Introduction to Control Systems

What are control systems? Why do we study them? How do we identify them? The chapters in this section should answer these questions and more.

Introduction

What are Control Systems?

The study and design of automatic **Control Systems**, a field known as **control engineering**, is a large and expansive area of study. Control systems, and control engineering techniques have become a pervasive part of modern technical society. From devices as simple as a toaster, to complex machines like space shuttles and rockets, control engineering is a part of our everyday life. This book will introduce the field of control engineering, and will build upon those foundations to explore some of the more advanced topics in the field. Note, however, that control engineering is a very large field, and it would be foolhardy of any author to think that they could include all the information into a single book. Therefore, we will be content here to provide the foundations of control engineering, and then describe some of the more advanced topics in the field.

Control systems are components that are added to other components, to increase functionality, or to meet a set of design criteria. Let's start off with an immediate example:

> We have a particular electric motor that is supposed to turn at a rate of 40 RPM. To achieve this speed, we must supply 10 Volts to the motor terminals. However, with 10 volts supplied to the motor at rest, it takes 30 seconds for our motor to get up to speed. This is valuable time lost.

Now, we have a little bit of a problem that, while simplistic, can be a point of concern to people who are both designing this motor system, and to the people who might potentially buy it. It would seem obvious that we should increase the power to the motor at the beginning, so that the motor gets up to speed faster, and then we can turn the power back down to 10 volts once it reaches speed.

Now this is clearly a simplisitic example, but it illustrates one important point: That we can add special "Controller units" to preexisting systems, to increase performance, and to meet new system specifications. There are essentially two methods to approach the problem of designing a new control system: the **Classical Approach**, and the **Modern Approach**.

It will do us good to formally define the term "Control System", and some other terms that are used throughout this book:

> Control System
>
> A Control System is a device, or a collection of devices that manage the behavior of other devices. Some devices are not controllable. A control system is an interconnection of components connected or related in such a manner as to command, direct, or regulate itself or another system.
>
> Controller
>
> A controller is a control system that manages the behavior of another device or system.
>
> Compensator
>
> A Compensator is a control system that regulates another system, usually by conditioning the input or the output to that system. Compensators are typically employed to correct a single design flaw, with the intention of affecting other aspects of the design in a minimal manner.

Classical and Modern

Classical and **Modern** control methodologies are named in a misleading way, because the group of techniques called "Classical" were actually developed later then the techniques labled "Modern". However, in terms of developing control systems, Modern methods have been used to great effect more recently, while the Classical methods have been gradually falling out of favor. Most recently, it has been shown that Classical and Modern methods can be combined to highlight their respective strengths and weaknesses.

Classical Methods, which this book will consider first, are methods involving the **Laplace Transform domain**. Physical systems are modeled in the so-called "time domain", where the response of a given system is a function of the various inputs, the previous system values, and time. As time progresses, the state of the system, and it's response change. However, time-domain models for systems are frequently modeled using high-order differential equations, which can become impossibly difficult for humans to solve, and some of which can even become impossible for modern computer systems to solve efficiently. To counteract this problem, integral transforms, such as the **Laplace Transform**, and the **Fourier Transform** can be employed to change an Ordinary Differential Equation (ODE) in the time domain into a regular algebraic polynomial in the transform domain. Once a given system has been converted into the transform domain, it can be manipulated with greater ease, and analyzed quickly and simply, by humans and computers alike.

Modern Control Methods, instead of changing domains to avoid the complexities of time-domain ODE mathematics, converts the differential equations into a system of lower-order time domain equations called **State Equations**, which can then be manipulated using techniques from linear algebra (matrices). This book will consider Modern Methods second.

A third distinction that is frequently made in the realm of control systems is to divide analog methods (classical and modern, described above) from digital methods. Digital Control Methods were designed to try and incorporate the emerging power of computer systems into previous control methodologies. A special transform, known as the **Z-Transform**, was developed that can adequately describe digital systems, but at the same time can be converted (with some effort) into the Laplace domain. Once in the Laplace domain, the digital system can be manipulated and analyzed in a very similar manner to Classical analog systems. For this reason, this book will not make a hard and fast distinction between Analog and Digital systems, and instead will attempt to study both paradigms in parallel.

History

The field of control systems started essentially in the ancient world. Early civilizations, notably the greeks and the arabs were heaviliy preoccupied with the accurate measurement of time, the result of which were several "water clocks" that were designed and implemented.

Pierre-Simon Laplace

1749-1827

Joseph Fourier

1768-1840

However, there was very little in the way of actual progress made in the field of engineering until the beginning of the renassiance in Europe. Leonhard Euler (for whom **Euler's Formula** is named) discovered a powerful integral transform, but Pierre Simon-Laplace used the transform (later called the **Laplace Transform**) to solve complex problems in probability theory.

Joseph Fourier was a court mathematician in France under Napoleon I. He created a special function decomposition called the **Fourier Series**, that was later generalized into an integral transform, and named in his honor (the **Fourier Transform**).

Oliver Heaviside

The "golden age" of control engineering occured between 1910-1945, where mass communication methods were being created and two world wars were being fought. During this period, some of the most famous names in controls engineering were doing their work: Nyquist and Bode.

Hendrik Wade Bode and **Harry Nyquist**, especially in the 1930's while working with Bell Laboratories, created the bulk of what we now call "Classical Control Methods". These methods were based off the results of the Laplace and Fourier Transforms, which had been previously known, but were made popular by **Oliver Heaviside** around the turn of the century. Previous to Heaviside, the transforms were not widely used, nor respected mathematical tools.

Bode is credited with the "discovery" of the closed-loop feedback system, and the logarithmic plotting technique that still bears his name (**bode plots**). Harry Nyquist did extensive research in the field of system stability and information theory. He created a powerful stability criteria that has been named for him (**The Nyquist Criteria**).

Modern control methods were introduced in the early 1950's, as a way to bypass some of the shortcomings of the classical methods. Modern control methods became increasingly popular after 1957 with the invention of the computer, and the start of the space program. Computers created the need for digital control methodologies, and the space program required the creation of some "advanced" control techniques, such as "optimal control", "robust control", and "nonlinear control". These last subjects, and several more, are still active areas of study among research engineers.

Branches of Control Engineering

Here we are going to give a brief listing of the various different methodologies within the sphere of control engineering. Oftentimes, the lines between these methodologies are blurred, or even erased completely.

Classical Controls
: Control methodologies where the ODEs that describe a system are transformed using the Laplace, Fourier, or Z Transforms, and manipulated in the transform domain.

Modern Controls
: Methods where high-order differential equations are broken into a system of first-order equations. The input, output, and internal states of the system are described by vectors called "state variables".

Robust Control
: Control methodologies where arbitrary outside noise/disturbances are accounted for, as well as internal inaccuracies caused by the heat of the system itself, and the environment.

Optimal Control
: In a system, performance metrics are identified, and arranged into a "cost function". The cost function is minimized to create an operational system with the lowest cost.

Adaptive Control
: In adaptive control, the control changes it's response characteristics over time to better control the system.

Nonlinear Control
: The youngest branch of control engineering, nonlinear control encompasses systems that cannot be described by linear equations or ODEs, and for which there is often very little supporting theory available.

Game Theory
: Game Theory is a close relative of control theory, and especially robust control and optimal control theories. In game theory, the external disturbances are not considered to be random noise processes, but instead are considered to be "opponents". Each player has a cost function that they attempt to minimize, and that their opponents attempt to maximize.

This book will definately cover the first two branches, and will hopefully be expanded to cover some of the later branches, if time allows.

MATLAB

MATLAB is a programming tool that is commonly used in the field of control engineering. We will not consider MATLAB in the main narrative of this book, but we will provide an appendix that will show how MATLAB is used to solve control problems, and design and model control systems. This appendix can be found at: Control Systems/MATLAB.

For more information on MATLAB in general, see: MATLAB Programming

Nearly all textbooks on the subject of control systems, linear systems, and system analysis will use MATLAB as an integral part of the text. Students who are learning this subject at an accredited university will certainly have

seen this material in their textbooks, and are likely to have had MATLAB work as part of their classes. It is from this perspective that the MATLAB appendix is written.

There are a number of other software tools that are useful in the analysis and design of control systems. Additional information can be added in the appendix of this book, depending on the experiance and prior knowledge of contributors.

System Identification

Systems

We will begin our study by talking about **systems**. Systems, in the barest sense, are devices that take input, and produce an output. The output is related to the input by a certain relation known as the **system response**. The system response usually can be modeled with a mathematical relationship between the system input and the system output.

There are many different types of systems, and the process of classifying systems in these ways is called **system identification**.

System Identification

Physical Systems can be divided up into a number of different catagories, depending on particular properties that the system exhibits. Some of these system classifications are very easy to work with, and have a large theory base for studying. Some system classifications are very complex, and have still not been investigated with any degree of success. This book will focus primarily on **linear time-invariant** (LTI) systems. LTI systems are the easiest class of system to work with, and have a number of properties that make them ideal to study. In this chapter, we will discuss some properties of systems, and we will define exactly what an LTI system is.

Additivity

A system satisfies the property of **additivity**, if a sum of inputs results in a sum of outputs. By definition: an input of $x_3(t) = x_1(t) + x_2(t)$ results in an output of $y_3(t) = y_1(t) + y_2(t)$. To determine whether a system is additive, we can use the following test:

Given a system f that takes an input x and outputs a value y, we use two inputs (x_1 and x_2) to produce two outputs:

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

Now, we create a composite input that is the sum of our previous inputs:

$$x_3 = x_1 + x_2$$

Then the system is additive if the following equation is true:

$$y_3 = f(x_3) = f(x_1 + x_2) = f(x_1) + f(x_2) = y_1 + y_2$$

Example: Sinusoids

Given the following equation:

$$y(t) = \sin(3x(t))$$

We can create a sum of inputs as:

$$x(t) = x_1(t) + x_2(t)$$

and we can construct our expected sum of outputs:

$$y(t) = y_1(t) + y_2(t)$$

Now, plugging these values into our equation, we can test for equality:

$$y_1(t) + y_2(t) = \sin(3[x_1(t) + x_2(t)])$$

And we can see from this that our equality is not satisfied, and the equation is not additive.

Homogeniety

A system satisfies the condition of **homogeniety** if an input scaled by a certain factor produces an output scaled by that same factor. By definition: an input of ax_1 results in an output of ay_1. In other words, to see if function f() is homogenous, we can perform the following test:

We stimulate the system f with an arbitrary input x to produce an output y:

$$y = f(x)$$

Now, we create a second input x_1, scale it by a multiplicative factor C (C is an arbitrary constant value), and produce a corresponding output y_1

$$y_1 = f(Cx_1)$$

Now, we assign x to be equal to x_1:

$$x_1 = x$$

Then, for the system to be homogenous, the following equation must be true:

$$y_1 = f(Cx) = Cf(x) = Cy$$

Example: Straight-Line

Given the equation for a straight line:

$$y = f(x) = 2x + 3$$

$$y_1 = f(Cx_1) = 2(Cx_1) + 3 = C2X_1 + 3$$

$$x_1 = x$$

And comparing the two results, we see they are not equal:

$$y_1 = C2x + 3 \neq Cy = C(2x + 3) = C2x + C3$$

Therefore, the equation is *not homogenous*.

Linearity

A system is considered **linear** if it satisfies the conditions of Additivity and Homogeniety. In short, a system is linear if the following is true:

We take two arbitrary inputs, and produce two arbitrary outputs:

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

Now, a linear combination of the inputs should produce a linear combination of the outputs:

$$f(Ax + By) = f(Ax) + f(By) = Af(x) + Bf(y)$$

This condition of additivity and homogeniety is called **superposition**. A system is linear if it satisfies the condition of superposition.

Example: Linear Differential Equations

Is the following equation linear:

$$\frac{dy(t)}{dt} + y(t) = x(t)$$

To determine whether this system is linear, we construct a new composite input:

$$x(t) = Ax_1(t) + Bx_2(t)$$

And we create the expected composite output:

$$y(t) = Ay_1(t) + By_2(t)$$

And plug the two into our original equation:

$$\frac{d[Ay_1(t) + By_2(t)]}{dt} + [Ay_1(t) + By_2(t)] = Ax_1(t) + Bx_2(t)$$

We can factor out the derivative operator, as such:

$$\frac{d}{dt}[Ay_1(t) + By_2(t)] + [Ay_1(t) + By_2(t)] = Ax_1(t) + Bx_2(t)$$

And we can convert the various composite terms into the respective variables, to prove that this system is linear:

$$\frac{dy(t)}{dt} + y(t) = x(t)$$

For the record, derivatives and integrals are linear operators, and ordinary differentialy equations typically are linear equations.

Causality

Causality is a property that is very similar to memory. A system is called **causal** if it is only dependant on past or current inputs. A system is called **non-causal** if the output of the system is dependant on future inputs. This book will only consider causal systems, because they are easier to work with and understand, and since most practical systems are causal in nature.

Memory

A system is said to have memory if the output from the system is dependant on past inputs (or future inputs!) to the system. A system is called **memoryless** if the output is only dependant on the current input. Memoryless systems are easier to work with, but systems with memory are more common in digital signal processing applications.

Systems that have memory are called **dynamic** systems, and systems that do not have memory are **instantaneous** systems.

Time-Invariance

A system is called **time-invariant** if the system relationship between the input and output signals is not dependant on the passage of time. If the input signal $x(t)$ produces an output $y(t)$ then any time shifted input, $x(t+\delta)$, results in a time-shifted output $y(t+\delta)$ This property can be satisfied if the transfer function of the system is not a function of time except expressed by the input and output. If a system is time-invariant then the system block is commutative with an arbitrary delay. We will discuss this facet of time-invariant systems later.

To determine if a system f is time-invariant, we can perform the following test:

We apply an arbitrary input x to a system and produce an arbitrary output y:

$$y(t) = f(x(t))$$

And we apply a second input x_1 to the system, and produce a second output:

$$y_1(t) = f(x_1(t))$$

Now, we assign x_1 to be equal to our first input x, time-shifted by a given constant value δ:

$$x_1(t) = x(t - \delta)$$

Finally, a system is time-invariant if y_1 is equal to y shifted by the same value δ:

$$y_1(t) = y(t - \delta)$$

LTI Systems

A system is considered to be a **Linear Time-Invariant** (LTI) system if it satisfies the requirements of time-invariance and linearity. LTI systems are one of the most important types of systems, and we will consider them almost exclusively in this book.

Lumpedness

A system is said to be **lumped** if one of the two following conditions are satisfied:

1. There are a finite number of states
2. There are a finite number of state variables.

Systems which are not lumped are called **distributed**. We will not discuss distributed systems much in this book, because the topic is very complex.

Relaxed

A system is said to be **relaxed** if the system is causal, and at the initial time t_0 the output of the system is zero.

$$y(t_0) = f(x(t_0)) = 0$$

Stability

Stability is a very important concept in systems, but it is also one of the hardest function properties to prove. There are several different criteria for system stability, but the most common requirement is that the system must produce a finite output when subjected to a finite input. For instance, if we apply 5 volts to the input terminals of a given circuit, we would like it if the circuit output didn't approach infinity, and the circuit itself didn't melt or explode. This type of stability is often known as "**Bounded Input, Bounded Output**" stability, or **BIBO**.

Control Systems engineers will frequently say that an unstable system has "exploded". Some physical systems actually can rupture or explode when they go unstable.

Digital and Analog

Digital and Analog

There is a significant distinction between an **analog system** and a **digital system**, in the same way that there is a significant difference between analog and digital data. This book is going to consider both analog and digital topics, so it is worth taking some time to discuss the differences, and to display the different notations that will be used with each.

Continuous Time

A signal is called **continuous-time** if it is defined at every time t.

A system is a continuous-time system if it takes a continuous-time input signal, and outputs a continuous-time output signal.

Discrete Time

A signal is called **discrete-time** if it is only defined for particular points in time. A digital system takes discrete-time input signals, and produces discrete-time output signals.

Quantized

A signal is called **Quantized** if it can only be certain values, and cannot be other values.

Analog

By definition:

> Analog
>
> A signal is considered analog if it is defined for all points in time, and if it can take any real magnitude value within it's range.

An analog system is a system that represents data using a direct conversion from one form to another.

Example: Motor

> If we have a given motor, we can show that the output of the motor (rotation in units of radians per second, for instance) is a function of the amount of voltage and current that are input to the motor. We can show the relationship as such:
>
> $$\Theta(v) = f(v)$$
>
> Where Θ is the output in terms of Rad/sec, and f(v) is the motor's conversion function between the input

voltage (**v**) and the output. For any value of **v** we can calculate out specifically what the rotational speed of the motor should be.

Example: Analog Clock

Consider a standard analog clock, which represents the passage of time though the angular position of the clock hands. We can denote the angular position of the hands of the clock with the system of equations:

$$\begin{aligned}\phi_h &= f_h(t)\\ \phi_m &= f_m(t)\\ \phi_s &= f_s(t)\end{aligned}$$

Where φ_h is the angular position of the hour hand, φ_m is the angular position of the minute hand, and φ_s is the angular position of the second hand. The positions of all the different hands of the clock are dependant on functions of time.

Different positions on a clock face correspond directly to different times of the day.

Digital

Digital data is represented by discrete number values. By definition:

Digital
: A signal or system that is discrete-time and quantized.

Digital data always have a certain granularity, and therefore there will almost always be an error associated with using such data, especially if we want to account for all real numbers. The tradeoff, of course, to using a digital system is that our powerful computers with our powerful, Moore's law microprocessor units, can be instructed to operate on digital data only. This benefit more then makes up for the shortcomings of a digital representation system.

Discrete systems will be denoted inside square brackets, as is a common notation in texts that deal with discrete values. For instance, we can denote a discrete data set of ascending numbers, starting at 1, with the following notation:

x[n] = [1 2 3 4 5 6 ...]

n, or other letters from the central area of the alphabet (m, i, j, k, l, for instance) are commonly used to denote discrete time values. Analog, or "non-discrete" values are denoted in regular expression syntax, using parenthesis.

Example: Digital Clock

As a common example, let's consider a digital clock: The digital clock represents time with binary electrical data signals of 1 and 0. The 1's are usually represented by a positive voltage, and a 0 is

generally represented by zero voltage. Counting in binary, we can show that any given time can be represented by a base-2 numbering system:

Minute	Binary Representation
1	1
10	1010
30	11110
59	111011

But what happens if we want to display a fraction of a minute, or a fraction of a second? A typical digital clock has a certain amount of **precision**, and it cannot express fractional values smaller then that precision.

Hybrid Systems

Hybrid Systems are systems that have both analog and digital components. Devices called **samplers** are used to convert analog signals into digital signals, and Devices called **reconstructors** are used to convert digital signals into analog signals. Because of the use of samplers, hybrid systems are frequently called **sampled-data systems**.

Example: Car Computer

Most modern automobiles today have integrated computer systems, that monitor certain aspects of the car, and actually help to control the performance of the car. The speed of the car, and the rotational speed of the transmission are analog values, but a sampler converts them into digital values so the car computer can monitor them. The digital computer will then output control signals to other parts of the car, to alter analog systems such as the engine timing, the suspension, the brakes, and other parts. Because the car has both digital and analog components, it is a **hybrid system**.

Continuous and Discrete

A system is considered **continuous-time** if the signal exists for all time. Frequently, the terms "analog" and "continuous" will be used interchangably, although they are not strictly the same.

Note:
We are not using the word "continuous" here in the sense of *continuously differentiable*, as is common in math texts.

Discrete systems can come in three flavors:

1. Discrete time
2. Discrete magnitude (quantized)
3. Discrete time and magnitude (digital)

Discrete magnitude systems are systems where the signal value can only have certain values. **Discrete time** systems are systems where signals are only available (or valid) at particular times. Computer systems are discrete in the sense of (3), in that data is only read at specific discrete time intervals, and the data can have only a limited

number of discrete values.

A discrete-time system has as **sampling time** value associated with it, such that each discrete value occurs at multiples of the given sampling time. We will denote the sampling time of a system as T. We can equate the square-brackets notation of a system with the continuous definition of the system as follows:

$$x[n] = x(nT)$$

Notice that the two notations show the same thing, but the first one is typically easier to write, *and* it shows that the system in question is a discrete system. This book will use the square brackets to denote discrete systems by the sample number n, and parenthesis to denote continuous time functions.

System Metrics

System Metrics

When a system is being designed and analyzed, it doesn't make any sense to test the system with all manner of strange input functions, or to measure all sorts of arbitrary performance metrics. Instead, it is in everybody's best interest to test the system with a set of standard, simple, reference functions. Once the system is tested with the reference functions, there are a number of different metrics that we can use to determine the system performance.

It is worth noting that the metrics presented in this chapter represent only a small number of possible metrics that can be used to evaluate a given system. This wikibook will present other useful metrics along the way, as their need becomes apparent.

Standard Inputs

There are a number of standard inputs that are considered simple enough and universal enough that they are considered when designing a system. These inputs are known as a **unit step**, a **ramp**, and a **parabolic** input.

> **Note**:
> All of the standard inputs are zero before time zero

Unit Step

A unit step function is defined piecewise as such:

$$u(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases}$$ **[Unit Step Function]**

The unit step function is a highly important function, not only in control systems engineering, but also in signal processing, systems analysis, and all branches of engineering. If the unit step function is input to a system, the output of the system is known as the **step response**. The step response of a system is an important tool, and we will study step responses in detail in later chapters.

Ramp

A unit ramp is defined in terms of the unit step function, as such:

$$r(t) = tu(t)$$ **[Unit Ramp Function]**

It is important to note that the ramp function is simply the integral of the unit step function:

$$r(t) = \int u(t)dt = tu(t)$$

This definition will come in handy when we learn about the **Laplace Transform**.

Parabolic

A unit parabolic input is similar to a ramp input:

$$p(t) = \frac{1}{2}t^2u(t)$$ **[Unit Parabolic Function]**

Notice also that the unit parabolic input is equal to the integral of the ramp function:

$$p(t) = \int r(t)dt = \int tu(t)dt = \frac{1}{2}t^2u(t) = \frac{1}{2}tr(t)$$

Again, this result will become important when we learn about the Laplace Transform.

Also, sinusoidal and exponential functions are considered basic, but they are too difficult to use in initial analysis of a system.

Steady State

When a unit-step function is input to a system, the **steady state** value of that system is the output value at time $t = \infty$. Since it is impractical (if not completely impossible) to wait till infinity to observe the system, approximations and mathematical calculations are used to determine the steady-state value of the system.

Target Value

The target output value is the value that our system attempts to obtain for a given output. This is not the same as the steady-state value, which is the actual value that the target does obtain. The target value is frequently referred to as the **reference value**, or the "reference function" of the system. In essence, this is the value that we want the system to produce. When we input a "5" into an elevator, we want the output (the final position of the elevator) to be the fifth floor. Pressing the "5" button is the reference input, and is the expected value that we want to obtain. If we press the "5" button, and the elevator goes to the third floor, then our elevator is poorly designed.

Rise Time

Rise time is the amount of time that it takes for the system response to reach the target value from an initial state of zero. Many texts on the subject define the rise time as being 80% of the total time it takes to rise between the initial position and the target value. This is because some systems never rise to 100% of the expected, target value, and therefore they would have an infinite rise-time. This book will specify which convention to use for each individual problem.

Note that rise time is not the amount of time it takes to acheive steady-state, only the amount of time it takes to reach the desired target value for the first time.

Percent Overshoot

Underdamped systems frequently overshoot their target value initially. This initial surge is known as the "overshoot value". The ratio of the amount of overshoot to the target steady-state value of the system is known as the **percent overshoot**. Percent overshoot represents an overcompensation of the system, and can output dangerously large output signals that can damage a system.

Example: Refrigerator

Consider an ordinary household refrigerator. The refridgerator has cycles where it is on and when it is off. When the refrigerator is on, the coolant pump is running, and the temperature inside the refrigerator decreases. The temperature decreases to a much lower level then is required, and then the pump turns off.

When the pump is off, the temperature slowly increases again as heat is absorbed into the refrigerator. When the temperature gets high enough, the pump turns back on. Because the pump cools down the refrigerator more then it needs to initially, we can say that it "overshoots" the target value by a certain specified amount.

Another example concerning a refrigerator concerns the electrical demand of the heat pump when it first turns on. The pump is an inductive mechanical motor, and when the motor first activates, a special counter-acting force known as "back EMF" resists the motion of the motor, and causes the pump to draw more electricity until the motor reaches it's final speed. During the startup time for the pump, lights on the same electrical circuit as the refrigerator may dim slightly, as electricity is drawn away from the lamps, and into the pump. This initial draw of electricity is a good example of overshoot.

Steady-State Error

Sometimes a system might never achieve the desired steady state value, but instead will settle on an output value that is not desired. The difference between the steady-state output value to the reference input value at steady state is called the **steady state error** of the system. We will use the variable e_{ss} to denote the steady-state error of the system.

Settling Time

After the initial rise time of the system, some systems will oscillate and vibrate for an amount of time before the system output settles on the final value. The amount of time it takes to reach steady state after the initial rise time is known as the **settling time**. Notice that damped oscillating systems may never settle completely, so we will define settling time as being the amount of time for the system to reach, and stay in, a certain acceptable range.

System Order

The **order** of the system is defined by the highest exponent in the transfer function. In a **proper system**, the system order is defined as the degree of the denominator polynomial.

Proper Systems

A **proper system** is a system where the degree of the denominator is larger than or equal to the degree of the numerator polynomial. A **strictly proper system** is a system where the degree of the denominator polynomial is larger then (but never equal to) the degree of the numerator polynomial.

It is important to note that only proper systems can be physically realized. In other words, a system that is not

proper cannot be built. It makes no sense to spend alot of time designing and analyzing imaginary systems.

Example: System Order

Find the order of this system:

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

The highest exponent in the denominator is s^2, so the system is order 2. Also, since the denominator is a higher degree then the numerator, this system is proper.

in the above example, G(s) is a second-order transfer function because in the denominator one of the s variables has an exponent of 2. Second-order functions are the easiest to work with, and this book will focus on second-order LTI systems.

System Type

Let's say that we have a transfer function that is in the following generalized form (known as **pole-zero form**):

$$G(s) = \frac{K \prod_i (s - s_i)}{s^N \prod_j (s - s_j)}$$ **[Pole-Zero Form]**

we call the parameter N the **system type**. Note that increased system type number correspond to larger numbers of poles at s = 0. More poles at the origin generally have a beneficial effect on the system, but they increase the order of the system, and make it increasingly difficult to implement physically. Now, we will define a few terms that are commonly used when discussing system type. These new terms are **Position Error**, **Velocity Error**, and **Acceleration Error**. These names are throwbacks to physics terms where acceleration is the derivative of velocity, and velocity is the derivative of position. Note that none of these terms are meant to deal with movement, however.

Poles at the origin are called **integrators**, because they have the effect of performing integration on the input signal.

Position Error

The position error, denoted by the **position error constant** K_p. This is the amount of steady state error of the system when multiplied by a unit step input. We define the position error constant as follows:

$$K_p = \lim_{s \to 0} G(s)$$ **[Position Error Constant]**

Where G(s) is the transfer function of our system.

Velocity Error

The velocity error is the amount of steady state error when the system is stimulated with a ramp input. We define the **velocity error constant** as such:

$$K_v = \lim_{s \to 0} sG(s)$$ **[Velocity Error Constant]**

Acceleration Error

The acceleration error is the amount of steady-state error when the system is stimulated with a parabolic input. We define the **acceleration error constant** to be:

$$K_a = \lim_{s \to 0} s^2 G(s)$$ **[Acceleration Error Constant]**

Now, this table will show breifly the relationship between the system type, the kind of input (step, ramp, parabolic), and the steady state error of the system:

	Unit System Input		
Type	**Au(t)**	**Ar(t)**	**Ap(t)**
0	$e_{ss} = \frac{A}{1 + K_p}$	$e_{ss} = \infty$	$e_{ss} = \infty$
1	$e_{ss} = 0$	$e_{ss} = \frac{A}{K_v}$	$e_{ss} = \infty$
2	$e_{ss} = 0$	$e_{ss} = 0$	$e_{ss} = \frac{A}{K_a}$
>2	$e_{ss} = 0$	$e_{ss} = 0$	$e_{ss} = 0$

Z-Domain Type

Likewise, we can show that the system order can be found from the following generalized transfer function in the Z domain:

$$G(z) = \frac{K \prod_i (z - z_i)}{(z-1)^N \prod_j (z - z_j)}$$

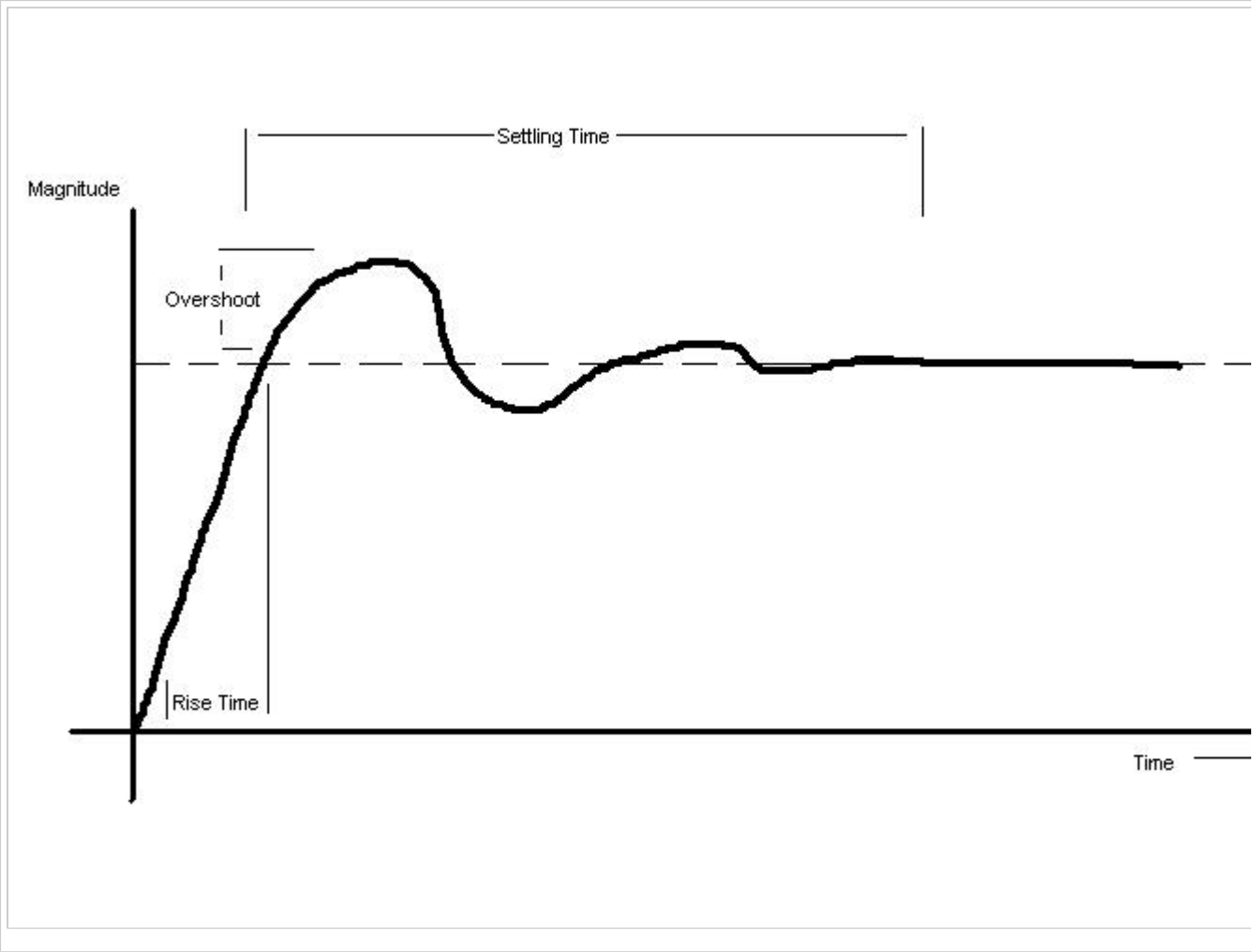
Where the constant N is the order of the digital system. Now, we will show how to find the various error constants in the Z-Domain:

Error Constant	**Equation**
Kp	$K_p = \lim_{z \to 1} G(z)$
Kv	$K_v = \lim_{z \to 1} (z-1) G(z)$
Ka	$K_a = \lim_{z \to 1} (z-1)^2 G(z)$

[Z-Domain Error Constants]

Visually

Here is an image of the various system metrics, acting on a system in response to a step input:



System Modeling

The Control Process

When designing a system, or implementing a controller to augment an existing system, we need to follow some basic steps:

1. Model the system mathematically
2. Analyze the mathematical model
3. Design system/controller
4. Implement system/controller and test

The vast majority of this book is going to be focused on (2), the analysis of the mathematical systems. This chapter alone will be devoted to a discussion of the mathematical modeling of the systems.

External Description

An **external description** of a system relates the system input to the system output without explicitly taking into account the internal workings of the system. The external description of a system is sometimes also referred to as the **Input-Output Description** of the system, because it only deals with the inputs and the outputs to the system.

If the system can be represented by a mathematical function h(t, r), where t is the time that the output is observed, and r is the time that the input is applied. We can relate the system function h(t, r) to the input (x) and the output (y) through the use of an integral:

$$y(t) = \int_{-\infty}^{\infty} g(t,r)x(r)dr$$ **[General System Description]**

This integral form holds for all linear systems, and every linear system can be described by such an equation.

If a system is causal, then there is no output of the system before time r, and we can change the limits of the integration:

$$y(t) = \int_{0}^{t} h(t,r)dr$$

Time-Invariant Systems

If a system is time-invariant (and causal), we can rewrite the system description equation as follows:

$$y(t) = \int_{0}^{t} h(t-r)x(r)dr$$

This equation is known as the **convolution integral**, and we will discuss it more in the next chapter.

Every Linear Time-Invariant (LTI) system can be used with the **Laplace Transform**, a powerful tool that allows

us to convert an equation from the time domain into the **S-Domain**, where many calculations are easier. Time-variant systems cannot be used with the Laplace Transform.

Internal Description

If a system is linear and lumped, it can also be described using a system of equations known as **state-space equations**. In state space equations, we use the variable x to represent the internal state of the system. We then use u as the system input, and we continue to use y as the system output. We can write the state space equations as such:

$$x'(t) = A(t)x(t) + B(t)u(t)$$
$$y(t) = C(t)x(t) + D(t)u(t)$$

We will discuss the state space equations more when we get to the section on **modern controls**

Complex Descriptions

Systems which are LTI and Lumped can also be described using a combination of the state-space equations, and the Laplace Transform. If we take the Laplace Transform of the state equations that we listed above, we can get a set of functions known as the **Transfer Matrix Functions**. We will discuss these functions in a later chapter.

Representations

To recap, we will prepare a table with the various system properties, and the available methods for describing the system:

Properties	State-Space Equations	Laplace Transform	Transfer Matrix
Linear, Time-Variant, Distributed	no	no	no
Linear, Time-Variant, Lumped	yes	no	no
Linear, Time-Invariant, Distributed	no	yes	no
Linear, Time-Invariant, Lumped	yes	yes	yes

We will discuss all these different types of system representation later in the book.

Analysis

Once a system is modeled using one of the representations listed above, the system needs to be analyszed. We can determine the system metrics, and then we can compare those metrics to our specification. If our system meets the specifications, you are finished (congratulations). If the system does not meet the specifications (as is typically the case), then suitable controllers and compensators need to be designed and added to the system.

Once the controllers and compensators have been designed, the job isn't finished: we need to analyze the new composite system to ensure that the controllers work properly. Also, we need to ensure that the systems are stable: unstable systems can be dangerous.