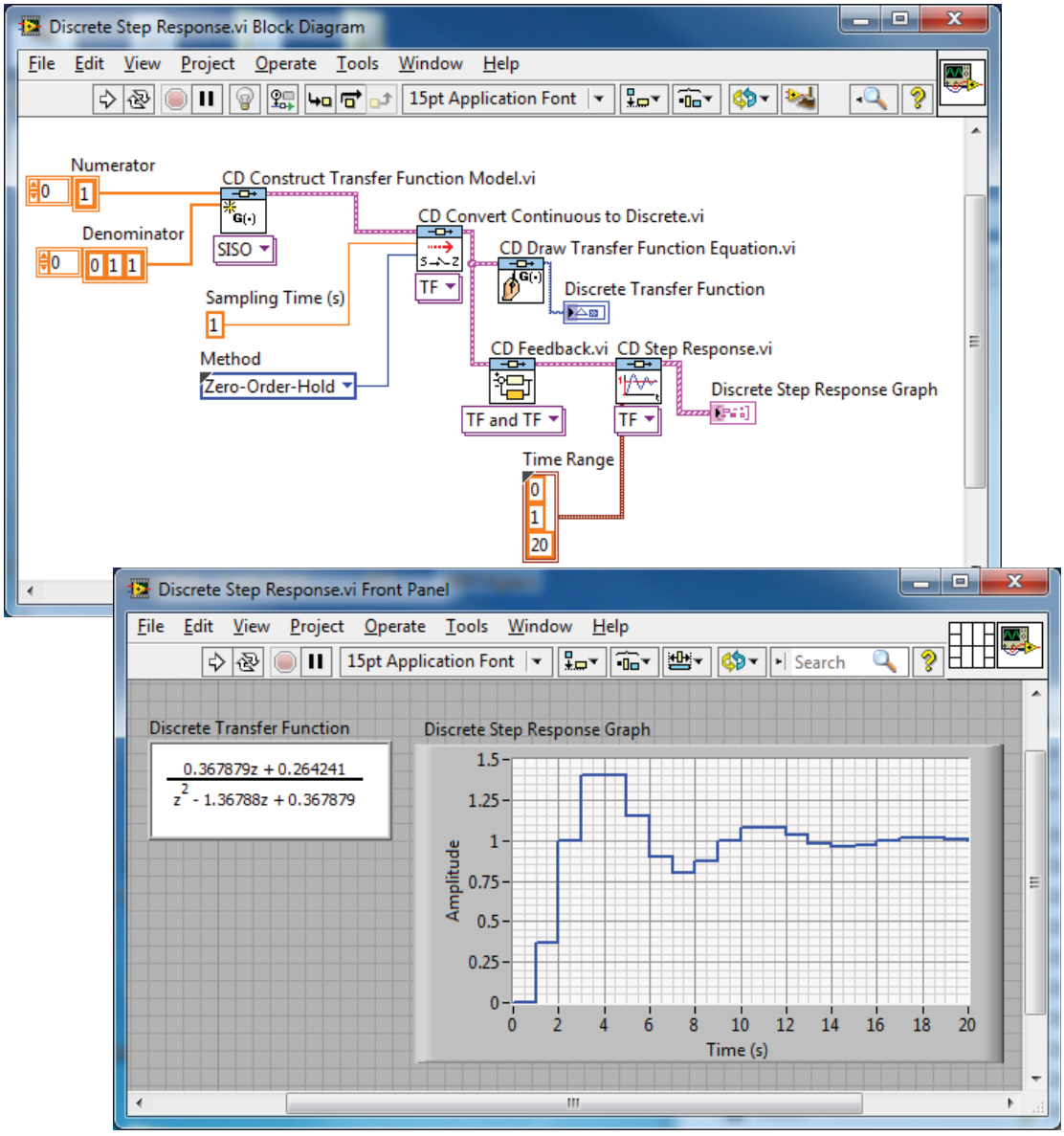


Figure 12.3: The CD Step Response function generates the output $y(kT)$ for a step input.

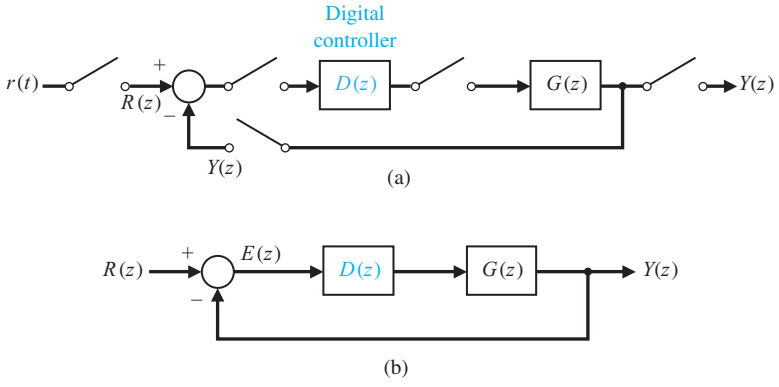


Figure 12.4: (a) Feedback control system with digital compensation. (b) Block diagram model with $G(z) = \mathcal{Z}\left[G_0(s)G_p(s)\right]$.

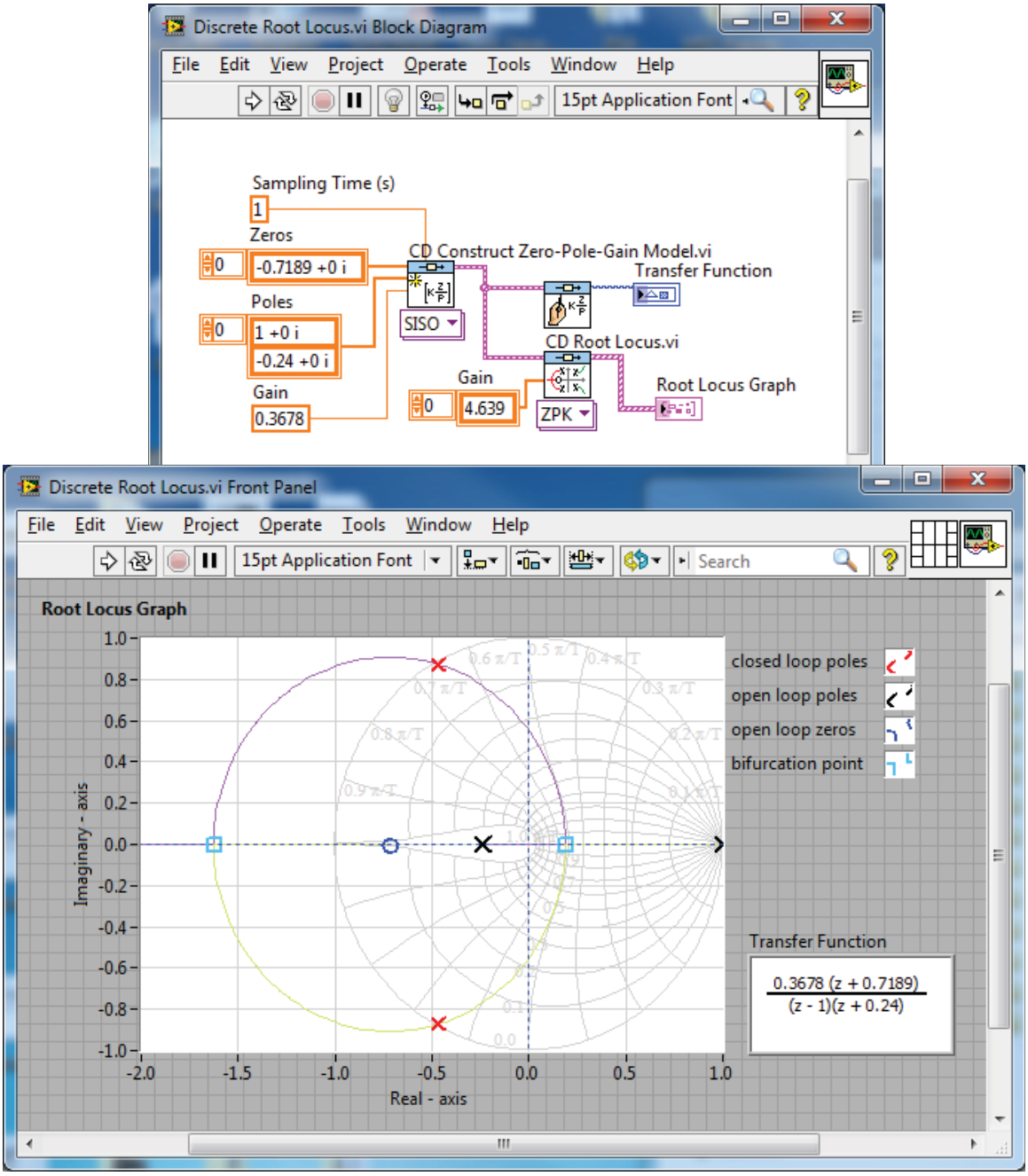


Figure 12.5: The CD Root Locus function for sampled data systems.

INDEX