Observers can be used to augment or replace sensors in a control system. Observers are algorithms that combine sensed signals with other knowledge of the control system to produce observed signals. These observed signals can be more accurate, less expensive to produce, and more reliable than sensed signals. Observers offer designers an inviting alternative to adding new sensors or upgrading existing ones.

This book is written as a guide for the selection and installation of observers in control systems. It will discuss practical aspects of observers such as how to tune an observer and what conditions make a system likely to benefit from their use. Of course, observers have practical shortcomings, many of which will be discussed here as well. Many books on observers give little weight to practical aspects of their use. Books on the subject often focus on mathematics to prove concepts that are rarely helpful to the working engineer. Here the author has minimized the mathematics while concentrating on intuitive approaches.

The author assumes that the typical reader is familiar with the use of traditional control systems, either from practical experience or from formal training. The nature of observers recommends that users be familiar with traditional (nonobserver-based) control systems in order to better recognize the benefits and shortcomings of observers. Observers offer important advantages: they can remove sensors, which reduces cost and improves reliability, and improve the quality of signals that come from the sensors, allowing performance enhancement. However, observers have disadvantages: they can be complicated to implement and they expend computational resources. Also, because observers form software control loops, they can become unstable under certain conditions. A person familiar with the application of control systems will be in a better position to evaluate where and how to use an observer.

The issues addressed in this book fall into two broad categories: design and implementation. Design issues are those issues related to the selection of observer techniques for a given product. How much will the observer improve performance? How much cost will it add? What are the limitations of observers? These issues will help the control-systems engineer in deciding whether an observer will be useful and in estimating the required resources. On the other hand, implementation issues are those issues related to the installation of observers. Examples include how to tune an observer and how to recognize the effects of changing system parameters on observer performance.

1.2 Preview of Observers

Observers work by combining knowledge of the plant, the power converter output, and the feedback device to extract a feedback signal that is superior to that which can be obtained by using a feedback device alone. An example from everyday life is when an experienced driver brings a car to a rapid stop. The driver combines knowledge of the applied stopping power (primarily measured through inertial forces acting on the
driver’s body) with prior knowledge of the car’s dynamic behavior during braking. An experienced driver knows how a car should react to braking force and uses that information to bring a car to a rapid but controlled stop.

The principle of an observer is that by combining a measured feedback signal with knowledge of the control-system components (primarily the plant and feedback system), the behavior of the plant can be known with greater precision than by using the feedback signal alone. As shown in Figure 1-1, the observer augments the sensor output and provides a feedback signal to the control laws.

In some cases, the observer can be used to enhance system performance. It can be more accurate than sensors or can reduce the phase lag inherent in the sensor. Observers can also provide observed disturbance signals, which can be used to improve disturbance response. In other cases, observers can reduce system cost by augmenting the performance of a low-cost sensor so that the two together can provide performance equivalent to a higher cost sensor. In the extreme case, observers can eliminate a sensor altogether, reducing sensor cost and the associated wiring. For example, in a method called acceleration feedback, which will be discussed in Chapter 8, acceleration is observed using a position sensor and thus eliminating the need for a separate acceleration sensor.

Observer technology is not a panacea. Observers add complexity to the system and require computational resources. They may be less robust than physical sensors, especially when plant parameters change substantially during operation. Still, an observer applied with skill can bring substantial performance benefits and do so, in many cases, while reducing cost or increasing reliability.

![Figure 1-1. Role of an observer in a control system.](image-url)
CHAPTER 1 CONTROL SYSTEMS AND THE ROLE OF OBSERVERS

1.3 Summary of the Book

This book is organized assuming that the reader has some familiarity with controls but understanding that working engineers and designers often benefit from review of the basics before taking up a new topic. Thus, the next two chapters will review control systems. Chapter 2 discusses practical aspects of control systems, seeking to build a common vocabulary and purpose between author and reader. Chapter 3 reviews the frequency domain and its application to control systems. The techniques here are discussed in detail assuming the reader has encountered them in the past but may not have practiced them recently.

Chapter 4 introduces the Luenberger observer structure, which will be the focus of this book. This chapter will build up the structure relying on an intuitive approach to the workings and benefits of observers. The chapter will demonstrate the key advantages of observers using numerous software experiments.

Chapters 5, 6, and 7 will discuss the behavior of observer-based systems in the presence of three common nonideal conditions. Chapter 5 deals with the effects of imperfect knowledge of model parameters, Chapter 6 deals with the effects of disturbances on observer-based systems, and Chapter 7 discusses the effects of noise, especially sensor noise, on observer-based systems.

Chapter 8 discusses the application of observer techniques to motion-control systems. Motion-control systems are unique among control systems, and the standard Luenberger observer is normally modified for those applications. The details of the necessary changes, and several applications, will be discussed.

Throughout this book, software experiments are used to demonstrate key points. A simulation environment, Visual ModelQ, developed by the author to aid those studying control systems, will be relied upon. More than two dozen models have been developed to demonstrate key points and all versions of Visual ModelQ can run them. Visit www.qxdesign.com to download a limited-capability version free of charge; detailed instructions on setting up and using Visual ModelQ are given in Chapter 2.

Readers wishing to contact the author are invited to do so. Write gellis@qxdesign.com or visit the Web site www.qxdesign.com. Your comments are most welcome. Also, visit www.qxdesign.com to review errata, which will be regularly updated by the author.
In this chapter...

- Common control-system structures
- Eight goals of control systems and implications of observer-based methods
- Instructions for downloading Visual ModelQ, a simulation environment that is used throughout this book
- Introductory Visual ModelQ software experiments

2.1 Control-System Structures

The basic control loop includes four elements: a control law, a power converter, a plant, and a feedback sensor. Figure 2-1 shows the typical interconnection of these functions. The command is compared to the feedback signal to generate an error signal. This error signal is fed into a control law such as a proportional-integral (PI) control to generate an excitation command. The excitation command is processed by a power converter to produce an excitation. The excitation is corrupted by a disturbance and then fed to a plant. The plant response is measured by a sensor, which generates the feedback signal.

There are numerous variations on the control loop of Figure 2-1. For example, the control-law is sometimes divided in two with some portion placed in the feedback path. In addition, the command path may be filtered. The command path may be differentiated and added directly (that is, without passing through the control laws) to the excitation command in a technique known as feed-forward. Still, the diagram of Figure 2-1 is broadly used and will be considered the basic control loop in this book.
2.1.1 Control Laws

Control laws are algorithms that determine the desired excitation based on the error signal. Typically, control laws have two or three terms: one scaling the present value of the error (the proportional term), another scaling the integral of the error (the integral term), and a third scaling the derivative of the error (the derivative term). In most cases a proportional term is used; an integral term is added to drive the average value of the error to zero. That combination is called a PI controller and is shown in Figure 2-2.

When the derivative or D-term is added, the PI controller becomes PID. Derivatives are added to stabilize the control loop at higher frequencies. This allows the value of the proportional term to be increased, improving the responsiveness of the control loop. Unfortunately, the process of differentiation is inherently noisy. The use of the D-term usually requires low-noise feedback signals and low-pass filtering to be effective. Filtering reduces noise but also adds phase lag, which reduces the ultimate effectiveness of the D-term. A compromise must be reached between stabilizing the loop, which requires the phase advance of differentiation, and noise attenuation, which retards phase. Usually such a compromise is application specific. Note that
when a derivative term is placed in series with a low-pass filter, it is sometimes referred to as a lead network. A typical PID controller is shown in Figure 2-3.

Other terms may be included in the control law. For example, a term scaling the second derivative can be used to provide more phase advance; this is equivalent to two lead terms in series. Such a structure is not often used because of the noise that it generates. In other cases, a second integral is added to drive the integral of the error to zero. Again, this structure is rarely used in industrial controls. First, few applications require driving the integral of error to zero; second, the additional integral term makes the loop more difficult to stabilize.

Filters are commonly used within control laws. The most common purpose is to reduce noise. Filters may be placed in line with the feedback device or the control-law output. Both positions provide similar benefits (reducing noise output) and similar problems (adding phase lag and thus destabilizing the loop). As discussed above, low-pass filters can be used to reduce noise in the differentiation process. Filters can be used on the command signal, sometimes to reduce noise and other times to improve step response. The improvement in step response comes about because, by removing high-frequency components from the command input, overshoot in the response can be reduced. Command filters do not destabilize a control system because they are outside the loop. A typical PI control law is shown in Figure 2-4 with three common filters.

While low-pass filters are the most common variety in control systems, other filter types are used. Notch filters are sometimes employed to attenuate a narrow band of frequencies. They may be used in the feedback or control-law filters to help stabilize the control loop in the presence of a resonant frequency, or they may be used to remove a narrow band of unwanted frequency content from the command. Also, phase-advancing filters are sometimes employed to help stabilize the control loop similar to the filtered derivative path in the PID controller.

Control laws can be based on numerous technologies. Digital control is common and is implemented by programmable logic controllers (PLCs), personal computers...
(PCs), and other computer-based controllers. Because the flexibility of digital controllers is almost required for observer implementation and because the control law and observer are typically implemented in the same device, examples in this book will assume control laws are implemented digitally.

2.1.2 Power Conversion

Power conversion is the process of delivering power to the plant as called for by the control laws. Four common categories of power conversion are chemical heat, electric voltage, evaporation/condensation, and fluid pressure. Note that all these methods can be actuated electronically and so are compatible with electronic control laws.

Electronically or electrically controlled voltage can be used as the power source for power supplies, current controllers for motors, and heating. For systems with high dynamic rates, power transistors can be used to apply voltage. For systems with low dynamic rates, relays can be used to switch power on and off. A simple example of such a system is an electric water heater.

Pressure-based flow-control power converters often use valves to vary pressure applied to a fluid-flow system. Chemical power conversion uses chemical energy such as combustible fuel to heat a plant. A simple example of such a system is a natural-gas water heater.

2.1.3 Plant

The plant is the final object under control. Most plants fall into one of six major categories: motion, navigation, fluid flow, heat flow, power supplies, and chemical processes. Most plants have at least one stage of integration. That is, the input to the plant is integrated at least once to produce the system response. For example, the temperature of an object is controlled by adding or taking away heat; that heat is

Figure 2-4. PI control law with several filters in place.
### Electrical

<table>
<thead>
<tr>
<th>Component</th>
<th>Transfer Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inductance ($L$)</td>
<td>$E(s) = LsI(s)$</td>
<td>$e(t) = L \frac{di(t)}{dt}$</td>
</tr>
<tr>
<td>Capacitance ($C$)</td>
<td>$E(s) = \frac{1}{sC}I(s)$</td>
<td>$e(t) = C \int i(t) , dt$</td>
</tr>
<tr>
<td>Resistance ($R$)</td>
<td>$E(s) = Ri(s)$</td>
<td>$e(t) = Ri(t)$</td>
</tr>
</tbody>
</table>

### Translational mechanics

<table>
<thead>
<tr>
<th>Component</th>
<th>Transfer Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position ($P$), Velocity ($V$), and Force ($F$)</td>
<td>$V(s) = sF(s)$</td>
<td>$v(t) = 1/K \frac{df(t)}{dt}$ or $p(t) = p_0 + 1/K \int f(t) , dt$</td>
</tr>
<tr>
<td>Spring ($K$)</td>
<td>$V(s) = 1/K \times F(s)$</td>
<td>$p(t) = p_0 + 1/K \int f(t) , dt$</td>
</tr>
<tr>
<td>Mass ($M$)</td>
<td>$V(s) = 1/M \times F(s)$</td>
<td>$v(t) = v_0 + 1/M \int f(t) , dt$</td>
</tr>
<tr>
<td>Damper ($c$)</td>
<td>$V(s) = F(s)/c$</td>
<td>$v(t) = f(t)/c$</td>
</tr>
</tbody>
</table>

### Rotational mechanics

<table>
<thead>
<tr>
<th>Component</th>
<th>Transfer Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotary position ($\theta$), Rotary velocity ($\omega$), and Torque ($T$)</td>
<td>$\omega(s) = s\omega_0$</td>
<td>$\omega(t) = \omega_0 + \int \omega(t) , dt$</td>
</tr>
<tr>
<td>Spring ($K$)</td>
<td>$\omega(s) = 1/K \times T(s)$</td>
<td>$\omega(t) = \omega_0 + \int T(t) , dt$</td>
</tr>
<tr>
<td>Inertia ($J$)</td>
<td>$\omega(s) = 1/J \times T(s)$</td>
<td>$\omega(t) = \omega_0 + \int T(t) , dt$</td>
</tr>
<tr>
<td>Damper ($b$)</td>
<td>$\omega(s) = T(s)/b$</td>
<td>$\omega(t) = T(t)/b$</td>
</tr>
</tbody>
</table>

### Fluid mechanics

<table>
<thead>
<tr>
<th>Component</th>
<th>Transfer Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure ($P$) and fluid flow ($Q$)</td>
<td>$P(s) = sI \times Q(s)$</td>
<td>$p(t) = I \times dq(t)/dt$</td>
</tr>
<tr>
<td>Inertia ($I$)</td>
<td>$P(s) = sI \times Q(s)$</td>
<td>$p(t) = I \times dq(t)/dt$</td>
</tr>
<tr>
<td>Capacitance ($C$)</td>
<td>$P(s) = 1/C \times Q(s)/s$</td>
<td>$p(t) = p_0 + 1/C \int q(t) , dt$</td>
</tr>
<tr>
<td>Resistance ($R$)</td>
<td>$P(s) = R \times Q(s)$</td>
<td>$p(t) = R \times q(t)$</td>
</tr>
</tbody>
</table>

### Heat flow

<table>
<thead>
<tr>
<th>Component</th>
<th>Transfer Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature difference ($J$) and heat flow ($Q$)</td>
<td>$J(s) = 1/C \times Q(s)/s$</td>
<td>$j(t) = j_0 + 1/C \int q(t) , dt$</td>
</tr>
<tr>
<td>Capacitance ($C$)</td>
<td>$J(s) = 1/C \times Q(s)/s$</td>
<td>$j(t) = j_0 + 1/C \int q(t) , dt$</td>
</tr>
<tr>
<td>Resistance ($R$)</td>
<td>$J(s) = R \times Q(s)$</td>
<td>$j(t) = R \times q(t)$</td>
</tr>
</tbody>
</table>

Table 2-1 shows the relationships in a variety of ideal plants.

The pattern of force, impedance, and flow is repeated for many physical elements. In Table 2-1, the close parallels between the categories of linear and rotational force, fluid mechanics, and heat flow are evident. In each case, a forcing function (voltage, force, torque, pressure, or temperature difference) applied to an impedance produces a flow (current, velocity, fluid flow, or thermal flow). The impedance takes three forms: resistance to the integral of flow (capacitance or mass), resistance to the derivative of flow (spring or inductance), and resistance to the flow rate (resistance or damping).
Table 2-1 reveals a central concept of controls. Controllers for these elements apply a force to control a flow. When the flow must be controlled with accuracy, a feedback sensor is often added to measure the flow; control laws are required to combine the feedback and command signals to generate the force. This results in the structure shown in Figure 2-1; it is this structure that sets control systems apart from other disciplines of engineering.

2.1.4 Feedback Sensors

Feedback sensors provide the control system with measurements of physical quantities necessary to close control loops. The most common sensors are for motion states (position, velocity, acceleration, and mechanical strain), temperature states (temperature and heat flow), fluid states (pressure, flow, and level), and electromagnetic states (voltage, current, charge, light, and magnetic flux). The performance of most traditional (nonobserver) control systems depends, in large part, on the quality of the sensor. Control-system engineers often go to great effort to specify sensors that will provide responsive, accurate, and low-noise feedback signals. While the plant and power converter may include substantial imperfections (for example, distortion and noise), such characteristics are difficult to tolerate in feedback devices.

2.1.4.1 Errors in Feedback Sensors

Feedback sensors measure signals imperfectly. The three most common imperfections, as shown in Figure 2-5, are intrinsic filtering, noise, and cyclical error.

The intrinsic filtering of a sensor limits how quickly the feedback signal can follow the signal being measured. The most common effect of this type is low-pass filtering. For all sensors there is some frequency above which the sensor cannot fully respond. This may be caused by the physical structure of the sensor. For example, many thermal sensors have thermal mass; time is required for the object under measurements

![Figure 2-5. A practical sensor is a combination of an ideal sensor and error sources.](image-url)
to warm and cool the sensor’s thermal mass. Filtering may also be explicit as in the case of electrical sensors where passive filters are connected to the sensor output to attenuate noise.

Whatever the source of the filtering, its primary effect on the control system is to add phase lag to the control loop. Phase lag reduces the stability margin of the control loop and makes the loop more difficult to stabilize. The result is often that system gains must be reduced to maintain stability in order to accommodate slow sensors. Reducing gains is usually undesirable because both command and disturbance response degrade.

Cyclical error is the repeatable error that is induced by sensor imperfections. For example, a strain gauge measures strain by monitoring the change in electrical parameters of the gauge material that is seen when the material is deformed. The behavior of these parameters for ideal materials is well known. However, there are slight differences between an ideal strain gauge and any sample. Those differences result in small, repeatable errors in measuring strain. Since cyclical errors are deterministic, they can be compensated out in a process where individual samples of sensors are characterized against a highly accurate sensor. However, in any practical sensor some cyclical error will remain. Because control systems are designed to follow the feedback signal as well as possible, in many cases the cyclical error will affect the control-system response.

Stochastic or nondeterministic errors are those errors that cannot be predicted. The most common example of stochastic error is high-frequency noise. High-frequency noise can be generated by electronic amplification of low-level signals and by conducted or transmitted electrical noise commonly known as electromagnetic interference (EMI). High-frequency noise in sensors can be attenuated by the use of electrical filters; however, such filters restrict the response rate of the sensor as discussed above. Designers usually work hard to minimize the presence of electrical noise, but as with cyclical error, some noise will always remain. Filtering is usually a practical cure for such noise; it can have minimal negative effect on the control system if the frequency content is high enough so that the filter affects only frequency ranges well above where phase lag is a concern in the application.

The end effect of sensor error on the control system depends on the error type. Limited responsiveness commonly introduces phase lag in the control system, reducing margins of stability. Noise makes the system unnecessarily active and may reduce the perceived value of the system or keep the system from meeting a specification. Deterministic errors corrupt the system output. Because control systems are designed to follow the feedback signal (including its deterministic errors) as well as possible, deterministic errors will carry through, at least in part, to the control-system response.

2.1.5 Disturbances

Disturbances are undesired inputs to the control system. Common examples include load torque in a motion-control system, changes in ambient temperature for a temperature controller, and 50/60-Hz noise in a power supply. In each case, the
primary concern is that the control law generate plant excitation to reject (i.e., prevent response to) these inputs. A correctly placed integrator will totally reject direct-current (DC) disturbances. High tuning gains will help the system reject alternating-current (AC) disturbance inputs, but will not reject those inputs entirely.

Disturbances can be either deterministic or stochastic. Deterministic disturbances are those disturbances that repeat when conditions are duplicated. Such disturbances are predictable. Stochastic disturbances are not predictable.

The primary way for control systems to reject disturbances is to use high gains in the control law. High gains force the control-system response to follow the command despite disturbances. Of course, there is an upper limit to gain values because high gains reduce system stability margins and, when set high enough, will cause the system to become unstable.

2.1.5.1 Measuring Disturbances

In the case where the control-system gains have been raised as high as is practical, disturbance rejection can still be improved by using a signal representing the disturbance in a technique known as disturbance decoupling [11, Chap. 7; 26; 27]. Disturbance decoupling, as shown in Figure 2-6, is a cancellation technique where a signal representing the disturbance is fed into the power converter in opposition to the effect of the disturbance. For the case of ideal disturbance measurement and ideal power

![Figure 2-6. Typical use of disturbance decoupling.](image-url)
conversion, disturbance decoupling eliminates the effects of the disturbance entirely. However, for practical systems, the effect of disturbance decoupling is to improve, but not eliminate, response to disturbances; this is especially true in the lower frequencies where the disturbance sensor and the power converter are often close to ideal.

For most control systems, direct measurement of disturbances is impractical. Disturbances are usually difficult to measure and physical sensors carry with them numerous disadvantages, especially increasing system cost and reducing reliability. One of the key benefits of observers is that disturbance signals can often be observed with accuracy without requiring additional sensors. For many applications, only modest computational resources must be added to implement such an observer. This topic will be discussed in detail in Chapters 6 and 8.

### 2.2 Goals of Control Systems

Control systems must fulfill a complicated combination of requirements. A large set of goals must be considered because no single measure can provide a satisfactory assessment. In fact, no single set of goals can be defined for general use because of the variation between applications. However, many common goals are broadly used in combination. In this section, eight common goals for control systems will be discussed. In addition, the role of observers in helping or, in some cases, hindering the realization of those goals will be discussed.

#### 2.2.1 Competitively Priced

Control systems, like almost all products in the industrial market, must be delivered at competitive prices. The virtues of a control system will be of little value if the application can be served equally well by a less expensive alternative. This is not to say that a customer will not pay a premium for enhanced performance. However, the manufacturer offering premium products must demonstrate that the premium will improve the cost–value position of the final product.

Arguments for observer-based methods can be at either end of the cost–value spectrum. For example, if an observer is used to help replace an existing sensor with one that is less expensive, the argument may be that for a modest investment in computational resources, sensor cost can be reduced. In other cases, it can be argued that observers increase value; for example, value could be increased by providing a more reliable feedback signal or a more accurate feedback signal that will lead to improved performance.

Those readers who are leading their companies in the use of observers should expect that they will have to demonstrate the practical advantages of observers if they want the methods to be adopted. Bear in mind that observers often produce undesirable characteristics, such as increased computational costs. At the very least, they require time to develop and training for staff or customers to learn new methods.
2.2.2 High Reliability

Control systems must be reliable. A proven way to enhance reliability is by reducing component count, especially connectorized cables. Electrical contacts are among the least reliable components in many systems. Observers can increase reliability when they are used to eliminate sensors and their cables.

Observers are not the only alternative for removing sensors. There is a wide variety of techniques to remove sensors, usually by measuring ancillary states; for example, the hard-disk-drive industry long ago began employing *sensorless* technology, eliminating commutation-position\(^1\) sensors in PC hard disks by measuring the electrical parameters of the motor driving the disk. This points out that *sensorless* is actually a misnomer; sensorless applications normally eliminate one sensor by relying on another. Still, the results are effective. In the case of sensorless hard drives, the position sensor and its cabling eliminated.

Observers offer a key enhancement for sensorless operation. The problem with most sensorless schemes is that the signals being measured usually have poor signal-to-noise characteristics, at least in some operating conditions. Returning to the example of a hard-disk controller, direct (that is, nonobserver-based) voltage measurement works well in the disk-drive industry where motor speeds are high so that the voltages created by the motor are relatively large. These same techniques work poorly at low speeds so that they cannot be used in many applications.\(^2\) Because observers combine the sensed signals (which may have high noise content) with the model signals (which are nearly noise free), they can remove noise from the calculated output, greatly extending the range of sensorless operation. So observers can be the best alternative to allow the elimination of sensors in some applications, and thus, they can be an effective way to simultaneously increase system reliability and reduce cost.

2.2.3 Stability

Control systems should remain stable in all operating conditions. The results of unstable operation are unpredictable; certainly, it is never desirable and in many cases, people may be injured or equipment damaged. In addition to maintaining absolute stability, systems must maintain reasonable margins of stability. For example, a temperature controller with low margins of stability may respond to a commanded

---

\(^1\)Note that this discussion relates to position sensing for commutation, the process of channeling current to produce torque in a motor. Commutation requires only coarse sensing, often just a dozen or so positions around the disk. Hard-disk drives use an additional track on the disk itself for the fine position sensing, which allows the much more accurate location of data on the disk surface.

\(^2\)This voltage, called the back-electro-motive force or back-EMF, is produced by motors in proportion to the moving magnetic field of the motor. In most cases, the back-EMF is proportional to the speed of the motor. Thus, at low speed, the back-EMF signal is low and noise has a greater effect.
temperature change of 5° by generating oscillatory changes of 5° or 10° that die out only after minutes of ringing. Such a system may meet an abstract definition of stability, but it would be unacceptable in most industrial applications. Margins of stability must be maintained so that performance can be predictable. Two common measures of stability, phase margin and gain margin, will be discussed in Chapter 3.

Observers can improve stability by reducing the phase lag within the control loop. For example, the process of converting a sensor signal often involves filtering or other sources of phase lag. In the motion-control industry, it is common to use the simple difference of two position samples to create a velocity signal. Such a process is well known to inject a time delay of half the sample time. By using an observer this phase delay can be removed. In applications requiring the highest performance, the removal of this phase lag can be significant.

### 2.2.4 Rapid Command Response

Command response measures how well the response follows a rapidly changing command. Most control systems follow slowly changing commands well but struggle to follow more rapidly changing signals. In most cases, it is considered an advantage for a control system to follow rapid commands accurately.

A key measure of system response is bandwidth. The bandwidth is defined as the frequency where the small signal response falls to 70.7% of the DC response. To find the bandwidth of a control system, create a sinusoidal command at a relatively low frequency and measure the amplitude of the response. Increase the frequency until the amplitude of the response falls to 70% of the low-frequency value; this frequency is the bandwidth.

The most common way to improve command response is to raise the gains of the control laws. Higher gains help the system follow dynamic commands but simultaneously reduce margins of stability. Tuning, the process of setting control-law gains, is often a compromise between command response and margins of stability. As discussed above, observers can increase margins of stability and thus allow incrementally higher gains in the control law.

### 2.2.5 Disturbance Rejection

Disturbance rejection is a measure of how well a control system resists the effect of disturbances. As with command response, higher gains help the system reject disturbances, but they reduce margins of stability. Again, tuning control-law gains requires a compromise of response and stability.

Observers can help disturbance rejection in two ways. As with command response, disturbance response can be improved incrementally through higher control-law gains when the observer allows the removal of phase lag. Second, as discussed in Section 2.1.5.1, observers can be used to observe disturbances, allowing the use of disturbance decoupling where it otherwise might be impractical.
2.2.6 Minimal Noise Response

Noise response is a measure of how much the control system responds to noise inputs. The problem may be in the plant response where the concern is that the noise unduly corrupts the system output. On the other hand, the concerns may be with noise generated by the power converter. Noise fed into the control law via the command, feedback, and control-laws calculations is transferred to the power converter where it can create high-frequency perturbations in the power output. That noise can be objectionable even if the plant filters the effect so much that it does not measurably affect the system response. For example, noise in a power supply may generate high-frequency current perturbations that cause audible noise. Such noise may make the noise generation unacceptable, even if final filtering components on the power supply output remove the effect of the noise on the power supply’s output voltage.

The concern with noise response is usually focused on response to high-frequency signals. High system gain is desirable at lower frequencies. A control system is expected to be responsive to signals at and below the system bandwidth. Well above the bandwidth, high gain becomes undesirable. The output does not respond to the input in any useful way (because it is greatly attenuated), but it still passes high-frequency noise, generating undesirable perturbations, audible noise, and unnecessary power dissipation. Lower gain at frequencies well above the bandwidth is equivalent to improved (reduced) noise response.

The first step to reducing noise response is reducing the amplitude of the noise feeding the control system. This may come by improving system wiring, increasing resolution of digital processes, or improving power supply quality to the sensors and control laws. After this path has been exhausted, the next step is usually to filter noise inputs. Filters are effective in reducing noise, but when filters are in the control loop, they add phase lag, reducing margins of stability; control-law gains often must be reduced to compensate. Since margins of stability must be maintained at an acceptable level, the end effect is that filtering often forces control-law gains down.

Observers can exacerbate problems with sensor-generated noise. One reason is that one of the primary benefits of observers is supporting increased control-law gains through the reduction of phase lag. The increase of control-law gains will directly increase the noise susceptibility of the typical control system. In addition, observers often amplify sensor noise above the bandwidth of the sensor. The details of this effect are complicated and will be explained in Chapter 7. For the present, readers should be aware that observers often will not work well in systems where sensor noise is a primary limitation.

2.2.7 Robustness

Robustness is a measure of how well a system maintains its performance when system parameters vary. The most common variations occur in the plant. As examples, the capacitance of a power supply storage capacitor may vary over time, the rotational
inertia of a mechanism may vary during different stages of machine operation, and the amount of fluid in a fluid bath may vary and change the thermal mass of the bath. The control system must remain stable and should maintain consistent performance through these changes. One challenge of observer-based techniques is that robustness can be reduced by their use. This is because observers rely on a model; when the plant changes substantially and the model is not changed accordingly, instability can result. Thus, robustness should be a significant concern any time observers are employed.

2.2.8 Easy Setup

Control systems should be easy to set up. One of the realities of modern industry is that the end users of control systems are often unfamiliar with the principles that make those systems work. This can be hard for control-system designers to accept. It limits the use of novel control methods because those people further down the product-use chain (for example, technicians, salespeople, and end users) may not fully understand why these methods are useful or how they should be configured. Certainly, observers fit into this class of solutions. In many cases, after they have been implemented, tested, and shown to be effective, they still must be clearly explained to be ultimately successful. In addition, designers must strive to keep observers easy to set up. Observers are software-based closed loops with control laws that must be tuned; as will be discussed, this process can be simplified by careful design.

2.3 Visual ModelQ Simulation Environment

When learning control-system techniques, finding equipment to practice on is often difficult. As a result, designers must often rely on computer simulations. To this end, the author developed Visual ModelQ, a stand-alone, graphical, PC-based simulation environment, as a companion to this book. The environment provides time-domain and frequency-domain analysis of analog and digital control systems. Visual ModelQ is an enhancement of the original ModelQ in that Visual ModelQ allows readers to view and build models graphically. More than two dozen Visual ModelQ models were developed for this book. These models are used extensively in the chapters that follow. Readers can run these experiments to verify results and then modify parameters and other conditions so they can begin to experiment with observers.

Visual ModelQ is written to teach control theory. It makes convenient those activities that are necessary for studying controls. Control-law gains are easy to change. Plots of frequency-domain response (Bode plots) are run with the press of a button. The models in Visual ModelQ run continuously, similar to the way real-time controllers run. The simulated measurement equipment runs independently so parameters can be changed and the effects seen immediately.
2.3.1 Installation of *Visual ModelQ*

*Visual ModelQ* is available at www.qxdesign.com. The unregistered version is available free of charge. While the unregistered version lacks several features, it can execute all the models used in this book. Readers may elect to register their copies of *Visual ModelQ* at any time; see www.qxdesign.com for details.

*Visual ModelQ* runs on PCs using Windows 95, Windows 98, Windows 2000, or Windows NT. Download and run the executable file setup.exe for *Visual ModelQ* V6.0 or later. Be aware that the original version of *ModelQ* is not compatible with *Visual ModelQ*. Note that *Visual ModelQ* comes with an online help manual. After installation, read this manual. Finally, check the Web site from time-to-time for updated software.

### 2.4 Software Experiments: Introduction to *Visual ModelQ*

The following section will review a few models to introduce the reader to *Visual ModelQ*.

#### 2.4.1 Default Model

When *Visual ModelQ* is launched, the default model is automatically loaded. The purpose of this model is to provide a simple system and to demonstrate a few functions. The default model and the control portion of the *Visual ModelQ* environment are shown in Figure 2-7.

The model compilation and execution are controlled with the block of three buttons at the upper left of the screen: compile (green circle), stop execution (black
square), and start execution (black triangle). These blocks, with the current execution
time (here, 9.16051 seconds), are shown in Figure 2-8. If a model must be compiled
before it can be run, the green circle will turn red. The circle will turn red at launch
and anytime either a block or a wire is added to or taken away from the model. Any
time a model is recompiled, the model timer will return to 0 seconds and all default
values of model blocks will be reloaded.

The default model is detailed in Figure 2-9. There are four blocks, two of which
are connected with a wire:

- **Solver**: The solver configures the differential-equation solver used to simulate
  system components. Note: One and only one solver is required for every
  model.
- **Scope**: The main scope provides a display for up to eight channels of input.
  The workings of the scope and its trigger mechanism are similar to those of a
  physical oscilloscope. Note: At least one scope is required for every model.
- **Waveform Generator**: The waveform generator can be used to generate
  standard waveforms such as sine waves and triangle waves. Frequency, ampli-
  tude, and phase are all adjustable while the model is running. The generator
  here is set to produce a square wave at 10 Hz.

![Figure 2-8. Compile and run controls.](image)

![Figure 2-9. Detail of default model.](image)
• **Live Scope**: The Live Scope displays its output on the block diagram. Live Scope variables automatically display on all main scope blocks as well. Notice in Figure 2-7 that a short wire connects the output of the waveform generator to the input of the scope; this connection specifies that the Live Scope should plot the output of the waveform generator.

### 2.4.1.1 Viewing and Modifying Node Values

Blocks have nodes, which are used to configure and wire the elements into the model. For example, the solver block, shown in Figure 2-10, has two nodes. There is a configuration node (a green diamond) at the left named \( h \). This node sets the sample time of the differential-equation solver. The sample time is set to 10 \( \mu s \) by default.

The solver block includes a documentation node (a rectangle) at the right. The documentation node, which is provided on almost all Visual ModelQ blocks, allows the user to enter notes about the block for reference. The name of the block, Solver in this case, is shown immediately below the block. The user can change the name of any Visual ModelQ block by positioning the cursor within the name and double-clicking.

There are several ways to read the values of nodes such as the \( h \) node of the sample block. The easiest is to use fly-over help. After the model is compiled, position the cursor over the node and the value will be displayed in a fly-over block for about a second, as shown in Figure 2-11.

The value of configuration nodes can be set in two ways. One way is to place the cursor over the node and double-click. The Change/View dialog box is then displayed as shown at the top right of Figure 2-12. The value can be viewed and changed from this dialog box.

The second way to set values is to use the Block set-up dialog box. Right-click in the body of the block; this brings up a pop-up menu as is shown center left in
2.4.1.2 The WaveGen Block

The WaveGen block has ten nodes, as shown in Figure 2-13. The nodes are:

- Waveform: Select initial value from several available waveforms such as sine or square waves.
- Frequency: Set initial frequency in Hertz.
- Amplitude: Set initial value of peak amplitude. For example, setting the amplitude to 1 produces an output of ±1.
- Enable: Allows automatic disabling of the waveform generator. When the value is 1, the generator is enabled. When 0, the generator is disabled. For digital inputs such as this node, Visual ModelQ considers any value greater than 0.5 to be equivalent to 1 (true); all values less than or equal to 0.5 are considered equivalent to 0 (false). This function will be especially useful when taking Bode plots since all waveform generators should be disabled in this case.
- Output: Output signal of waveform generator.
• Offset: Initial value by which the waveform generator output should be offset.
• Phase: Initial value of waveform phase, in degrees, of the waveform generator. For example, if the output is a sine wave, the output will be:

\[ \text{Output} = \text{Amplitude} \times \sin (\text{Frequency} \times 2\pi \times t + \text{Phase} \times \pi/180) + \text{Offset}. \]

• Duty cycle: Initial value of percentage duty cycle for pulse waveforms.
• Multiplier: Value by which to multiply waveform generator output. This is normally used for unit conversion. For example, most models are coded in Systeme International (SI) units. If the user finds RPM more convenient for viewing than the SI radians/second, the multiplier can be set to 0.105 to convert RPM (the user units) to radians/second (SI units). The multiplier node is present in most instruments such as scopes and waveform generators to simplify conversion to and from user to SI units.

The Enable node of the WaveGen block is an input node, as the inward-pointing triangle indicates. Input nodes can be changed while the model is running and they can be wired in the model. Neither of these characteristics is true of configuration nodes (those shaped like diamonds).

Using the block set-up dialog box can speed the setup of more complicated blocks such as the WaveGen. The WaveGen block set-up dialog is shown in Figure 2-14. The benefit of the block set-up dialog is that all of the parameters are identified by name and can be set one after the other. Notice that the first node in the dialog, Output, cannot be changed (the button at right allows only “View . . .”). This is necessary because some nodes, such as output nodes, cannot be configured manually.
The parameters of the waveform generator set in the nodes are only initial (precompiled) values. To change the configuration of the waveform generator when the model is running, double-click anywhere inside the block and bring up the real-time WaveGen control panel. This panel, shown in Figure 2-15, allows six parameters of the waveform to be changed while the model is running. The buttons marked “<” and

![Waveform Generator control panel]

**Figure 2-15.** Waveform generator control panel which is displayed by double-clicking on the WaveGen block after the model has been compiled.
“>” move the value up and down by about 20% for each click. Changing these values has no permanent effect on the model; each time the model is recompiled, these values will be returned to the initial values as specified by the nodes.

### 2.4.1.3 The Scope Block

The Scope block, with a list of its nodes, is shown in Figure 2-16. Most of the nodes set functions that are consistent with laboratory oscilloscopes and thus will be familiar to most readers. One node that should be discussed is the **Trigger Source** node. This node sets the initial variable that will trigger the scope when the scope mode is set to *Auto* or *Normal*. If this variable is not set prior to compiling the model, a warning will be generated. To eliminate this warning, simply double-click on the node and select a variable from a drop-down list to trigger the scope. Choose from any **Variable** or **Live Scope**, as shown in Figure 2-16.

The scope display is normally not visible. However, it can be made visible by double-clicking inside the scope block after the model has been compiled. The block can be made not visible by clicking the “X” icon at the top right of the scope window.

The scope display provides two tabs: *Scale* and *Trigger*. The *Scale* tab (shown in Figure 2-17) provides control of the horizontal and vertical scaling. The *Trigger* tab provides various trigger settings. At the bottom of the scope there are a few controls. Starting at the bottom left of Figure 2-17:

![Graphical representation of the Scope Block](image)

**Figure 2-16.** The **Trigger Source** of a Scope can be set to any variable (such as **Variable6**) or any **Live Variable** (such as **LiveVariable3**).
• the Trig button flashes green for each trigger event;
• the sunglasses button hides the control panel at left, maximizing the display area of the plot;
• the single-shot check box enables single-shot mode;
• the scale-legend control button turns the scale legend (immediately below the plot) on and off;
• the three cursor buttons select 0, 1, or 2 cursors.

Note that single-shot mode stops the model from running after the scope screen has filled up. Restart the model using the Run (black triangle) button after each single-shot event.

2.4.1.4 The Live Scope Block

The default model also includes a Live Scope block, as shown in Figure 2-18. The input comes in at top left, with the scale, offset, and time scale set in the nodes just below that. The Show node determines whether the variable in the Live Scope is displayed in the main scopes after each compile (note that variables that display in a
**Chapter 2 Control-System Background**

Live Scope also can be displayed in any main scope block). The Mult node specifies a multiplier, which scales the variable before plotting.

The next five nodes are trigger nodes. The Trigger Source node specifies the signal that triggers the Live Scope. If this variable is unwired, the Input (first) node will be used as the trigger. Most of the remaining nodes have equivalent functions on standard oscilloscopes except the last two nodes, Width and Height, which set the size of the Live Scope block in pixels.

Live Scopes provide simple display features compared to the main scope block, and there are several limitations. No more than two channels can be displayed using a Live Scope. There are fewer trigger options. Another limitation is that Live Scopes only show input vs time; there is no option for Input1 vs Input2 (x vs y) as there is for the main Scope blocks.

The Live Scope also has several advantages. First, the wiring to a Live Scope makes it clear which variable is being plotted; this makes the display more intuitive, especially in larger models. Second, because the result is displayed on the model, it is often easier to convey information to others using the Live Scope. It is this reason that caused the author to prefer the Live Scope to the standard scope throughout this book. Finally, almost all of the Live Scope parameters are input nodes, and all input nodes can be wired into the circuit. This means that a model can be constructed to automatically change those values as the model executes.

### 2.4.2 Experiment 2A: Simple Control System

The remainder of this chapter will discuss three experiments written to introduce the reader to control-system modeling in *Visual ModelQ*. Experiment 2A is a simple control system. The model diagram is shown in Figure 2-19. The model is comprised of several elements:
A waveform generator, which produces the command.
A summing junction, which compares the command and the feedback (output from the feedback filter) and produces an error signal.
A PI control law, which is configured with two `Live Constants`, a proportional gain, $K_P$, and an integral gain, $K_I$. These blocks will be discussed shortly.
A filter simulating the power converter. The power converter is a two-pole low-pass filter set for a bandwidth of 800 Hz and with a zeta (damping ratio) of 0.707.
An integrating plant with an intrinsic gain of 500.
A filter simulating the feedback conversion process. The feedback filter is a two-pole low-pass filter set with a bandwidth of 350 Hz and with a zeta of 0.707.
A two-channel `Live Scope` that plots command (above) against actual plant output (below).
A solver and scope, both of which are required for a valid `Visual ModelQ` model.

### 2.4.2.1 Visual ModelQ Constants: Many Ways to Change Parameters

`Visual ModelQ` provides numerous ways to change model parameters. Of course, any unwired node can be changed by double-clicking on a node or right-clicking and bringing up the `Block set-up` dialog box (see Figure 2-12). However, numerous blocks are provided to simplify the task of changing node values.
The Constants tab in the Visual ModelQ environment (top of Figure 2-7) currently provides seven constant types: simple constants, standard and inverse Live Constants, simple scaling constants, standard and inverse scaling Live Constants, and string constants. The selection buttons for each of these constants are shown in Figure 2-20, which is a screen capture of the top portion of the Visual ModelQ environment.

Live Constants, such as $K_P$ and $K_I$ in the PI controller of Figure 2-19, provide the most control. The icons of blocks have a “<<>>” symbol. After the model has compiled, double-click anywhere inside the block and the adjustment box of Figure 2-21 will appear. Using the adjustment box, the value of the parameter can be changed while the model runs. A new value can be typed in with the keyboard by clicking the cursor in the value edit box. (Note that when using the keyboard, the new value does not take effect until the enter key is hit.) In addition, there are six logarithmic adjustment buttons in the adjustment box. The double less-than block (<<) reduces the value to the next lowest value with the first digit being 1, 2, or 5. For example, if the value of the variable is 1.75, clicking “<<” will change the value to 1, clicking again will reduce it to 0.5, clicking again will reduce it to 0.2, and so on. Each click reduces the value approximately by half. The double greater-than (>>) performs a similar function except it moves to the next higher value: 1, 2, 5, 10, 20, and so on.
The remaining adjust buttons are straightforward. The bold single less-than button reduces the value of the variable by about 20% for each click; the nonbold single less-than button reduces the value by about 4%. The bold and nonbold single greater-than blocks perform a similar function, only raising the value. If the parameter can take on values of both signs, the +/- button will be enabled, allowing a change in sign at the click of a button.

The *Live Constant* model block and its nodes are shown in Figure 2-22. The initial value node specifies the value that the constant is reset to after each compile. The minimum and maximum nodes specify the range that the input can take on. The multiplier and documentation nodes are standard *Visual ModelQ* nodes. The output makes available the value of the *Live Constant* so it can be wired in the model. The value displayed in text inside the block is not scaled by the *Mult* node, while the value in the output node is.

### 2.4.2.2 Inverse Live Constants

The inverse *Live Constant* works like the standard *Live Constant* except the output is one divided by the parameter value and then multiplied by the value of the *Mult* node (Figure 2-23). This constant is used when the model needs to scale by the inverse (1/x) of the parameter value such as is usually the case for mass, moment of inertia, thermal mass, capacitance, inductance, and many other physical parameters. The inverse *Live Constant* is a space-saving alternative to combining a standard *Live Constant* and a 1/x block.

---

**Figure 2-22.** Detail of a *Live Constant.*

**Figure 2-23.** Inverse *Live Constant* generates divided output.
2.4.2.3 Scale-by Live Constants

Scale-by Live Constants are similar to Live Constants except that the output node is the product of an input node and the value of the constant. In fact, if the input node is set to one, scale-by Live Constants behave identically to standard Live Constants. The two scale-by Live Constants (standard and inverting) are shown in Figure 2-24.

2.4.2.4 String Constants

String constants allow the model constants to be adjusted as strings. The user selects a string from a list and the string Live Constant block outputs an integer value according to the position in the list occupied by the selected string. For example, the string Live Constant named Select XN in Figure 2-25 is configured to allow the user to select one of four strings: X0, X1, X2, and X3. Depending on which constant the user selects, the output node will be 0, 1, 2, or 3, according to the position of the string within the list.

The string Live Constant node is configured with two input nodes on the left side of the block. The Strings node should be filled first; double-click on this node and type in a list of string constants. The Strings node dialog box for the block of Figure 2-25 is shown in Figure 2-26. There is no specific limit on constant length or on the number of strings that one string Live Constant can hold. Next, select the Live Constant’s initial string by double-clicking on the upper left node.

The string Live Constant is often used with an analog switch, as is the case in Figure 2-25. The switch has a control node at top center, the value of which determines which of the four inputs at left is routed to the output: moving from top to bottom, 0 selects the first input, 1 the second, and so on. In the case of Figure 2-25, Select XN is equal to X1, as is indicated inside the string Live Constant block. This produces an output of 1, which is fed to the switch control node. That causes the Position-1 (second) input node to be connected to the output node of the switch block. Figure 2-25 also has an Inspector block, which can display the value of any node. Here, the inspector shows that output of Switch4 is 10.01, matching the value of the Live Constant X1 which is connected to the Position-1 input node. Note that the naming of the four Live Constants at left to match the list of strings in Select XN is for clarity and has no effect on the operation of the model.
2.4.2.5 Simple Constants

The last Visual ModelQ constants are the simple constants. These constants are similar to the Live Constant. However, the simple constants do not support the adjustment box of Figure 2-21; changes to the value are made via double-clicking on the input.

Figure 2-25. A string Live Constant outputs an integer based on user-selected character strings.

2.4.2.5 Simple Constants

The last Visual ModelQ constants are the simple constants. These constants are similar to the Live Constant. However, the simple constants do not support the adjustment box of Figure 2-21; changes to the value are made via double-clicking on the input.

Figure 2-26. The user types in a string list to configure the string Live Constant block.
node. (Note that changing a node value is permanent after the model is saved.) Also, neither maximum nor minimum limits can be set. The simple constants take a little less screen space than a Live Constant in the model diagram. Use the simple constant for parameters that are changed only occasionally. The simple constant is shown in Figure 2-27; the simple scale-by constant adds scaling.

2.4.2.6 Hot Connections on a Live Scope

The Live Scope supports a feature called hot connection. Anytime the model is running, double-click on the Live Scope and a Live Scope control panel will appear. Click “Hot Connect” to close the dialog box; move the mouse over any wire or input–output node in the model and click. The Live Scope will temporarily graph the value of that node or wire; the scope outline will turn green to indicate that the scope is in hot connect mode. (For two-channel Live Scopes, Channel 1 displays the hot connection; Channel 2 is unaffected.) Click “Restore Scope” in the Live Scope control panel or recompile the model to restore the scope to its original display. Note that the scope scale and offset nodes may need to be adjusted to view the signal; any changes to scaling and offset will be restored when the scope is restored. The operation of hot connection is displayed in Figure 2-28. Hot connections are especially useful when debugging a model, as wires and nodes can be viewed without adding a scope, which forces recompilation.

2.4.3 Command Response and Control-Law Gains

Visual ModelQ is designed to simplify the process of evaluating the effects of parameter value variation. This is a common need when modeling control systems, for example, in the tuning process. Tuning is the adjustment of control-loop gains to achieve optimal performance. It is often carried out in working systems (and in models) by observing the effect of numerous incremental changes of control-law gains. For example, in Experiment 2A, $K_p$ might be adjusted up and down in small steps while observing the effect on the step response. Experiment 2A is constructed to make this process fast and simple.

The two components of Experiment 2A that simplify tuning are the Live Constant and the Live Scope. After model compilation, double-clicking on the Live Constant named $K_p$ brings up the $K_p$ adjustment box, which allows rapid changes of value,
perhaps one per second. Compare this to standard modeling environments where the model must be stopped, modified, and recompiled. A simple change can take on the order of a minute. In addition, the Live Scope gives immediate feedback of the effect of the new parameter, without the need for the user to issue a command to display a plot. To experiment, launch Visual ModelQ. Click File, Open... to open the model Experiment_2A.mqd. Click Run. Double-click on the $K_p$ block and use the << and >> buttons to move the value up and down. The results should be equivalent to those shown in Figure 2-29.

2.4.4 Frequency Domain Analysis of a Control System

Control systems often need to be analyzed in the frequency domain. The most intuitive method of frequency-domain analysis for most people is the Bode plot, which graphs gain and phase across a range of frequencies. A gain plot displays the amplitude of an output signal divided by the amplitude of the input signal at many frequencies as if sine waves at many frequencies had been applied to the model. A phase plot displays the time lag of the output compared to the input for many sine waves. In the laboratory, the instrument that is commonly used to generate Bode plots is called a dynamic signal analyzer (DSA). Visual ModelQ provides a DSA, which is used regularly in Chapters 4 through 8. Experiment 2B, shown in Figure 2-30, is
Experiment 2A modified to include a DSA, which is shown just right of the waveform generator.

### 2.4.4.1 The Visual ModelQ DSA

The DSA is wired in line with the excitation path. In most cases, the DSA is used to analyze command response and so will normally be inserted in line with the command as it is in Figure 2-30. All DSAs read all model variables, no matter how they are wired. In Visual ModelQ, the term variables includes three types of signals:

![Diagram of Visual ModelQ](image)

**Figure 2-30.** Experiment 2B: Experiment 2A with a DSA.
• The input to 1-channel Live Scopes,
• The input to Channel 1 of 2-channel Live Scopes, such as Command in Figure 2-30, and
• ModelQ variables blocks such as Feedback in Figure 2-30.

The DSA here will be used to show the relationship between command and feedback. Notice that Experiment 2B required the addition of the variable block Feedback at top right. In Experiment 2A that node was not connected to a variable block as it was only needed for display as Channel 2 of a Live Scope. In Experiment 2B, an explicit variable block named Feedback is required to grant access of the signal to the DSA.

2.4.4.2 DSA Nodes

The complete details on configuring a DSA go beyond the scope of this chapter. However, a few details should be mentioned to prepare the reader for the use of DSAs in this book. The four most important nodes of a DSA block are shown in Figure 2-31. At left is the input node. Normally, the DSA is inactive and the input node passes directly to the output node. However, when the user wants a new Bode plot, the DSA is commanded to excite the model. This temporarily disconnects the input node and replaces it with a random signal excitation. The Excitation Amplitude and DSA Inactive nodes will be discussed in Section 2.4.4.5.

2.4.4.3 The DSA Display

A Bode plot from a DSA is shown in Figure 2-32. This shows the relationship between command and feedback, commonly called the closed-loop response, for Experiment 2B where \( K_p = 1 \) and \( K_p = 2 \); the gain plots are above and the phase plots below. Most of the time, the closed-loop gain plot will be of primary interest. The two cases here behave similarly at low frequency (shown at left) and the plots below about 100 Hz are nearly indistinguishable. However, above 100 Hz, there are significant differences, especially where the gain of the \( K_p = 2 \) case sharply rises before falling, displaying an undesirable characteristic called peaking. The purpose of this section is to introduce Visual ModelQ, so a detailed discussion of resulting waveforms is outside the scope.
of this discussion. However, it may be interesting to readers to notice that the two cases plotted in Figure 2-32 match the time-domain plots for Figures 2-29a and 2-29b, where the less stable Figure 2-29b corresponds to the plot in Figure 2-32 with peaking. Peaking and ringing are both indicators of inadequate margins of stability. Stability issues will be discussed in Chapter 3.

Like the Scope display, the DSA display is normally not visible when a model starts to run. The DSA display can be made visible by double-clicking inside the DSA block after the model has been compiled.

### 2.4.4.4 DSA Controls

The user can request a new Bode plot when the model is running by clicking on the GO button at bottom left of the DSA. This starts a new excitation period. For the experiments in this book, this process will continue for roughly 1 s of simulation time. After that, a new Bode plot will be displayed. Up to four plots can be saved. Right-click in the graph area of the DSA to bring up a pop-up menu and select Save as to save the most recent plot. Pressing the GO button a second time during the excitation period cancels the command for a new Bode plot.

To the right of the GO button, the sunglasses button hides the control panel. The gear button brings to view a dialog box for setting up the DSA excitation signal. The
autofind button places a cursor according to the criteria in the adjoining combo box, which is set to 3 dB in Figure 2-32. The last three buttons control the number of cursors visible, allowing no cursors, one cursor, or two cursors.

2.4.4.5 The DSA Excitation Signal

The DSA works by generating a random command for a short period of time. The random signal is rich—it contains all the frequencies of the Bode plot. During the period of excitation, the DSA monitors all variables in the model. After the excitation, the DSA executes a fast Fourier transform (FFT) to convert the recorded data to a frequency-domain plot. When the random signal is applied to the model, the richness of the signal allows it to excite all frequencies at once. This is ideal for a modeling environment because it minimizes the time the DSA must excite the system. However, it also presents problems. First, the system must remain out of saturation—the power converter must not be driven beyond its maximum during the excitation. If a system is driven into saturation, the excitation amplitude can be reduced using the Excitation Amplitude node at the top left of the DSA (see Figure 2-31). However, if the amplitude is set too low, the signal-to-noise ratio of the system will be insufficient and the Bode plot will be distorted at high frequencies. Setting the amplitude of the excitation is sometimes a matter of experimentation. When doing so, always monitor the power converter output to ensure the system remains out of saturation for the entire excitation period. For all models in this book, the amplitude is set appropriately and users normally need not be concerned about this.

All commands except the DSA excitation must be shut off during the excitation period. The DSA will automatically disconnect the input node so that any signals connected to the input are disabled during DSA excitation; this is the case with the waveform generator in Figure 2-30. If there are waveform generators connected to other parts of the model, the DSA Inactive node at the lower right of Figure 2-31 can be wired to disable those generators. The DSA Inactive node is set to zero during the excitation period; when wired to a waveform generator Enable node, the desired behavior is realized.

2.4.5 Modeling Digital Control Systems

Experiment 2C, the final model of this chapter, will demonstrate how to model a simple digital control system in Visual ModelQ. This model, shown in Figure 2-33, is similar to Experiment 2B except that three blocks have been added. First, the PI controller, just below $K_p$ and $K_i$, is now digital. The border area of this block is yellow in the Visual ModelQ environment and prints gray in the monochrome Figure 2-33.

Digital PI controllers sample the error at regular periods of time. The sample period for digital blocks is set via the controller node, the diamond at the bottom left of the PI block. The controller can be selected from multiple digital controllers, which
can be running simultaneously in a *Visual Model* model. Fortunately, most models are simple enough that one controller is sufficient. That controller is called *Main* in Experiment 2C and is near the center-left of Figure 2-33. The sole input node of the controller block is the sample time, which can be changed while the model is running. In Experiment 2C, that parameter is connected to a *Live Constant* named *TSample* to simplify changing the value.

Notice that the step response in Figure 2-33 overshoots and rings in Experiment 2C. All the parameters of Experiments 2B and 2C have identical defaults so one might have expected them to have a similar step response. Obviously, something is significantly different.

The difference between the two models is that Experiment 2C is the digital equivalent of Experiment 2B. The problem in Experiment 2C is that the sample time is too long for the dynamics of the system. As a result, the system is nearly unstable. Some experimentation can prove the point. Launch *Visual Model* and load the file *Experiment_2C.mqd*. Click *Run*. Now, double-click on the *Live Constant* named *TSample*. Reduce the sample time by repeatedly clicking on the *Live Constant* “<<” button. When the sample time falls below about 0.0002 s, the response is equivalent to the analog performance. This is shown in Figure 2-34.

### 2.4.6 *Visual Model* and This Book

This section has introduced several functions of *Visual Model* used in this book. All key points of this book are demonstrated in *Visual Model* models. Readers are encouraged to run these experiments and work the exercises at the end of each chapter.
2.5 Exercises

1. Open **Experiment_2A.mqd** and click the Run button.
   A. Change the gain $K_p$ from 1 to 2, and then raise it to 5. Describe what happens in the command response. What conclusion could you draw?
   B. Set $K_p=2$ and change waveform to triangle. Are signs of marginal stability easier or harder to recognize? Repeat for sine wave and s-curve. What conclusion could you draw?
   C. Restore $K_p$ to 1. Set $K_I$ to 0. Describe what happens in the command response. Set $K_I$ to a range of values from 10 to 1000. Describe what happens in the command response. What conclusion could you draw?

2. Open **Experiment_2B.mqd** and click the Run button.
   A. Run a Bode plot. Find the $-3$ dB frequency (the frequency where the gain falls to $-3$ dB) using the autofind combo box at the bottom of the DSA display window.
   B. Reduce control-loop gains. Set $K_p$ to 0.5 and set $K_I$ to 50. What is the gain at the frequency from 2A.
   C. Compare 2A and 2B. What conclusion could you draw?

3. Open **Experiment_2C.mqd** and click the Run button.
   A. Change $T_{Sample}$ to several values spanning the range between 0.002 and $1 \times 10^{-3}$ s. Over what range does the sample time significantly affect command response as viewed in the Live Scope?
   B. Does faster sampling make the system more stable or less stable?
   C. Set the sample time to 0.0001 s. Compare the step response of the digital system in Experiment 2C to the analog system in Experiment 2B. Repeat with $K_p=2$. What conclusion could you draw?

---

Figure 2-34. From Experiment 2C: Reducing sample time can stabilize a system. (a) $T_{Sample} = 0.002$ s; (b) $T_{Sample} = 0.0002$ s.
In this chapter...

- Overview of the $s$-domain and the $z$-domain
- Detailed presentation of Mason’s signal flow graphs
- Bode plots
- Measuring command response and stability
- The open-loop method
- A zone-based tuning procedure

This chapter will review the frequency domain, which is the basis for most analysis performed on control systems. The principles reviewed in this chapter are commonly taught in control-systems books, courses, and seminars so that many readers will find much of it familiar. In addition to the review, the final section provides a process for consistent tuning of controller gains; this process will be necessary to measure performance objectively, for example, when comparing traditional and observer-based systems. In addition, the same process will be applied to tuning observers in later chapters. For reference, most of this discussion is taken from [11, Chaps. 2–5].

### 3.1 Overview of the $s$-Domain

The Laplace transform underpins classic control theory [17, 37] and is defined in Equation 3.1 [7, p. 102] as
\[ F(s) = \int_0^\infty f(t)e^{-st}dt, \quad (3.1) \]

where \(f(t)\) is a function of time, \(s\) is the Laplace operator, and \(F(s)\) is the transformed function. The terms \(F(s)\) and \(f(t)\), commonly known as a transform pair, represent the same function in the two domains. For example, if \(f(t) = \sin(\omega t)\), then \(F(s) = \omega/(\omega^2 + s^2)\). The Laplace transform moves functions between the time and the frequency domains. The most important benefit of the Laplace transform is that it provides \(s\), the Laplace operator, and through that the frequency-domain transfer function.

### 3.1.1 Transfer Functions

Frequency-domain transfer functions describe the relationship between two signals as a function of \(s\). For example, consider an integrator as a function of time. From Table 3-1, the integrator has an \(s\)-domain transfer function of \(1/s\). So, it can be said for a system that produced an output, \(V_o\), which was equal to the integral of the input, \(V_i\), that:

\[ \frac{V_o(s)}{V_i(s)} = \frac{1}{s}. \quad (3.2) \]

The Laplace operator is a complex (as opposed to real or imaginary) variable. It is defined as

\[ s = \sigma + j\omega. \quad (3.3) \]

The constant \(j\) is \(\sqrt{-1}\). The \(\omega\) term translates to a sinusoid in the time domain; \(\sigma\) translates to an exponential \((e^{\sigma t})\) term. The primary concern here is with steady-state sinusoidal signals, in which case \(\sigma = 0\). So in this book, \(\sigma\) will be ignored. To evaluate the DC response of a transfer function, set \(s\) to zero.

### 3.1.2 Linearity and the Frequency Domain

A frequency-domain transfer function is limited to describing elements that are linear and time invariant. These are severe restrictions and, in fact, virtually no real-world system fully meets them. The three criteria that follow define these attributes, the first two defining linearity and the third defining time invariance.

1. **Homogeneity.** Assume that an input to a system \(r(t)\) generates an output \(c(t)\). For an element to be homogeneous, an input \(k \times r(t)\) would have to generate an output \(k \times c(t)\), for any value of \(k\). An example of homogeneous behavior is an ideal resistor where \(V = IR\). An example of nonhomogeneous behavior is saturation where twice as much input delivers less than twice as much output.
2. **Superposition.** Assume that an element, when subjected to the input \( r_1(t) \) will generate the output \( c_1(t) \). Further, assume that this same element, when subjected to the input \( r_2(t) \) will generate the output \( c_2(t) \). Superposition requires that if the element is subjected to the input, \( r_1(t) + r_2(t) \), it will produce the output, \( c_1(t) + c_2(t) \) [16, p. 93; 36].

3. **Time invariance.** Assume that an element has an input \( r(t) \) that generates an output \( c(t) \). Time invariance requires that \( r(t - \tau) \) will generate \( c(t - \tau) \) for all \( \tau > 0 \).

So the controls engineer faces a dilemma: transfer functions, the basis of classic control theory, require linear, time invariant (LTI) systems, but no real-world system is completely LTI. This is a complex problem that is dealt with in many ways. However, for most control systems, the solution is simple: design components close enough to being LTI that the non-LTI behavior can be ignored or avoided.

### 3.1.3 Examples of s-Domain Transfer Functions

Examples of transfer functions used in control laws are shown in Table 3-1. These functions can all be derived from Equation 3.1.

Integration and differentiation are the simplest operations. The \( s \)-domain operation of integration is \( 1/s \) and of differentiation is \( s \). Filters are commonly used by control-systems designers such as when low-pass filters are added to reduce noise. Table 3-1 lists the \( s \)-domain representation for a few common examples. A compensator is a specialized filter: one that is designed to provide a specific gain and phase shift at one frequency. The effects on gain and phase either above or below that

<table>
<thead>
<tr>
<th><strong>Operation</strong></th>
<th><strong>Transfer function</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration</td>
<td>( 1/s )</td>
</tr>
<tr>
<td>Differentiation</td>
<td>( s )</td>
</tr>
<tr>
<td>Delay</td>
<td>( e^{-sT} )</td>
</tr>
<tr>
<td>Simple filters</td>
<td></td>
</tr>
<tr>
<td>Single-pole low-pass filter</td>
<td>( K/(s+K) )</td>
</tr>
<tr>
<td>Double-pole low-pass filter</td>
<td>( \omega^2/(s^2 + 2\zeta\omega s + \omega^2) )</td>
</tr>
<tr>
<td>Notch filter</td>
<td>( (s^2 + \omega^2)/(s^2 + 2\zeta\omega s + \omega^2) )</td>
</tr>
<tr>
<td>Bilinear-quadratic (bi-quad) filter</td>
<td>( (s^2 + 2\zeta\omega s + \omega^2)/(s^2 + 2\zeta\omega_D s + \omega_D^2) )</td>
</tr>
<tr>
<td>Compensators</td>
<td></td>
</tr>
<tr>
<td>Lag</td>
<td>( K(\tau_p s + 1)/(\tau_p s + 1), \tau_p &gt; \tau_z )</td>
</tr>
<tr>
<td>PI</td>
<td>((K_p s + 1)K_p)</td>
</tr>
<tr>
<td>PID</td>
<td>((K_p s + 1 + K_D)/K_p)</td>
</tr>
<tr>
<td>Lead</td>
<td>(1 + K_p s((\tau_p s + 1) or [(\tau_D + K_D) s + 1])(\tau_D s + 1))</td>
</tr>
</tbody>
</table>
frequency are secondary. Table 3-1 shows a lag compensator, a proportional-integral (PI) compensator, and a lead compensator.

Delays add time lag without changing amplitude. Since microprocessors have inherent delays for sampling, the delay function is especially important when analyzing digital controls. A delay of $T$ seconds is defined in the time domain as

$$c(t) = r(t - T). \quad (3.4)$$

The corresponding function in the frequency domain is

$$T_{\text{Delay}}(s) = e^{-sT}. \quad (3.5)$$

### 3.1.4 Block Diagrams

Block diagrams are graphical representations developed to make control systems easier to understand. Blocks are marked to indicate transfer functions. In North America, transfer functions are usually indicated with their $s$-domain representation. The convention in Europe is to use schematic representation of a step response; Appendix C provides a listing of many North American and European block-diagram symbols.

Block diagrams can be simplified by combining blocks. Two blocks in parallel can be combined as their sum; two blocks in series can be represented as their product. When blocks are arranged to form a loop, they can be reduced using the $G/(1 + GH)$ rule. The forward path is $G(s)$ and the feedback path is $H(s)$. The transfer function of the loop is $G(s)/(1 + G(s)H(s))$ as shown in Figure 3-1.

The $G/(1 + GH)$ rule can be derived by observing in Figure 3-1a that the error signal $E(s)$ is formed from the left as:

$$E(s) = R(s) - C(s) \times H(s).$$

$E(s)$ can also be formed from the right side (from $C(s)$ back through $G(s)$) as

$$E(s) = C(s)/G(s).$$

![Figure 3-1. Simple feedback loop in equivalent forms.](image-url)
So,

\[ R(s) - C(s) \times H(s) = \frac{C(s)}{G(s)}. \]

One or two steps of algebra produce:

\[ \frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)}. \]  

(3.3b)

### 3.1.4.1 Mason's Signal Flow Graphs

An alternative to the \( G/(1 + GH) \) rule developed by Mason [10, p. 69; 7, p. 162; 28; 29; 31] provides graphical means for reducing block diagrams with multiple loops. The formal process begins by redrawing the block diagram as a signal flow graph.\(^1\) The control system is redrawn as a collection of nodes and lines. Nodes define where three lines meet; lines represent the \( s \)-domain transfer function of blocks. Lines must be unidirectional; when drawn, they should have one and only one arrowhead. A typical block diagram is shown in Figure 3-2, and its corresponding signal flow graph is shown in Figure 3-3.

**Step-by-step procedure.** This section will present a step-by-step procedure to produce the transfer function from the signal flow graph based on Mason’s signal flow graphs. The signal flow graph of Figure 3-3 will be used for an example. This graph has two independent inputs, \( R(s) \) and \( D(s) \). The example will find the transfer function from these two inputs to \( D_O(s) \).

---

\(^1\) For convenience, this step will be omitted in most cases and the block diagram will be used directly.

---

**Figure 3-2.** An example control-loop block diagram.
Step 1: Find the loops. Locate and list all loop paths. For the example of Figure 3-3, there is one loop:

\[ L_1 = -G_{PEst}(s) \times G_{SEst}(s) \times G_{CO}(s). \]

Step 2: Find the determinant of the control loop. Find the determinant, \( \Delta \), of the control loop, which is defined by the loops:

\[ \Delta = 1 - (\text{sum of all loops}) \]
\[ + (\text{sum of products of all combinations of two nontouching loops}) \]
\[ - (\text{sum of products of all combinations of three nontouching loops}) \]
\[ + \ldots. \]

Two loops are said to be touching if they share at least one node. For this example there is only one loop:

\[ \Delta = 1 + G_{PEst}(s) \times G_{SEst}(s) \times G_{CO}(s). \]

Step 3: Find all the forward paths. The forward paths are all the different paths that flow from the inputs to the output. For the example of Figure 3-3, there is one forward path from \( D(s) \) to \( D_o(s) \) and two from \( R(s) \):

\[ P_1 = D(s) \times G_P(s) \times G_i(s) \times G_{CO}(s) \]
\[ P_2 = R(s) \times G_P(s) \times G_i(s) \times G_{CO}(s) \]
\[ P_3 = R(s) \times G_{PEst}(s) \times G_{SEst}(s) \times -1 \times G_{CO}(s). \]
Step 4: Find the cofactors for each of the forward paths. The cofactor ($\Delta_k$) for a particular path ($P_k$) is equal to the determinant ($\Delta$) less loops that touch that path. For the example of Figure 3-3, all cofactors are 1 because every forward path includes $G_{CO}(s)$, which is in $L_1$, the only loop.

$$\Delta_1=\Delta_2=\Delta_3=1$$

Step 5: Build the transfer function. The transfer function is formed as the sum of all the paths multiplied by their cofactors, divided by the determinant:

$$T(s) = \frac{\sum_k (P_k\Delta_k)}{\Delta}.$$  \hspace{1cm} (3.6)

For the example of Figure 3-3, the signal $D_o(s)$ is

$$D_o(s) = R(s) \frac{(G_p(s)G_s(s) - G_{PEst}(s)G_{SEst}(s)G_{CO}(s))}{1 + G_{PEst}(s)G_{SEst}(s)G_{CO}(s)} + D(s) \frac{G_p(s)G_s(s)G_{CO}(s)}{1 + G_{PEst}(s)G_{SEst}(s)G_{CO}(s)}.$$ 

Using a similar process, $C_o(s)$ can be formed as a function of $C(s)$ and $D(s)$:

$$C_o(s) = R(s) \frac{G_{PEst}(s)(1 + G_p(s)G_s(s)G_{CO}(s))}{1 + G_{PEst}(s)G_{SEst}(s)G_{CO}(s)} + D(s) \frac{G_{PEst}(s)G_p(s)G_s(s)G_{CO}(s)}{1 + G_{PEst}(s)G_{SEst}(s)G_{CO}(s)}.$$ 

As will be discussed in later chapters, a great deal of insight can be gained from transfer functions of this sort. Using Mason’s signal flow graphs, transfer functions of relatively complex block diagrams can be written by inspection. Using the $G/(1 + GH)$ rule to derive transfer functions from multiple-loop block diagrams will work but is more tedious.

It may be of interest that Figure 3-2 is an observer. The two blocks above represent the plant ($G_p(s)$) and the sensor ($G_s(s)$); those below are the approximations of those functions: the estimated plant, $G_{PEst}(s)$, and the estimated sensor, $G_{SEst}(s)$. The error between the actual and the estimated functions is fed through the observer compensator, $G_{CO}(s)$, which has high gains and will drive the output of the model (that is, the estimated plant and sensor) toward the output of the actual system. This form and its associated transfer functions will be the subject of the remaining chapters of this book.

3.1.5 Phase and Gain

The sine wave is unique among repeating waveforms; it is the only waveform that does not change shape when passing through LTI blocks. A sine-wave input generates a sine-wave output at the same frequency; the only differences possible between input