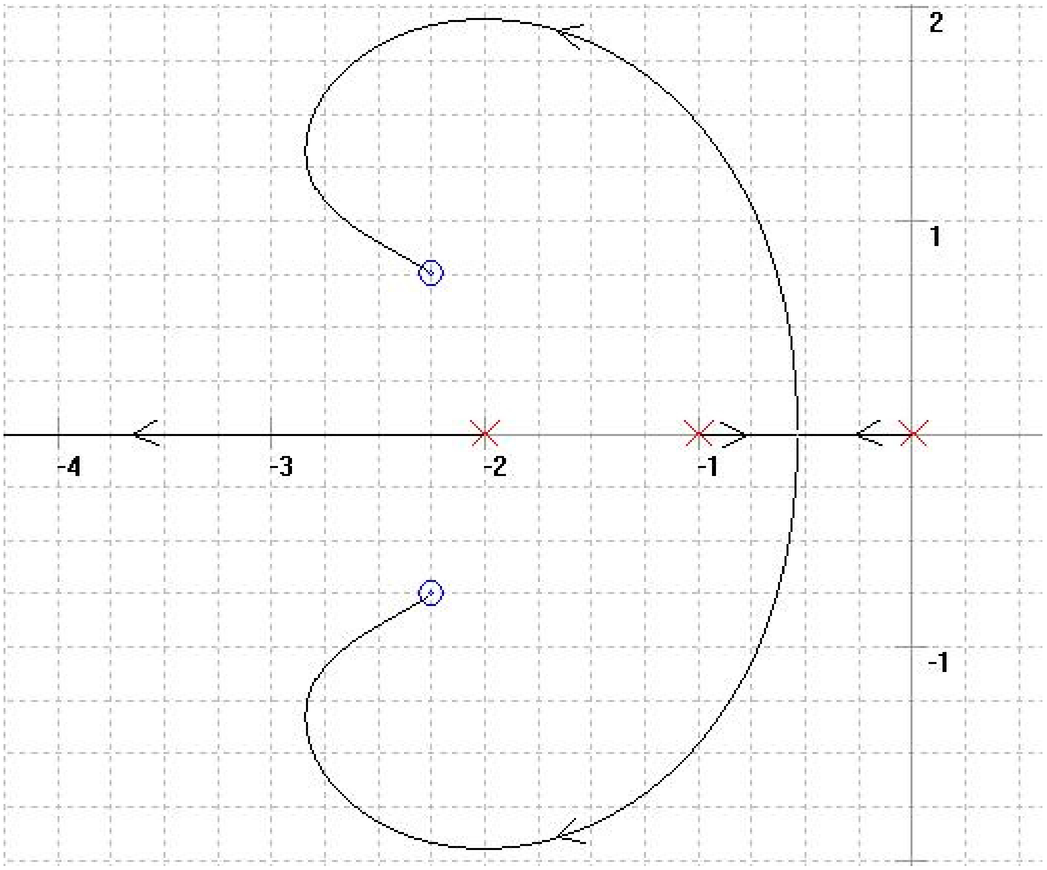


The Wikibook of automatic

Control Systems

And Control Systems Engineering
With
Classical and Modern Techniques
And
Advanced Concepts

Table of Contents

Current Status:

Preface

This book is a **wiki**, and is therefore open to be edited by anybody. Feel free to help out and contribute to this book in any way.

This book will discuss the topic of Control Systems, which is an interdisciplinary engineering topic. Methods considered here will consist of both "Classical" control methods, and "Modern" control methods. Also, discretely sampled systems (digital/computer systems) will be considered in parallel with the more common analog methods. This book will not focus on any single engineering discipline (electrical, mechanical, chemical, etc), although readers should have a solid foundation in the fundamentals of at least one discipline.

This book will require prior knowledge of linear algebra, integral and differential calculus, and at least some exposure to ordinary differential equations. In addition, a prior knowledge of integral transforms, specifically the Laplace and Z transforms will be very beneficial. Also, prior knowledge of the Fourier Transform will shed more light on certain subjects. Wikibooks with information on calculus topics or transformation topics required for this book will be listed below:

- Calculus
- Linear Algebra
- Signals and Systems
- Digital Signal Processing

Table of Contents

Special Pages

Print Version:	**Print version** ()
	Warning: Print version is over *200 pages long* as of 19 Oct, 2006.
Cover Page:	Cover Page
All Pages:	Page Listing
Search This Book:	Search this book (google) (http://www.google.com/custom?sa=Google+Search&domains=en.wikibooks.org/wiki/Control_Systems&sitesearch=en.wikibooks

Controls Introduction

Classical Control Methods

Modern Control Methods

System Representation

Stability

Controllers and Compensators

Optimal Control

Robust Control

Nonlinear Systems

Appendices

Resources, Glossary, and License

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.

Who is This Book For?

This book is intended to accompany a course of study in under-graduate and graduate engineering. As has been mentioned previously, this book is not focused on any particular discipline within engineering, however any person who wants to make use of this material should have some basic background in the Laplace transform (if not other transforms), calculus, etc. The material in this book may be used to accompany several semesters of study, depending on the program of your particular college or university. The study of control systems is generally a topic that is reserved for students in their 3rd or 4th year of a 4 year undergraduate program, because it requires so much previous information. Some of the more advanced topics may not be covered until later in a graduate program.

Many colleges and universities only offer one or two classes specifically about control systems at the undergraduate level. Some universities, however, do offer more then that, depending on how the material is broken up, and how much depth that is to be covered. Also, many institutions will offer a handful of graduate-level courses on the subject. This book will attempt to cover the topic of control systems from both a graduate and undergraduate level, with the advanced topics built on the basic topics in a way that is intuitive. As such, students should be able to begin reading this book in any place that seems an appropriate starting point, and should be able to finish reading where further information is no longer needed.

What are the Prerequisites?

Understanding of the material in this book will require a solid mathematical foundation. This book does not currently explain, nor will it ever try to fully explain most of the necessary mathematical tools used in this text. For that reason, the reader is expected to have read the following wikibooks, or have background knowledge comparable to them:

- Calculus
- Algebra
- Linear Algebra
- Differential Equations
- Engineering Analysis

The last book in the list, Engineering Analysis is especially recommended, because it analyzes a number of mathematical topics from the perspective of engineering. However the subject matter in that book relies on the 4 previous books.

Also, an understanding of the material presented in the following wikibooks will be helpful, but is not required:

- Signals and Systems

The Signals and Systems book will provide a basis in the field of **systems theory**, of which control systems is a subset.

How is this Book Organized?

This book will be organized following a particular progression. First this book will discuss the basics of system theory, and it will offer a brief refresher on integral transforms. Section 2 will contain a brief primer on digital information, for students who are not necessarily familiar with them. This is done so that digital and analog signals can be considered in parallel throughout the rest of the book. Next, this book will introduce the state-space method of system description and control. After section 3, topics in the book will use state-space and transform methods interchangably (and occasionally simultaneously). It is important, therefore, that these three chapters be well read and understood before venturing into the later parts of the book.

After the "basic" sections of the book, we will delve into specific methods of analyzing and designing control systems. First we will discuss Laplace-domain stability analysis techniques (Routh-Hurwitz, root-locus), and then frequency methods (Nyquist Criteria, Bode Plots). After the classical methods are discussed, this book will then discuss Modern methods of stability analysis. Finally, a number of advanced topics will be touched upon, depending on the knowledge level of the various contributers.

As the subject matter of this book expands, so too will the prerequisites. For instance, when this book is expanded to cover **nonlinear systems**, a basic background knowledge of nonlinear mathematics will be required.

Differential Equations Review

Implicit in the study of control systems is the underlying use of differential equations. Even if they aren't visible on the surface, all of the continuous-time systems that we will be looking at are described in the time domain by ordinary differential equations (ODE), some of which are relatively high-order.

Let's review some differential equation basics. Consider the topic of interest from a bank. The amount of interest accrued on a given principle balance (the amount of money you put into the bank) P, is given by:

$$\frac{dP}{dt} = rP$$

Where $\frac{dP}{dt}$ is the interest (rate of change of the principle), and r is the interest rate. Notice in this case that P is a function of time (t), and can be rewritten to reflect that:

$$\frac{dP(t)}{dt} = rP(t)$$

To solve this basic, first-order equation, we can use a technique called "separation of variables", where we move all instances of the letter P to one side, and all instances of t to the other:

$$\frac{dP(t)}{P(t)} = r\ dt$$

And integrating both sides gives us:

$$\log|P(t)| = rt + C$$

This is all fine and good, but generally, we like to get rid of the logarithm, by raising both sides to a power of e:

$$P(t) = e^{rt+C}$$

Where we can separate out the constant as such:

$$D = e^C$$
$$P(t) = De^{rt}$$

D is a constant that represents the **initial conditions** of the system, in this case the starting principle.

Differential equations are particularly difficult to manipulate, especially once we get to higher-orders of equations. Luckily, several methods of abstraction have been created that allow us to work with ODEs, but at the same time, not have to worry about the complexities of them. The classical method, as described above, uses the Laplace, Fourier, and Z Transforms to convert ODEs in the time domain into polynomials in a complex domain. These complex polynomials are significantly easier to solve then the ODE counterparts. The Modern method instead breaks differential equations into systems of low-order equations, and expresses this system in terms of matricies. It is a common precept in ODE theory that an ODE of order N can be broken down into N equations of order 1.

Readers who are unfamiliar with differential equations might be able to read and understand the material in this book reasonably well. However, all readers are encouraged to read the related sections in **Calculus**.

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.

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.

Pierre-Simon Laplace 1749-1827	Joseph Fourier 1768-1840

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.

Information about using MATLAB for control systems can be found in
the Appendix

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.

About Formatting

This book will use some simple conventions throughout:

Mathematical equations will be labled with the {{eqn}} template, to give them names. Equations that are labeled in such a manner are important, and should be taken special note of. For instance, notice the label to the right of this equation:

$$f(t) = \mathcal{L}^{-1}\left\{F(s)\right\} = \frac{1}{2\pi}\int_{c-i\infty}^{c+i\infty} e^{st}F(s)\,ds$$

[Inverse Laplace Transform]

Information which is tangent or auxiliary to the main text will be placed in these "sidebox" templates.

Examples will appear in TextBox templates, which show up as large grey boxes filled with text and equations.

Important Definitions

> Will appear in TextBox templates as well, except we will use this formatting to show that it is a definition.

Notes of interest will appear in "infobox" templates. These notes will often be used to explain some nuances of a mathematical derivation or proof.

Warnings will appear in these "warning" boxes. These boxes will point out common mistakes, or other items to be careful of.

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.

The study of control systems is highly dependant on the study of stability. Therefore, this book will spend a large amount of time discussing system stability.

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:

$$\phi_h = f_h(t)$$
$$\phi_m = f_m(t)$$
$$\phi_s = f_s(t)$$

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.

Sampling and Reconstruction

The process of converting analog information into digital data is called "Sampling". The process of converting digital data into an analog signal is called "Reconstruction". We will talk about both processes in a later chapter. For more information on the topic then is available in this book, see the Analog and Digital Conversion wikibook.

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:

$$u(t) = \begin{cases} 0, & t < 0 \\ 1, & t \ge 0 \end{cases}$$

[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]

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

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.

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:

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

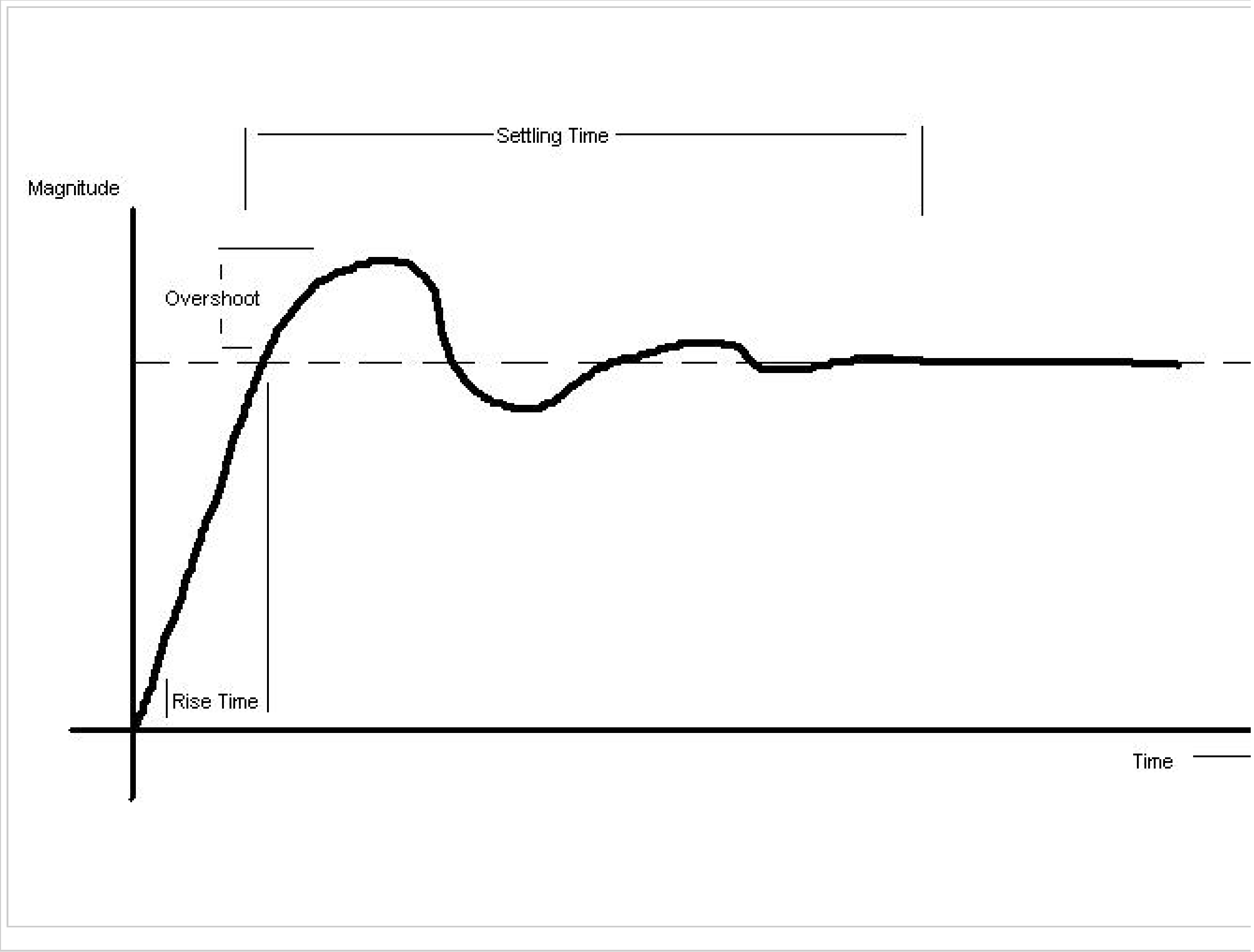
Where the constant N is the order of the digital system. Now, we will show how to find the various error constants in the Z-Domain:

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:

$$\begin{aligned} x'(t) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t) \end{aligned}$$

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.

Manufacture

Once the system has been properly designed, we can prototype our system and test it. Assuming our analysis was correct, and our design is good, the prototype should work as expected. Now we can move on to manufacture and distribute our completed systems.

Classical Controls

The classical method of controls involves analysis and manipulation of systems in the complex frequency domain. This domain, entered into by applying the Laplace or Fourier Transforms, is useful in examining the characteristics of the system, and determining the system response.

Transforms

Transforms

There are a number of transforms that we will be discussing throughout this book, and the reader is assumed to have at least a small prior knowledge of them. It is not the intention of this book to teach the topic of transforms to an audience that has had no previous exposure to them. However, we will include a brief refresher here to refamiliarize people who maybe cannot remember the topic perfectly. If you do not know what the **Laplace Transform** or the **Fourier Transform** are yet, it is highly recommended that you use this page as a simple guide, and look the information up on other sources. Specifically, Wikipedia has lots of information on these subjects.

Laplace Transform

The **Laplace Transform** converts an equation from the time-domain into the so-called "S-domain", or the **Laplace domain**, or even the "Complex domain". These are all different names for the same mathematical space, and they all may be used interchangably in this book, and in other texts on the subject. The Transform can only be applied under the following conditions:

1. The system or signal in question is analog.
2. The system or signal in question is Linear.
3. The system or signal in question is Time-Invariant.

The transform is defined as such:

$$F(s) = \mathcal{L}[f(t)] = \int_0^\infty f(t)e^{-st}dt$$

[Laplace Transform]

Laplace transform results have been tabulated extensively. More information on the Laplace transform, including a transform table can be found in **the Appendix**.

If we have a linear differential equation in the time domain:

$$y(t) = ax(t) + bx'(t) + cx''(t)$$

With zero initial conditions, we can take the Laplace transform of the equation as such:

$$Y(s) = aX(s) + bsX(s) + cs^2X(s)$$

And separating, we get:

$$Y(s) = X(s)[a + bs + cs^2]$$

Inverse Laplace Transform

The **inverse Laplace Transform** is defined as such:

$$f(t) = \mathcal{L}^{-1}\left\{F(s)\right\} = \frac{1}{2\pi}\int_{c-i\infty}^{c+i\infty} e^{st}F(s)\,ds$$ **[Inverse Laplace Transform]**

The inverse transfrom converts a function from the Laplace domain back into the time domain.

Matrices and Vectors

The Laplace Transform can be used on systems of linear equations in an intuitive way. Let's say that we have a system of linear equations:

$$y_1(t) = a_1x_1(t)$$
$$y_2(t) = a_2x_2(t)$$

We can arrange these equations into matrix form, as shown:

$$\begin{bmatrix}y_1(t)\\ y_2(t)\end{bmatrix} = \begin{bmatrix}a_1 & 0\\ 0 & a_2\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}$$

And write this symbolically as:

$$\mathbf{y}(t) = A\mathbf{x}(t)$$

We can take the Laplace transform of both sides:

$$\mathcal{L}[\mathbf{y}(t)] = \mathbf{Y}(s) = \mathcal{L}[A\mathbf{x}(t)] = A\mathcal{L}[\mathbf{x}(t)] = A\mathbf{X}(s)$$

Which is the same as taking the transform of each individual equation in the system of equations.

Example: RL Circuit

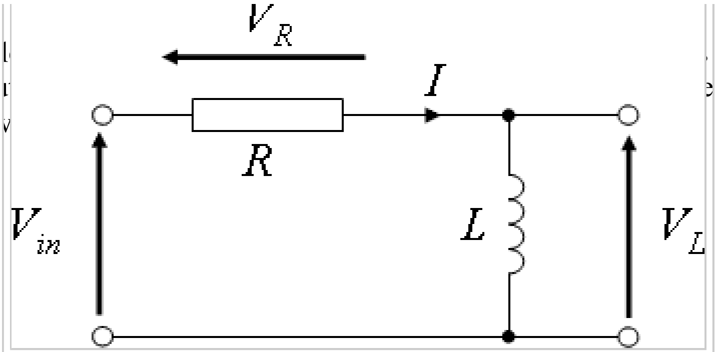
Here, we are going to show a common example of a first-order system, an **RL Circuit**. In an inductor, the relationship between the current (i), and the voltage (v) in the time domain is expressed as a derivative:

For more information about electric circuits, see:
Circuit Theory

$$v(t) = L\frac{di(t)}{dt}$$

Where L is a special quantity called the "Inductance" that is a property of inductors.

Let's say that we have a 1st order RL seri
inductance L, and the voltage source has i
over the inductor, V_{out}. In the time domai
describe the circuit:



Circuit diagram for the RL circuit example problem. V_L is the voltage over the inductor, and is the quantity we are trying to find.

$$V_{out}(t) = L\frac{di(t)}{dt}$$

$$V_{in}(t) = Ri(t) + L\frac{di(t)}{dt}$$

However, since the circuit is essentially a g as a voltage divider, we can put the output in terms of the input as follows:

$$V_{out}(t) = \frac{L\frac{di(t)}{dt}}{Ri(t) + L\frac{di(t)}{dt}}V_{in}(t)$$

This is a very complicated equation, and will be difficult to solve unless we employ the Laplace transform:

$$V_{out}(s) = \frac{Ls}{R + Ls}V_{in}(s)$$

We can divide top and bottom by L, and move V_{in} to the other side:

$$\frac{V_{out}}{V_{in}} = \frac{s}{\frac{R}{L} + s}$$

And using a simple table look-up, we can solve this for the time-domain relationship between the circuit input and the circuit output:

$$\frac{v_{out}}{v_{in}} = \frac{d}{dt}e^{\left(\frac{Rt}{L}\right)}u(t)$$

Partial Fraction Expansion

For more information about Partial Fraction Expansion, see: **Calculus**

Laplace transform pairs are extensively tabulated, but frequently we have transfer functions and other equations that do not have a tabulated inverse transform. If our equation is a fraction, we can often utilize **Partial Fraction Expansion** (PFE) to create a set of simpler terms that will have readily available inverse transforms. This section is going to give a brief reminder about PFE, for those who have already learned the topic. This refresher will be in the form of several examples of the process, as it relates to the Laplace Transform. People who are unfamiliar with PFE are encouraged to read more about it in **Calculus**.

First Example

If we have a given equation in the s-domain:

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

We can expand it into several smaller fractions as such:

$$F(s) = \frac{2s+1}{(s+1)(s+2)} = \frac{A}{(s+1)} + \frac{B}{(s+2)} = \frac{A(s+2)+B(s+1)}{(s+1)(s+2)}$$

This looks impossible, because we have a single equation with 3 unknowns (s, A, B), but in reality s can take any arbitrary value, and we can "plug in" values for s to solve for A and B, without needing other equations. For instance, in the above equation, we can multiply through by the denominator, and cancel terms:

$$(2s+1) = A(s+2) + B(s+1)$$

Now, when we set s → -2, the A term disappears, and we are left with B → 3. When we set s → -1, we can solve for A → -1. Putting these values back into our original equation, we have:

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

Remember, since the Laplace transform is a linear operator, the following relationship holds true:

$$\mathcal{L}[F(s)] = \mathcal{L}\left[\frac{-1}{(s+1)} + \frac{3}{(s+2)}\right] = \mathcal{L}\left[\frac{-1}{s+1}\right] + \mathcal{L}\left[\frac{3}{(s+2)}\right]$$

Finding the inverse transform of these smaller terms should be an easier process then finding the inverse transform of the whole function. Partial fraction expansion is a useful, and oftentimes necessary tool for finding the inverse of an s-domain equation.

Second example

If we have a given equation in the s-domain:

$$F(s) = \frac{79s^2 + 916s + 1000}{s(s+10)^3}$$

We can expand it into several smaller fractions as such:

$$F(s) = \frac{A}{s} + \frac{B}{(s+10)^3} + \frac{C}{(s+10)^2} + \frac{D}{s+10}$$

$$F(s) = \frac{A(s+10)^3 + Bs + Cs(s+10) + Ds(s+10)^2}{s(s+10)^3}$$

$$A(s+10)^3 + Bs + Cs(s+10) + Ds(s+10)^2 = 79^2 + 916s + 1000$$

Canceling terms wouldn't be enough here, we will open the brackets:

$$As^3+30As^2+300As+1000A+Bs+Cs^2+10Cs+Ds^3+20Ds^2+100Ds = 7$$

Let's compare coefficients:

$$A + D = 0$$

$$30A + C + 20D = 79$$

$$300A + B + 10C + 100D = 916$$

$$1000A = 1000$$

$$A = 1$$

$$B = 26$$

$$C = 69$$

$$D = -1$$

$$\frac{t^n}{n!}e^{-\alpha t} \cdot u(t) \rightarrow \frac{1}{(s+\alpha)^{n+1}}$$

According to the Laplace Transform table:

$$F(s) = \frac{A}{s} + \frac{B}{(s+10)^3} + \frac{C}{(s+10)^2} + \frac{D}{s+10}$$

$$F(s) = A\frac{1}{s} + B\frac{1}{(s+10)^3} + C\frac{1}{(s+10)^2} + D\frac{1}{s+10}$$

$$F(s) = 1\frac{1}{s} + 26\frac{1}{(s+10)^3} + 69\frac{1}{(s+10)^2} - 1\frac{1}{s+10}$$

$$f(t) = u(t) + 13t^2e^{-10t} + 69te^{-10t} - e^{-10t}$$

Third example (complex numbers):

$$F(s) = \frac{7s + 26}{s^2 - 80s + 1681} = \frac{As + B}{s^2 - 80s + 1681}$$

When the solution of the denominator is a complex number, we use a complex representation "As + B", like "3+i4"; in oppose to the use of a single letter (e.g. "D") - which is for real numbers:

$$As + B = 7s + 26$$

$$A = 7$$

$$B = 26$$

We will need to reform it into two fractions that look like this (without changing its value):

$$e^{-\alpha t}\sin(\omega t) \cdot u(t) \rightarrow \frac{\omega}{(s + \alpha)^2 + \omega^2}$$

$$e^{-\alpha t}\cos(\omega t) \cdot u(t) \rightarrow \frac{s + \alpha}{(s + \alpha)^2 + \omega^2}$$

Let's start with the denominator (for both fractions):

The roots of $s^2 - 80s + 1681$ are $40 \pm j9$

$$(s + a)^2 + \omega^2 = (s - 40)^2 + 9^2 \rightarrow \frac{As + B}{(s - 40)^2 + 9^2}$$

And now the numerators:

$$\frac{As + 40A - 40A + B}{(s - 40)^2 + 9^2}$$

$$\frac{As - 40A}{(s - 40)^2 + 9^2} + \frac{B + 40A}{(s - 40)^2 + 9^2}$$

$$A\frac{(s - 40)}{(s - 40)^2 + 9^2} + \frac{B + 40A}{9}\frac{9}{(s - 40)^2 + 9^2}$$

Inverse Laplace Transform:

$$f(t) = 7e^{40t}cos(9t) + 34e^{40t}sin(9t)$$

Fourth example:

$$F(s) = \frac{90s^2 - 1110}{s(s-3)(s^2-12s+37)} = \frac{A}{s} + \frac{B}{s-3} + \frac{Cs+D}{s^2-12s+37}$$

$$A(s-3)(s^2-12s+37) + Bs(s^2-12s+37) + (Cs+D)s(s-3) = 90s^2 -$$

$$As^3 - 15As^2 + 73As - 111A + Bs^3 - 12Bs^2 + 37Bs + Cs^3 - 3Cs^2 + Ds^2 - 3Ds$$

$$A + B + C = 0$$

$$-15A - 12B - 3C + D = 90$$

$$73A + 37B - 3D = 0$$

$$-111A = -1110$$

$$A = 10$$

$$B = -10$$

$$C = 0$$

$$D = 120$$

And now for the "fitting":

The roots of $s^2 - 12s + 37$ are $6 \pm j1$

$$A\frac{1}{s} + B\frac{1}{s-3} + C\frac{s}{(s-6)^2+1^2} + D\frac{1}{(s-6)^2+1^2}$$

No need to fit the fraction of D, because it is complete; no need to bother fitting the fraction of C, because C is equal to zero.

$$10\frac{1}{s} - 10\frac{1}{s-3} + 0\frac{s}{(s-6)^2+1^2} + 120\frac{1}{(s-6)^2+1^2}$$

$$f(t) = 10u(t) - 10e^{3t} + 120e^{6t}sin(t)$$

Final Value Theorem

The **Final Value Theorem** allows us to determine the value of the time domain equation, as the time approaches infinity, from the S domain equation. In Control Engineering, the Final Value Theorem is used most frequently to determine the steady-state value of a system.

$$x(\infty) = \lim_{s \to 0} sX(s)$$

[Final Value Theorem (Laplace)]

From our chapter on system metrics, you may recognize the value of the system at time infinity as the steady-state time of the system. The difference between the steady state value, and the expected output value we remember as being the steady-state error of the system. Using the Final Value Theorem, we can find the steady-state value, and the steady-state error of the system in the Complex S domain.

Example: Final Value Theorem

Find the final value of the following polynomial:

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

This is an admittedly simple example, because we can separate out the denominator into roots:

$$T(s) = \frac{1+s}{(1+s)(1+s)}$$

And we can cancel:

$$T(s) = \frac{1}{1+s}$$

Now, we can apply the **Final Value Theorem**:

$$\lim_{s \to \infty} s\frac{1}{1+s}$$

Using L'Hospital's rule (because this is an indeterminate form), we obtain the value:

$$\lim_{s \to \infty} s\frac{1}{1+s} = 1$$

Initial Value Theorem

Akin to the final value theorem, the **Initial Value Theorem** allows us to determine the initial value of the system (the value at time zero) from the S-Domain Equation. The initial value theorem is used most frequently to determine the starting conditions, or the "initial conditions" of a system.

$$x(0) = \lim_{s \to \infty} sX(s)$$

[Initial Value Theorem (Laplace)]

Common Transforms

We will now show you the transforms of the three functions we have already learned about: The unit step, the unit ramp, and the unit parabola. The transform of the unit step function is given by:

$$\mathcal{L}[u(t)] = \frac{1}{s}$$

And since the unit ramp is the integral of the unit step, we can multiply the above result times 1/s to get the transform of the unit ramp:

$$\mathcal{L}[r(t)] = \frac{1}{s^2}$$

Again, we can multiply by 1/s to get the transform of the unit parabola:

$$\mathcal{L}[p(t)] = \frac{1}{s^3}$$

Fourier Transform

The **Fourier Transform** is very similar to the Laplace transform. The fourier transform uses the assumption that any finite time-domain can be broken into an infinite sum of sinusoidal (sine and cosine waves) signals. Under this assumption, the Fourier Transform converts a time-domain signal into it's frequency-domain representation, as a function of the radial frequency, ω . The Fourier Transform is defined as such:

$$F(j\omega) = \mathcal{F}[f(t)] = \int_0^\infty f(t)e^{-j\omega t}dt$$

[Fourier Transform]

We can now show that the Fourier Transform is equivalent to the Laplace transform, when the following condition is true:

$$s = j\omega$$

Because the Laplace and Fourier Transforms are so closely related, it does not make much sense to use both transforms for all problems. This book, therefore, will concentrate on the Laplace transform for nearly all subjects, except those problems that deal directly with frequency values. For frequency problems, it makes life much easier to use the Fourier Transform representation.

Like the Laplace Transform, the Fourier Transform has been extensively tabulated. Properties of the Fourier transform, in addition to a table of common transforms is available in **the Appendix**.

Inverse Fourier Transform

The **inverse Fourier Transform** is defined as follows:

$$f(t) = \mathcal{F}^{-1}\left\{F(j\omega)\right\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{-j\omega t}d\omega$$ **[Inverse Fourier Transform]**

This transform is nearly identical to the Fourier Transform.

Complex Plane

Using the above equivalence, we can show that the Laplace transform is always equal to the Fourier Transform, if the variable s is an imaginary number. However, the Laplace transform is different if s is a real or a complex variable. As such, we generally define s to have both a real part and an imaginary part, as such:

$$s = \sigma + j\omega$$

And we can show that

$s = j\omega$, if $\sigma = 0$

Since the variable s can be broken down into 2 independant values, it is frequently of some value to graph the variable s on it's own special "S-plane". The S-plane graphs the variable σ on the horizontal axis, and the value of jω on the vertical axis.

Euler's Formula

There is an important result from calculus that is known as **Euler's Formula**, or "Euler's Relation". This important formula relates the important values of e, j, π, 1 and 0:

$$e^{j\pi} - 1 = 0$$

However, this result is derived from the following equation, setting ω to π:

$$e^{j\omega} = \cos(\omega) + j\sin(\omega)$$

This formula will be used extensively in some of the chapters of this book, so it is important to become familiar with it now.

[Euler's Formula]

Further Reading

- Digital Signal Processing/Continuous-Time Fourier Transform
- Signals and Systems/Aperiodic Signals
- Circuit Theory/Laplace Transform

Transfer Functions

Transfer Functions

A **Transfer Function** is the ratio of the output of a system to the input of a system, in the Laplace domain. If we have an input function of X(s), and an output function Y(s), we define the transfer function H(s) to be:

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

[Transfer Function]

Readers who have read the Circuit Theory book will recognize the tranfer function as being the Laplace transform of a circuit's impulse response.

Impulse Response

For comparison, we will consider the time-domain equivalent to the above input/output relationship. In the time domain, we generally denote the input to a system as x(t), and the output of the system as y(t). The relationship between the input and the output is denoted as the **impulse response**, h(t).

Note::
Time domain variables are generally written with lower-case letters. Laplace-Domain, and other transform domain variables are generally written using upper-case letters.

We define the impulse response as being the relationship between the system output to it's input. We can use the following equation to define the impulse response:

$$h(t) = \frac{y(t)}{x(t)}$$

Impulse Function

It would be handy at this point to define precisely what an "impulse" is. The **Impulse Function**, denoted with δ(t) is a special function defined peice-wise as follows:

$$\delta(t) = \begin{cases} 0, & t < 0 \\ \text{undefined}, & t = 0 \\ 0, & t > 0 \end{cases}$$

[Impulse Function]

An examination of the impulse function will show that it is related to the unit-step function as follows:

$$\delta(t) = \frac{du(t)}{dt}$$

and

$$u(t) = \int \delta(t) dt$$

The impulse function is not defined at point t = 0, but the impulse response must always satisfy the following condition, or else it is not a true impulse function:

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$

The response of a system to an impulse input is called the **impulse response**. Now, to get the Laplace Transform of the impulse function, we take the derivative of the unit step function, which means we multiply the transform of the unit step function by s:

$$\mathcal{L}[u(t)] = U(s) = \frac{1}{s}$$

$$\mathcal{L}[\delta(t)] = sU(s) = \frac{s}{s} = 1$$

This result can be verified in the transform tables in **The Appendix**.

Convolution

However, the impulse response cannot be used to find the system output from the system input in the same manner as the transfer function. If we have the system input and the impulse response of the system, we can calculate the system output using the *convolution operation* as such:

$$y(t) = h(t) * x(t)$$

Where " * " (asterisk) denotes the convolution operation. Convolution is a complicated combination of multiplication, integration and time-shifting. We can define the convolution between two functions, a(t) and b(t) as the following:

Remember: an asterisk means **convolution**, not **multiplication**!

$$(a * b)(t) = (b * a)(t) = \int_{-\infty}^{\infty} a(\tau) b(t - \tau) d\tau$$

[Convolution]

(The variable τ (greek tau) is a dummy variable for integration). This operation can be difficult to perform. Therefore, many people prefer to use the Laplace Transform (or another transform) to convert the convolution operation into a multiplication operation, through the **Convolution Theorem**.

Time-Invariant System Response

If the system in question is time-invariant, then the general description of the system can be replaced by a convolution integral of the system's impulse response and the system input. We can call this the **convolution description** of a system, and define it below:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

[Convolution Description]

Convolution Theorem

This method of solving for the output of a system is quite tedious, and in fact it can waste a large amount of time if you want to solve a system for a variety of input signals. Luckily, the Laplace transform has a special property, called the **Convolution Theorem**, that makes the operation of convolution easier:

Convolution Theorem

Convolution in the time domain becomes multiplication in the complex Laplace domain.
Multiplication in the time domain becomes convolution in the complex Laplace domain.

The Convolution Theorem can be expressed using the following equations:

$$\mathcal{L}[f(t) * g(t)] = F(s)G(s)$$
$$\mathcal{L}[f(t)g(t)] = F(s) * G(s)$$

[Convolution Theorem]

This also serves as a good example of the property of **Duality**.

Using the Transfer Function

The Transfer Function fully decribes a control system. The Order, Type and Frequency response can all be taken from this specific function. Nyquist and Bode plots can be drawn from the open loop Transfer Function. These plots show the stability of the system when the loop is closed. Using the denominator of the transfer function, called the characteristic equation the roots of the system can be derived.

For all these reasons and more, the Transfer function is an important aspect of classical control systems. Let's start out with the definition:

Transfer Function

The Transfer function of a system is the relationship of the system's output to it's input, represented in the complex Laplace domain.

If the complex Laplace variable is 's', then we generally denote the transfer function of a system as either G(s) or H(s). If the system input is X(s), and the system output is Y(s), then the transfer function can be defined as such:

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

If we know the input to a given system, and we have the transfer function of the system, we can solve for the system output by multiplying:

$$Y(s) = H(s)X(s)$$

[Transfer Function Description]

Example: Impulse Response

From a Laplace transform table, we know that the Laplace transform of the impulse function, δ(t) is:

$$\mathcal{L}[\delta(t)] = 1$$

So, when we plug this result into our relationship between the input, output, and transfer function, we get:

$$\begin{aligned}Y(s) &= X(s)H(s)\\ Y(s) &= (1)H(s)\\ Y(s) &= H(s)\end{aligned}$$

In other words, the "impulse response" is the output of the system when we input an impulse function.

Example: Step Response

From the Laplace Transform table, we can also see that the transform of the unit step function, u(t) is given by:

$$\mathcal{L}[u(t)] = \frac{1}{s}$$

Plugging that result into our relation for the transfer function gives us:

$$\begin{aligned}Y(s) &= X(s)H(s)\\ Y(s) &= \frac{1}{s}H(s)\\ Y(s) &= \frac{H(s)}{s}\end{aligned}$$

And we can see that the step response is simply the impulse response divided by s.

Frequency Response

The **Frequency Response** is similar to the Transfer function, except that it is the relationship between the system output and input in the complex Fourier Domain, not the Laplace domain. We can obtain the frequency response from the transfer function, by using the following change of variables:

$$s = j\omega$$

Frequency Response

The frequency response of a system is the relationship of the system's output to it's input, represented in the Fourier Domain.

Sampled Data Systems

Ideal Sampler

In this chapter, we are going to introduce the ideal sampler and the **Star Transform**. First, we need to introduce (or review) the **Geometric Series** infinite sum. The results of this sum will be very useful in calculating the Star Transform, later.

Consider a sampler device that operates as follows: every T seconds, the sampler reads the current value of the input signal at that exact moment. The sampler then holds that value on the output for T seconds, before taking the next sample. We have a generic input to this system, f(t), and our sampled output will be denoted f*(t). We can then show the following relationship between the two signals:

$$f^*(t) = f(0)(u(0) - u(T)) + f(T)(u(T) - u(2T)) + ...$$

Note that the value of f* at time t = 1.5T = T. This relationship works for any fractional value.

Taking the Laplace Transform of this infinite sequence will yield us with a special result called the **Star Transform**. The Star Transform is also occasionally called the "Starred Transform" in some texts.

Geometric Series

Before we talk about the Star Transform or even the Z-Transform, it is useful for us to review the mathematical background behind solving infinite series. Specifically, because of the nature of these transforms, we are going to look at methods to solve for the sum of a **geometric series**.

A geometic series is a sum of values with increasing exponents, as such:

$$\sum_{k=0}^{n} ar^k = ar^0 + ar^1 + ar^2 + ar^3 + \cdots + ar^n$$

In the equation above, notice that each term in the series has a coefficient value, a. We can optionally factor out this coefficient, if the resulting equation is easier to work with:

$$a\sum_{k=0}^{n} r^k = a\left(r^0 + r^1 + r^2 + r^3 + \cdots + r^n\right)$$

Once we have an infinite series in either of these formats, we can conveniently solve for the total sum of this series using the following equation:

$$a\sum_{k=0}^{n} r^k = a\frac{1 - r^{n+1}}{1 - r}$$

Let's say that we start our series off at a number that isn't zero. Let's say for instance that we start our series off at n=1 or n=100. Let's see:

$$\sum_{k=m}^{n} ar^k = ar^m + ar^{m+1}1 + ar^{m+2} + ar^{m+3} + \cdots + ar^n$$

We can generalize the sum to this series as follows:

$$\sum_{k=m}^{n} ar^k = \frac{a(r^m - r^{n+1})}{1-r}$$ **[Geometric Series]**

With that result out of the way, now we need to worry about making this series converge. In the above sum, we know that n is approaching infinity (because this is an *infinite sum*). Therefore, any term that contains the variable n is a matter of worry when we are trying to make this series converge. If we examine the above equation, we see that there is one term in the entire result with an n in it, and from that, we can set a fundamental inequality to govern the geometric series.

$$r^{n+1} < \infty$$

To satisfy this equation, we must satisfy the following condition:

$$r \le 1$$ **[Geometric convergence condition]**

Therefore, we come to the final result: **The geometric series converges if and only if the value of r is less than one.**

The Star Transform

The **Star Transform** is defined as such:

$$F^*(s) = \mathcal{L}^*[f(t)] = \sum_{i=0}^{\infty} f(iT)e^{-siT}$$ **[Star Transform]**

The Star Transform depends on the sampling time, T, and is different for a single signal, depending on the speed at which the signal is sampled. Since the Star Transform is defined as an infinite series, it is important to note that some inputs to the Star Transform will not converge, and therefore some functions do not have a valid Star Transform. Also, it is important to note that the Star Transform may only be valid under a particular region of convergance. We will cover this topic more when we discuss the Z-transform.

Star ↔ Laplace

The Laplace transform and the Star transform are clearly related, because we obtained the Star Transform by using the Laplace transform on a time-domain signal. However, the method to convert between the two results can be a slightly difficult one. To find the Star Transform of a Laplace function, we must take the residues of the Laplace equation, as such:

$$X^*(s) = \sum \left[residues\ of\ X(\lambda) \frac{1}{1 - e^{-T(s-\lambda)}} \right]_{at\ poles\ of\ E(\lambda)}$$

This math is advanced for most readers, so we can also use an alternate method, as follows:

$$X^*(s) = \frac{1}{T} \sum_{n=-\infty}^{\infty} X(s + jm\omega_s) + \frac{x(0)}{2}$$

Neither one of these methods are particularly easy, however, and therefore we will not discuss the relationship between the Laplace transform and the Star Transform any more then is absolutely necessary in this book. Suffice it to say, however, that the Laplace transform and the Star Transform *are related* mathematically.

Star + Laplace

In some systems, we may have components that are both continuous and discrete in nature. For instance, if our feedback loop consists of an Analog-To-Digital converter, followed by a computer (for processing), and then a Digital-To-Analog converter. In this case, the computer is acting on a digital signal, but the rest of the system is acting on continuous signals. Star transforms can interact with Laplace transforms in some of the following ways:

Given:

$$Y(s) = X^*(s)H(s)$$

Then:

$$Y^*(s) = X(s)^*H(s)^*$$

Given:

$$Y(s) = X(s)H(s)$$

Then:

$$Y^*(s) = \overline{XH}^*(s)$$
$$Y^*(s) \neq X^*(s)H^*(s)$$

Where $\overline{XH}^*(s)$ is the Star Transform of the product of X(s)H(s).

Convergence of the Star Transform

The Star Transform is defined as being an infinite series, so it is critically important that the series converge (not reach infinity), or else the result will be nonsensical. Since the Star Transform is a geometic series (for many input signals), we can use geometric series analysis to show whether the series converges, and even under what particular conditions the series converges.

The Z-Transform

Let us say now that we have a discrete data set that is sampled at regular intervals. We can call this set x[n]:

```
x[n] = [ x[0] x[1] x[2] x[3] x[4] ... ]
```

we can utilize a special transform, called the Z-transform, to make dealing with this set more easy:

This is also known as the **Bilateral Z-Transform**. We will only discuss this version of the transform in this book

$$X(z) = \mathcal{Z}\left\{x[n]\right\} = \sum_{i=-\infty}^{\infty} x[n]z^{-n}$$

[Z Transform]

Like the Star Transform, the Z Transform is defined as an infinite series, and therefore we need to worry about convergance. In fact, there are a number of instances that have identical Z-Transforms, but different regions of convergance (ROC). Therefore, when talking about the Z transform, you must include the ROC, or you are missing valuable information.

Z-Transform properties, and a table of common transforms can be found in: **the Appendix**.

Inverse Z Transform

The **inverse Z Transform** is defined by the following path integral:

$$x[n] = Z^{-1}\{X(z)\} = \frac{1}{2\pi j}\oint_C X(z)z^{n-1}dz$$

[Inverse Z Transform]

This integral is sufficiently complicated that we won't discuss it any further in this book. There are a number of Z-transform pairs available in table form in **The Appendix**.

Final Value Theorem

Like the Laplace Transform, the Z Transform also has an associated final value theorem:

$$x[\infty] = \lim_{z\to 1}(z-1)X(z)$$

[Final Value Theorem (Z)]

This equation can be used in the same way that the other equation can be used.

Star ↔ Z

The Z transform is related to the Star transform though the following change of variables:

$$z = e^{sT}$$

Notice that in the Z domain, we don't maintain any information on the sampling period, so converting to the Z domain from a Star Transformed signal loses that information. When converting back to the star domain however, the value for T can be re-insterted into the equation, if it is still available.

Also of some importance is the fact that the Z transform is bilinear, while the Star Transform is unilinear. This means that we can only convert between the two transforms if the sampled signal is zero for all values of n < 0.

Because the two transforms are so closely related, it can be said that the Z transform is simply a notational convenience for the Star Transform. With that said, this book could easily use the Star Transform for all problems, and ignore the added burden of Z transform notation entirely. A common example of this is Richard Hamming's book "Numerical Methods for Scientists and Engineers" which uses the Fourier Transform for all problems, considering the Laplace, Star, and Z-Transforms to be merely notational conveniences. However, the Control Systems wikibook is under the impression that the correct utilization of different transforms can make problems more easy to solve, and we will therefore use a multi-transform approach.

Z plane

Z is a complex variable with a real part and an imaginary part. In other words, we can define Z as such:

$$z = \mathrm{Re}(z) + j\mathrm{Im}(z)$$

Since Z can be broken down into two independant components, it often makes sense to graph the variable z on the **z-plane**. In the z-plane, the horizontal axis represents the real part of z, and the vertical axis represents the magnitude of the imaginary part of z.

Notice also that if we define z in terms of the star-transfrom relation:

$$z = e^{sT}$$

we can separate out s into real and imaginary parts:

$$s = \sigma + j\omega$$

We can plug this into our equation for z:

$$z = e^{(\sigma + j\omega)T} = e^{\sigma T} e^{j\omega T}$$

Through **Euler's formula**, we can separate out the complex exponential as such:

$$z = e^{\sigma T}(\cos(\omega T) + j\sin(\omega T))$$

If we define two new variables, M and φ:

$$M = e^{\sigma T}$$

$$\phi = \omega T$$

We can write z in terms of M and φ. Notice that it is euler's equation:

$$z = M\cos(\phi) + jM\sin(\phi)$$

Which is clearly a polar representation of z, with the magnitude of the polar function (M) based on the real-part of s, and the angle of the polar function (φ) is based on the imaginary part of s.

Region of Convergence

To best teach the region of convergance (ROC) for the Z-transform, we will do a quick example.

We have the following discrete series or a decaying exponential:

$$x[n] = e^{-2n}u[n]$$

Now, we can plug this function into the Z transform equation:

$$X(z) = \mathcal{Z}[x[n]] = \sum_{i=-\infty}^{\infty} e^{-2n}u[n]z^{-n}$$

Note that we can remove the unit step function, and change the limits of the sum:

$$X(z) = \sum_{i=0}^{\infty} e^{-2n}z^{-n}$$

This is because the series is 0 for all time less then n → 0. If we try to combine the n terms, we get the following result:

$$X(z) = \sum_{i=0}^{\infty} e^{2}(ez)^{-n}$$

Once we have our series in this term, we can break this down to look like our geometric series:

$$a = e^2$$

$$r = ez^{-1}$$

And finally, we can find our final value, using the geometric series formula:

$$a\sum_{k=0}^{n} r^k = a\frac{1-r^{n+1}}{1-r} = e^2\frac{1-((ez)^{-1})^{n+1}}{1-(ez)^{-1}}$$

Again, we know that to make this series converge, we need to make the r value less then 1:

$$|(ez)^{-1}| = \left|\frac{1}{ez}\right| \le 1$$

$$|ez| \ge 1$$

And finally we obtain the region of convergance for this Z-transform:

$$|z| \ge \frac{1}{e}$$

Note: z and s are complex variables, and therefore we need to take the magnitude in our ROC calculations. The "Absolute Value symbols" are actually the "magnitude calculation", and is defined as such:

$$x = A + jB$$
$$|x| = \sqrt{A^2 + B^2}$$

Inverse Z Transform

The **inverse Z-Transform** is defined as:

$$x[n] = Z^{-1}\{X(z)\} = \frac{1}{2\pi j}\oint_C X(z)z^{n-1}dz$$ **[Inverse Z Transform]**

Where C is a counterclockwise closed path encircling the origin and entirely in the region of convergence (ROC). The contour or path, C, must encircle all of the poles of $X(z)$.

This math is relatively advanced compared to some other material in this book, and therefore little or no further attention will be paid to solving the inverse Z-Transform in this manner. Z transform pairs are heavily tabulated in reference texts, so many readers can consider that to be the primary method of solving for inverse Z transforms.

Laplace ↔ Z

There are no easy, direct ways to convert between the Laplace transform and the Z transform directly. Nearly all methods of conversions reproduce some aspects of the original equation faithfully, and incorrectly reproduce other aspects. For some of the main mapping techniques between the two, see the Z Transform Mappings Appendix.

However, there are some topics that we need to discuss. First and foremost, conversions between the Laplace domain and the Z domain *are not linear*, this leads to some of the following problems:

1. $\mathcal{L}[G(z)H(z)] \ne G(s)H(s)$
2. $\mathcal{L}[G(s)H(s)] \ne G(z)H(z)$

This means that when we combine two functions in one domain multiplicatively, we must find a combined transform in the other domain. Here is how we denote this combined transform:

$$\mathcal{Z}[G(s)H(s)] = \overline{GH}(z)$$

Notice that we use a horizontal bar over top of the multiplied functions, to denote that we took the transform of the product, not of the individual peices. However, if we have a system that incorporates a sampler, we can show a simple result. If we have the following format:

$$Y(s) = X^*(s)H(s)$$

Then we can put everything in terms of the Star Transform:

$$Y^*(s) = X^*(s)H^*(s)$$

and once we are in the star domain, we can do a direct change of variables to reach the Z domain:

$$Y^*(s) = X^*(s)H^*(s) \rightarrow Y(z) = X(z)H(z)$$

Note that we can only make this equivalence relationship if the system incorporates an ideal sampler, and therefore one of the multiplicative terms is in the star domain.

Example

Let's say that we have the following equation in the Laplace domain:

$$Y(s) = A^*(s)B(s) + C(s)D(s)$$

And because we have a discrete sampler in the system, we want to analyze it in the Z domain. We can break up this equation into two separate terms, and transform each:

$$\mathcal{Z}[A^*(s)B(s)] \rightarrow \mathcal{Z}[A^*(s)B^*(s)] = A(z)B(z)$$

And

$$\mathcal{Z}[C(s)D(s)] = \overline{CD}(z)$$

And when we add them together, we get our result:

$$Y(z) = A(z)B(z) + \overline{CD}(z)$$

Reconstruction

Some of the easiest reconstruction circuits are called "Holding circuits". Once a signal has been transformed using the Star Transform (passed through an ideal sampler), the signal must be "reconstructed" using one of these hold systems (or an equivalent) before it can be analyzed in a Laplace-domain system.

If we have a sampled signal denoted by the Star Transform $X^*(s)$, we want to **reconstruct** that signal into a continuous-time waveform, so that we can manipulate it using Laplace-transform techniques.

Let's say that we have the sampled input signal, a reconstruction circuit denoted G(s), and an output denoted with the Laplace-transform variable Y(s). We can show the relationship as follows:

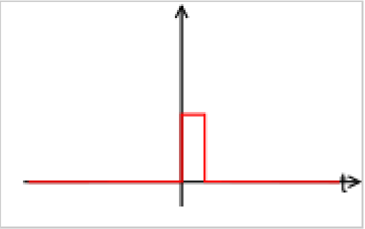
$$Y(s) = X^*(s)G(s)$$

Reconstruction circuits then, are physical devices that we can use to convert a digital, sampled signal into a continuous-time domain, so that we can take the Laplace transform of the output signal.

Zero order Hold

A **zero-order hold** circuit is a circuit that essentially inverts the sampling process: The value of the sampled signal at time t is held on the output for T time. The output waveform of a zero-order hold circuit therefore looks like a staircase approximation to the original waveform.

The transfer function for a zero-order hold circuit, in the Laplace domain, is written as such:

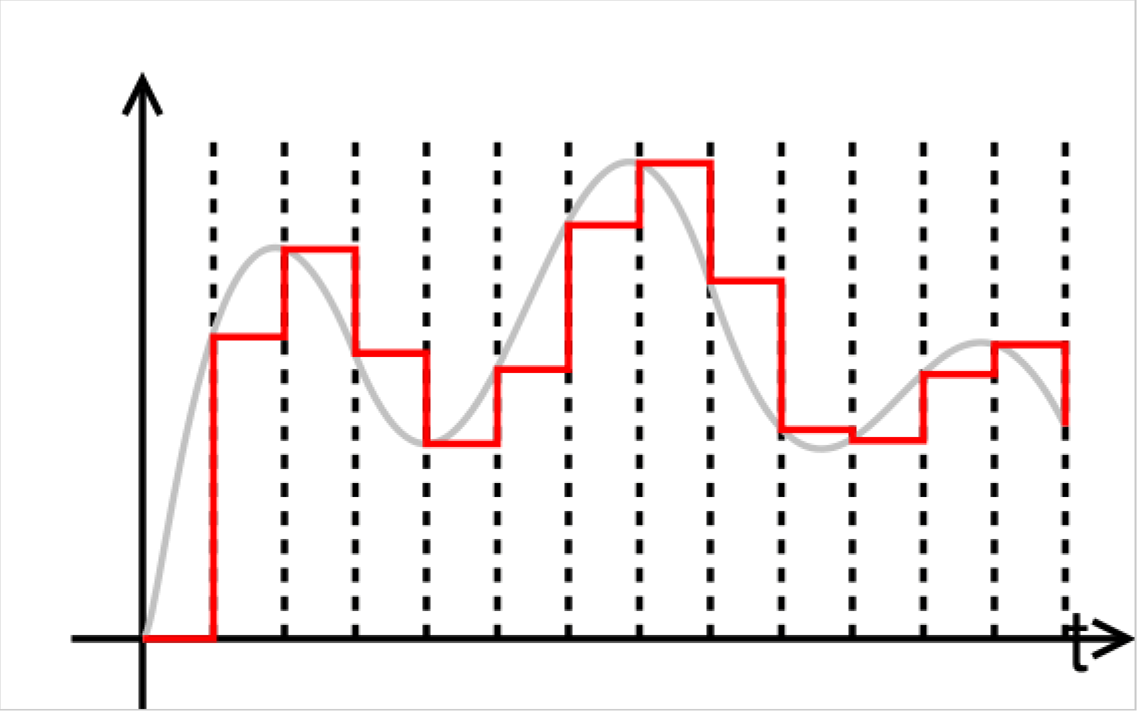


Zero-Order Hold impulse response

$$G_{h0} = \frac{1 - e^{-}Ts}{s}$$

[Zero Order Hold]

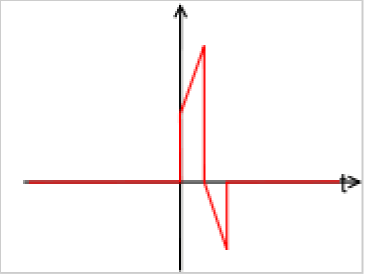
The Zero-order hold is the simplest reconstruction circuit, and (like the rest of the circuits on this page) assumes zero processing delay in converting between digital to analog.



A continuous input signal (grey) and the sampled signal with a zero-order hold (red)

First Order Hold

The zero-order hold creates a step output waveform, but this isn't always the best way to reconstruct the circuit. Instead, the **First-Order Hold** circuit takes the derivative of the waveform at the time t, and uses that derivative to make a guess as to where the output waveform is going to be at time (t + T). The first-order hold circuit then "draws a line" from the current position to the expected future position, as the output of the waveform.

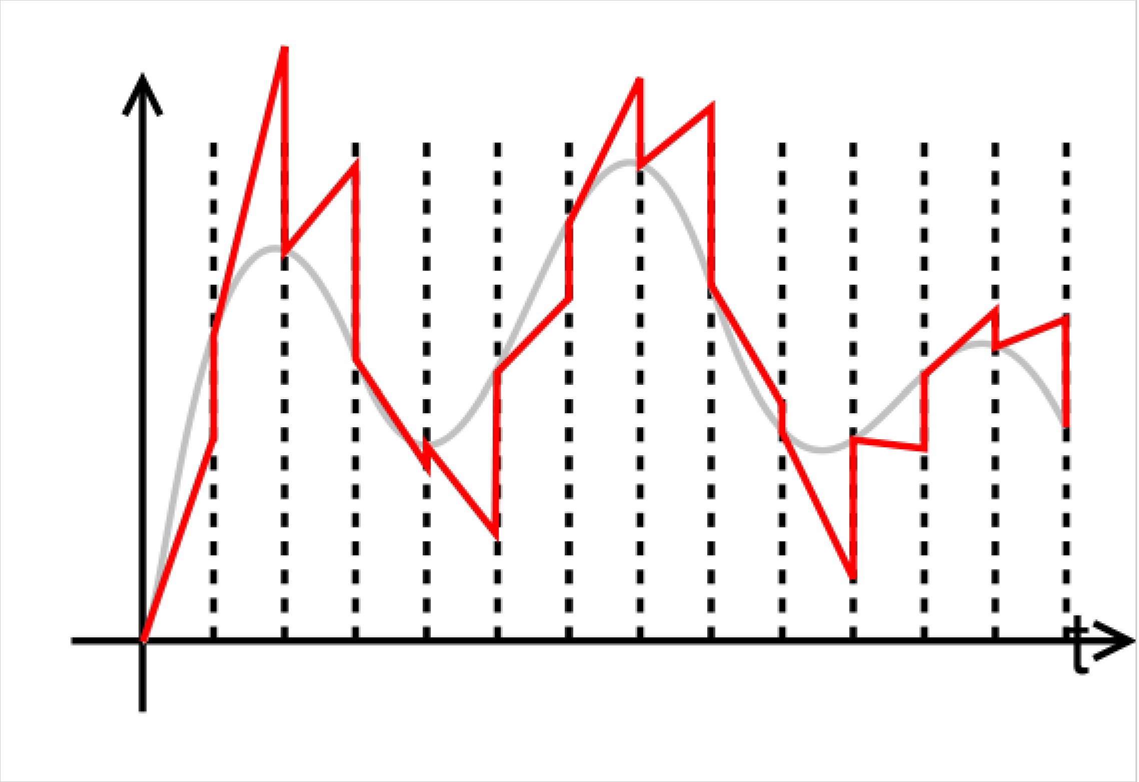


Impulse response of a first-order hold.

$$G_{h1} = \frac{1+Ts}{T}\left[\frac{1-e^{-Ts}}{s}\right]^2$$

[First Order Hold]

Keep in mind, however, that the next value of the signal will probably not be the same as the expected value of the text data point, and therefore the first-order hold may have a number of discontinuities.



An imput signal (grey) and the first-order hold circuit output (red)

Fractional Order Hold

The Zero-Order hold outputs the current value onto the output, and keeps it level throughout the entire bit time. The first-order hold uses the function derivative to predict the next value, and produces a series of ramp outputs to produce a fluctuating waveform. Sometimes however, neither of these solutions are desired, and therefore we have a compromise: **Fractional-Order Hold**. Fractional order hold acts like a mixture of the other two holding circuits, and takes a fractional number k as an argument. notice that k must be between 0 and 1 for this circuit to work correctly.

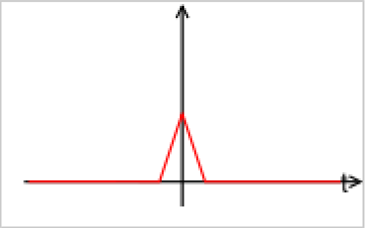
$$G_{hk} = (1 - ke^{-Ts})\frac{1 - e^{-Ts}}{s} + \frac{k}{Ts}(1 - e^{-Ts})^2$$

[Fractional Order Hold]

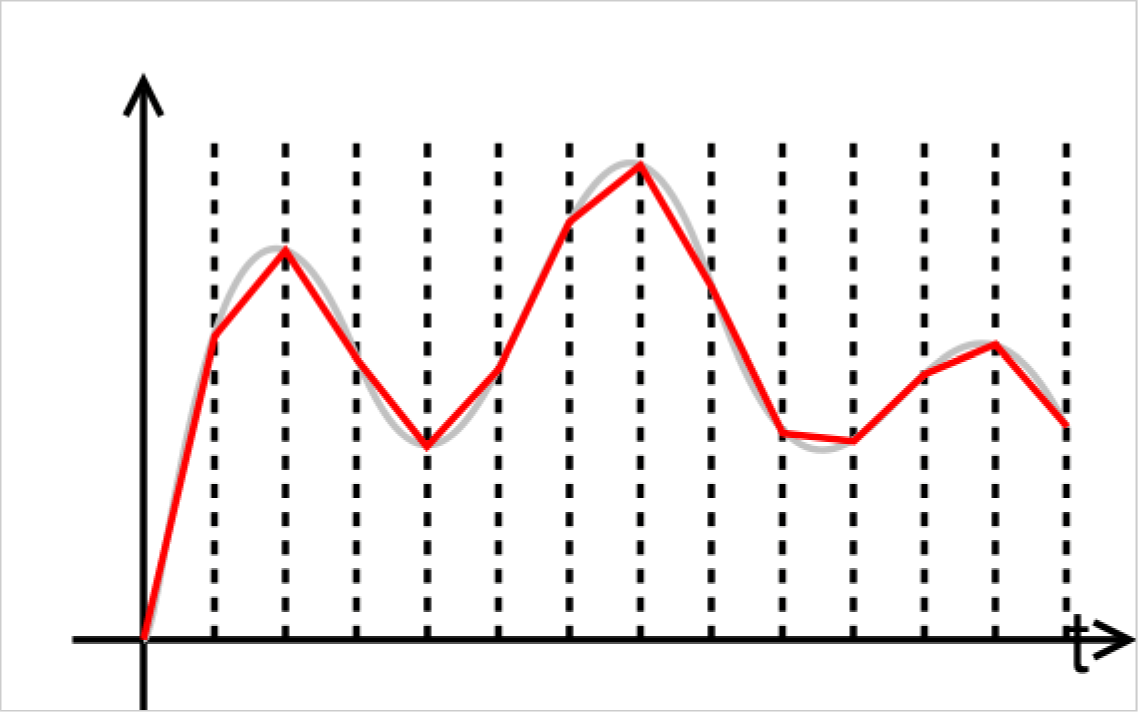
This circuit is more complicated than either of the other hold circuits, but sometimes added complexity is worth it if we get better performance from our reconstruction circuit.

Other Reconstruction Circuits

Another type of circuit that can be used is a **linear approximation** circuit.



Impulse response to a linear-approximation circuit.



An input signal (grey) and the output signal through a linear approximation circuit

Further Reading

- Hamming, Richard. "Numerical Methods for Scientists and Engineers" ISBN 0486652416
- Digital Signal Processing/Z Transform
- Residue Theory
- Analog and Digital Conversion

System Delays

Delays

A system can be built with an inherent **delay**. Delays are units that cause a time-shift in the input signal, but that don't affect the signal characteristics. An **ideal delay** is a delay system that doesn't affect the signal characteristics at all, and that delays the signal for an exact amount of time. Some delays, like processing delays or transmission delays, are unintentional. Other delays however, such as synchronization delays, are an integral part of a system. This chapter will talk about how delays are utilized and represented in the Laplace Domain.

Ideal Delays

An ideal delay causes the input function to be shifted forward in time by a certain specified amount of time. Systems with an ideal delay cause the system output to be delayed by a finite, predetermined amount of time.

Time Shifts

Let's say that we have a function in time that is time-shifted by a certain constant time period T. For convenience, we will denote this function as x(t - T). Now, we can show that the Laplace transform of x(t - T) is the following:

$$\mathcal{L}\{x(t-T)\} \Leftrightarrow e^{-sT}X(s)$$

What this demonstrates is that time-shifts in the time-domain become exponentials in the complex Laplace domain.

Shifts in the Z-Domain

Since we know the following general relationship between the Z Transform and the Star Transform:

$$z \Leftrightarrow e^{st}$$

We can show what a time shift in a discrete time domain becomes in the Z domain:

$$x[n-T] \Leftrightarrow z^{-T}X(z)$$

Delays and Stability

A time-shift in the time domain becomes an exponential increase in the laplace domain. This would seem to show that a time shift can have an effect on the stability of a system, and occasionally can cause a system to become unstable. We define a new parameter called the **time margin** as the amount of time that we can shift an input function before the system becomes unstable. If the system can survive any arbitrary time shift without going unstable, we say that the time margin of the system is infinite.

Delay Margin

When speaking of sinusoidal signals, it doesn't make sense to talk about "time shifts", so instead we talk about "phase shifts". Therefore, it is also common to refer to the time margin as the **phase margin** of the system. The phase margin denotes the amount of phase shift that we can apply to the system input before the system goes unstable.

We denote the phase margin for a system with a lowercase greek letter phi. Phase margin is defined as such for a second-order system:

$$\phi_m = \tan^{-1}\left[\frac{2\zeta}{(\sqrt{4\zeta^4+1}-2\zeta^2)^{1/2}}\right]$$

[Delay Margin]

Often times, the phase margin is approximated by the following relationship:

$$\phi_m \approx 100\zeta$$

[Delay Margin (approx)]

The greek letter zeta (ζ) is a quantity called the **damping ratio**, and we discuss this quantity in more detail in the next chapter.

Transform-Domain Delays

The ordinary Z-Transform does not account for a system which experiances an arbitrary time delay, or a processing delay. The Z-Transform can, however, be modified to account for an arbitrary delay. This new version of the Z-transform is frequently called the **Modified Z-Transform**, although in some literature (notably in Wikipedia), it is know as the **Advanced Z-Transform**.

Delayed Star Transform

To demonstrate the concept of an ideal delay, we will show how the star transform responds to a time-shifted input with a specified delay of time T. The function :$X^*(s, \Delta)$ is the delayed star transform with a delay parameter Δ. The delayed star transform is defined in terms of the star transform as such:

$$X^*(s, \Delta) = \mathcal{L}^*\left\{x(t-\Delta)\right\} = X(s)e^{-\Delta T s}$$

[Delayed Star Transform]

As we can see, in the star transform, a time-delayed signal is multiplied by a decaying exponential value in the transform domain.

Delayed Z-Transform

Since we know that the star transfrom is related to the z transform through the following change of variables:

$$z = e^{-sT}$$

We can interpret the above result to show how the Z-transform responds to a delay:

$$\mathcal{Z}(x[t-T]) = X(z)z^{-T}$$

This result is expected.

Now that we know how the Z transform responds to time shifts, it is often useful to generalize this behavior into a form known as the **Delayed Z-Transform**. The Delayed Z-Transform is a function of two variables, z and Δ, and is defined as such:

$$X(z,\Delta) = \mathcal{Z}\left\{x(t-\Delta)\right\} = \mathcal{Z}\left\{X(s)e^{-\Delta Ts}\right\}$$

And finally:

$$\mathcal{Z}(x[n],\Delta) = X(z,\Delta) = \sum_{n=-\infty}^{\infty} x[n-\Delta]z^{-n}$$

[Delayed Z Transform]

Modified Z-Transform

The Delayed Z-Transform has some uses, but mathematicians and engineers have decided that a more useful version of the transform was needed. The new version of the Z-Transform, which is similar to the Delayed Z-transform with a change of variables, is known as the **Modified Z-Transform**. The Modified Z-Transform is defined in terms of the delayed Z transform as follows:

$$X(z,m) = X(z,\Delta)\big|_{\Delta\to 1-m} = \mathcal{Z}\left\{X(s)e^{-\Delta Ts}\right\}\big|_{\Delta\to 1-m}$$

And it is defined explicitly:

$$X(z,m) = \mathcal{Z}(x[n],m) = \sum_{n=-\infty}^{\infty} x[n+m-1]z^{-n}$$

[Modified Z Transform]

Poles and Zeros

Poles and Zeros

Poles and Zeros are special values of a system where important events happen. The values of the poles and the zeros of a system determine whether the system is stable, and how well the system performs. Control systems, in general, can be designed simply by assigning specific values to the poles and zeros of the system.

Physically realizeable control systems must have a number of poles greater then or equal to the number of zeros. We will elaborate on this below.

Time-Domain Relationships

Let's say that we have a transfer function with 3 poles:

$$H(s) = \frac{a}{(s+l)(s+m)(s+n)}$$

The poles are located at s = **-l**, **-m**, **-n**. Now, we can use partial fraction expansion to separate out the transfer function:

$$H(s) = \frac{a}{(s+l)(s+m)(s+n)} = \frac{A}{s+l} + \frac{B}{s+m} + \frac{C}{s+n}$$

Using the inverse transform on each of these component fractions (looking up the transforms in our table), we get the following:

$$h(t) = Ae^{-lt}u(t) + Be^{-mt}u(t) + Ce^{-nt}u(t)$$

But, since s is a complex variable, **l m** and **n** can all potentially be complex numbers, with a real part (σ) and an imaginary part (jω). If we just look at the first term:

$$Ae^{-lt}u(t) = Ae^{-(\sigma_l + j\omega_l)t}u(t) = Ae^{-\sigma_l t}e^{-j\omega_l t}u(t)$$

Using **Euler's Equation** on the imaginary exponent, we get:

$$Ae^{-\sigma_l t}[cos(\omega_l t) - j\sin(\omega_l t)]u(t)$$

And taking the real part of this equation, we are left with our final result:

$$Ae^{-\sigma_l t}cos(\omega_l t)u(t)$$

We can see from this equation that every pole will have an exponential part, and a sinusoidal part to it's response. We can also go about constructing some rules:

1. if $\sigma_1 = 0$, the response of the pole is a perfect sinusoid (an oscillator)
2. if $\omega_1 = 0$, the response of the pole is a perfect exponential.
3. if $\sigma_1 > 0$, the exponential part of the response will decay towards zero.
4. if $\sigma_1 < 0$, the exponential part of the response will rise towards infinity.

From the last two rules, we can see that all poles of the system must have negative real parts, and therefore they must all have the form (s + l) for the system to be stable. We will discuss stability in later chapters.

What are Poles and Zeros

Let's say we have a transfer function defined as a ratio of two polynomials:

$$H(s) = \frac{N(s)}{D(s)}$$

Where N(s) and D(s) are simple polynomials. **Zeros** are the roots of N(s) (the numerator of the transfer function) obtained by setting

$$N(s) = 0$$

and solving for s. **Poles** are the roots of D(s) (the denominator of the transfer function), obtained by setting

$$D(s) = 0$$

and solving for s. Because of our restriction above, that a transfer function must not have more zeros then poles, we can state that the polynomial order of D(s) must be greater then or equal to the polynomial order of N(s)

The **polynomial order** of a function is the value of the highest exponent in the polynomial.

Example

Consider the transfer function:

$$H(s) = \frac{s+2}{s^2+0.25}$$

We define N(s) and D(s) to be the numerator and denominator polynomials, as such:

$$N(s) = s + 2$$

$$D(s) = s^2 + 0.25$$

We set N(s) to zero, and solve for s:

$$N(s) = s + 2 = 0 \rightarrow s = -2$$

So we have a zero at s → -2. Now, we set D(s) to zero, and solve for s to obtain the poles of the equation:

$$D(s) = s^2 + 0.25 = 0 \rightarrow s = +i\sqrt{0.25}, -i\sqrt{0.25}$$

And simplifying this gives us poles at: -i/2 , +i/2. Remember, s is a complex variable, and it can therefore take imaginary and complex values.

Effects of Poles and Zeros

As s approaches a zero, the numerator of the transfer function (and therefore the transfer function itself) approaches the value 0. When s approaches a pole, the denominator of the transfer function approaches zero, and the value of the transfer function approaches infinity. An output value of infinity should raise an alarm bell for people who are familiar with BIBO stability. We will discuss this later.

As we have seen above, the locations of the poles, and the values of the real and imaginary parts of the pole determine the response of the system. Real parts correspond to exponentials, and imaginary parts correspond to sinusoidal values.

Second-Order Systems

The cannonical form for a second order system is as follows:

$$H(s) = \frac{2\zeta}{s^2 + 2\zeta\omega s + \omega^2}$$

[Second-order transfer function]

Where ζ is called the **damping ratio** of the function, and ω is called the **natural frequency** of the system.

Damping Ratio

The **damping ratio** of a second-order system, denoted with the greek letter zeta (ζ), is a real number that defines the damping properties of the system. More damping has the effect of less percent overshoot, and faster settling time.

Natural Frequency

The natural frequency is occasionally written with a subscript:

$$\omega \rightarrow \omega_n$$

We will omit the subscript when it is clear that we are talking about the natural frequency, but we will include the subscript when we are using other values for the variable ω.

Higher-Order Systems

Modern Controls

The modern method of controls uses systems of special state-space equations to model and manipulate systems. The state variable model is broad enough to be useful in describing a wide range of systems, including systems that cannot be adequately described using the Laplace Transform. These chapters will require the reader to have a solid background in linear algebra, and multi-variable calculus.

State-Space Equations

Time-Domain Approach

The "Classical" method of controls (what we have been studying so far) has been based mostly in the transform domain. When we want to control the system in general we use the Laplace transform (Z-Transform for digital systems) to represent the system, and when we want to examine the frequency characteristics of a system, we use the Fourier Transform. The question arises, why do we do this:

Let's look at a basic second-order Laplace Transform transfer function:

$$\frac{Y(s)}{X(s)} = G(s) = \frac{1+s}{1+2s+5s^2}$$

And we can decompose this equation in terms of the system inputs and outputs:

$$(1+2s+5s^2)Y(s) = (1+s)X(s)$$

Now, when we take the inverse laplace transform of our equation, we can see the terrible truth:

$$y(t) + 2\frac{dy(t)}{dt} + 5\frac{d^2y(t)}{dt^2} = x(t) + \frac{dx(t)}{dt}$$

That's right, the laplace transform is hiding the fact that we are actually dealing with second-order differential equations. The laplace transform moves us out of the time-domain (messy, second-order ODEs) into the complex frequency domain (simple, second-order polynomials), so that we can study and manipulate our systems more easily. So, why would anybody want to work in the time domain?

It turns out that if we decompose our second-order (or higher) differential equations into multiple first-order equations, we can find a new method for easily manipulating the system *without having to use integral transforms*. The solution to this problem is **state variables** . By taking our multiple first-order differential equations, and analyzing them in vector form, we can not only do the same things we were doing in the time domain using simple matrix algebra, but now we can easily account for systems with multiple inputs and multiple outputs, without adding much unnecessary complexity. All these reasons demonstrate why the "modern" state-space approach to controls has become so popular.

State-Space

In a state space system, the internal state of the system is explicitly accounted for by an equation known as the **state equation**. The system output is given in terms of a combination of the current system state, and the current system input, through the **output equation**. These two equations form a linear system of equations known collectively as **state-space equations**. The state-space is the linear vector space that consists of all the possible internal states of the system. Because the state-space must be finite, a system can only be described by state-space equations if the system is lumped.

For a system to be modeled using the state-space method, the system must meet these requirements:

1. **The system must be linear**
2. **The system must be lumped**

State Variables

When modeling a system using a state-space equation, we first need to define three vectors:

Input variables
: A SISO (Single Input Single Output) system will only have a single input value, but a MIMO system may have multiple inputs. We need to define all the inputs to the system, and we need to arrange them into a vector.

Output variables
: This is the system output value, and in the case of MIMO systems, we may have several. Output variables should be independant of one another, and only dependant on a linear combination of the input vector and the state vector.

State Variables
: The state variables represent values from inside the system, that can change over time. In an electric circuit, for instance, the node voltages or the mesh currents can be state variables. In a mechanical system, the forces applied by springs, gravity, and dashpots can be state variables.

We denote the input variables with a u, the output variables with y, and the state variables with x. In essence, we have the following relationship:

$$y = f(x, u)$$

Where f( ) is our system. Also, the state variables can change with respect to the current state and the system input:

$$x' = g(x, u)$$

Where x' is the rate of change of the state variables. We will define f(u, x) and g(u, x) in the next chapter.

Multi-Input, Multi-Output

In the Laplace domain, if we want to account for systems with multiple inputs and multiple outputs, we are going to need to rely on the principle of superposition, to create a system of simultaneous laplace equations for each output and each input. For such systems, the classical approach not only doesn't simplify the situation, but because the systems of equations need to be transformed into the frequency domain first, manipulated, and then transformed back into the time domain, they can actually be more difficult to work with. However, the Laplace domain technique can be combined with the State-Space techniques discussed in the next few chapters to bring out the best features of both techniques.

State-Space Equations

In a state-space system representation, we have a system of two equations: an equation for determining the state of the system, and another equation for determining the output of the system. We will use the variable y(t) as the output of the system, x(t) as the state of the system, and u(t) as the input of the system. We use the notation x'(t) to denote the future state of the system, as dependant on the current state of the system and the current input. Symbolically, we say that there are transforms **g** and **h**, that display this relationship:

$$x'(t) = g[t_0, t, x(t), x(0), u(t)]$$
$$y(t) = h[t, x(t), u(t)]$$

The first equation shows that the system state is dependant on the previous system state, the initial state of the system, the time, and the system inputs. The second equation shows that the system output is depentant on the current system state, the system input, and the current time.

If the system state change x'(t) and the system output y(t) are linear combinations of the system state and unput vectors, then we can say the systems are linear systems, and we can rewrite them in matrix form:

> **Note**:
> If x'(t) and y(t) are not linear combinations of x(t) and u(t), the system is said to be **nonlinear**. We will attempt to discuss non-linear systems in a later chapter.

$$x' = A(t)x(t) + B(t)u(t)$$

[State Equation]

$$y(t) = C(t)x(t) + D(t)u(t)$$

[Output Equation]

If the systems themselves are time-invariant, we can re-write this as follows:

$$x' = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

These equations show that in a given system, the current output is dependant on the current input and the current state. The **State Equation** shows the relationship between the system's current state and it's input, and the future state of the system. The **Output Equation** shows the relationship between the system state and the output. These equations show that in a given system, the current output is dependant on the current input and the current state. The future state is also dependant on the current state and the current input.

It is important to note at this point that the state space equations of a particular system are not unique, and there are an infinite number of ways to represent these equations by manipulating the A, B, C and D matrices using row operations. There are a number of "standard forms" for these matricies, however, that make certain computations easier. Converting between these forms will require knowledge of linear algebra.

> *Any system that can be described by a finite number of n^{th} order differential equations or n^{th} order difference equations, or any system that can be approximated by by them, can be described using state-space equations. The general solutions to the state-space equations, therefore, are solutions to all such sets of equations.*

Digital Systems

For digital systems, we can write similar equations, using discrete data sets:

$$x[k+1] = Ax[k] + Bu[k]$$

$$y[k] = Cx[k] + Du[k]$$

We will show how to obtain all these equations below.

Matrices: A B C D

In our time-invariant state space equations:

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

We have 4 constant matrices: A, B, C, and D. We will explain these matrices below:

Matrix A

Matrix A is the **system matrix**, and relates how the current state affects the state change x'. If the state change is not dependant on the current state, A will be the zero matrix. The exponential of the state matrix, e^{At} is called the **state transition matrix**, and is an important function that we will describe below.

Matrix B

Matrix B is the **control matrix**, and determines how the system input affects the state change. If the state change is not dependant on the system input, then B will be the zero matrix.

Matrix C

Matrix C is the **output matrix**, and determines the relationship between the system state and the system output.

Matrix D

Matrix D is the **feedforward matrix**, and allows for the system input to affect the system output directly. A basic feedback system like those we have previously considered do not have a feedforward element, and therefore for most of the systems we have already considered, the D matrix is the zero matrix.

Matrix Dimensions

Because we are adding and multiplying multiple matrices and vectors together, we need to be absolutely certain that the matrices have compatable dimensions, or else the equations will be undefined. For integer values p, q, and r, the dimensions of the system matrices and vectors are defined as follows:

Vectors	Matrices
▪ $x : p \times 1$ ▪ $x' : p \times 1$ ▪ $u : q \times 1$ ▪ $y : r \times 1$	▪ $A : p \times p$ ▪ $B : p \times q$ ▪ $C : r \times p$ ▪ $D : r \times q$

If the matrix and vector dimensions do not agree with one another, the equations are invalid and the results will be meaningless. Matrices and vectors must have compatable dimensions or them can not be combined using matrix operations.

Relating Continuous and Discrete Systems

Continuous and discrete systems that perform similarly can be related together through a set of relationships. It should come as no surprise that a discrete system and a continuous system will have different characteristics and different coefficient matrices. If we consider that a discrete system is the same as a continuous system, except that it is sampled with a sampling time T, then the relationships below will hold.

Here, we will use "d" subscripts to denote the system matrices of a discrete system, and we will use a "c" subscript to denote the system matrices of a continuous system. T is the sampling time of the digital system.

$A_d = e^{A_c T}$
$B_d = A_c^{-1}(A_d - I)B_c$
$C_d = C_c$
$D_d = D_c$

If the A_c matrix is singular, and we cannot find it's inverse, we can instead define B_d as:

$$B_d = \int_0^T e^{A\tau} d\tau$$

If A is nonsingular, this integral equation will reduce to the equation listed above.

Obtaining the State-Space Equations

The beauty of state equations, is that they can be used to transparently describe systems that are both continuous and discrete in nature. Some texts will differentiate notation between discrete and continuous cases, but this wikitext will not. Instead we will opt to use the generic coefficient matrices A, B, C and D. Other texts may use the letters F, H, and G for continuous systems and Γ, and Θ for use in discrete systems. However, if we keep track of our time-domain system, we don't need to worry about such notations.

From Differential Equations

Let's say that we have a general 3rd order differential equation in terms of input(u) and output (y):

$$\frac{d^3y(t)}{dt^3} + a_2\frac{d^2y(t)}{dt^2} + a_1\frac{dy(t)}{dt} + a_0y(t) = u(t)$$

We can create the state variable vector x in the following manner:

$$x_1 = y(t)$$
$$x_2 = \frac{dy(t)}{dt}$$
$$x_3 = \frac{d^2y(t)}{dt^2}$$

Which now leaves us with the following 3 first-order equations:

$$x_1' = x_2$$
$$x_2' = x_3$$
$$x_3' = \frac{d^3y(t)}{dt^3}$$

Now, we can define the state vector x in terms of the individual x components, and we can create the future state vector as well:

$$x = \begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} \; x' = \begin{bmatrix}x_1'\\x_2'\\x_3'\end{bmatrix}$$

And with that, we can assemble the state-space equations for the system:

$$x' = \begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\\-a_0 & -a_1 & -a_2\end{bmatrix}x(t) + \begin{bmatrix}0\\0\\1\end{bmatrix}u(t)$$

$$y(t) = \begin{bmatrix}1 & 0 & 0\end{bmatrix}x(t)$$

Granted, this is only a simple example, but the method should become apparent to most readers.

From Difference Equations

Now, let's say that we have a 3rd order difference equation, that describes a discrete-time system:

$$y[n+3] + a_2y[n+2] + a_1y[n+1] + a_0y[n] = u[n]$$

From here, we can define a set of discrete state variables x in the following manner:

$$x_1[n] = y[n]$$

$$x_2[n] = y[n+1]$$

$$x_3[n] = y[n+2]$$

Which in turn gives us 3 first-order difference equations:

$$x_1[n+1] = y[n+1] = x_2[n]$$

$$x_2[n+1] = y[n+2] = x_3[n]$$

$$x_3[n+1] = y[n+3]$$

Again, we say that matrix x is a vertical vector of the 3 state variables we have defined, and we can write our state equation in the same form as if it were a continuous-time system:

$$x[n+1] = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} x[n] + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u[n]$$

$$y[n] = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x[n]$$

From Transfer Functions

The method of obtaining the state-space equations from the laplace domain transfer functions are very similar to the method of obtaining them from the time-domain differential equations. In general, let's say that we have a transfer function of the form:

$$T(s) = \frac{s^m + a_{m-1}s^{m-1} + \cdots + a_0}{s^n + b_{n-1}s^{n-1} + \cdots + b_0}$$

We can write our A, B, C, and D matrices as follows:

$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -b_0 & -b_1 & -b_2 & \cdots & -b_n \end{bmatrix}$
$B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$
$C = \begin{bmatrix} a_0 & a_1 & \cdots & a_{m-1} \end{bmatrix}$
$D = 0$

This form of the equations is known as the **controllable cannonical form** of the system matrices, and we will discuss this later.

State-Space Representation

As an important note, remember that the state variables x are user-defined and therefore are abitrary. There are any number of ways to define x for a particular problem, each of which are going to lead to different state space equations.

Note: There are an infinite number of equivalent ways to represent a system using state-space equations. Some ways are better then others. Once the state-space equations are obtained, they can be manipulated to take a particular form if needed.

Consider the previous continuous-time example. We can rewrite the equation in the form

$$\frac{d}{dt}\left[\frac{d^2y(t)}{dt^2} + a_2\frac{dy(t)}{dt} + a_1y(t)\right] + a_0y(t) = u(t).$$

We now define the state variables

$$x_1 = y(t)$$

$$x_2 = \frac{dy(t)}{dt}$$

$$x_3 = \frac{d^2y(t)}{dt^2} + a_2\frac{dy(t)}{dt} + a_1y(t)$$

with first-order derivatives

$$x_1' = \frac{dy(t)}{dt} = x_2$$

$$x_2' = \frac{d^2y(t)}{dt^2} = -a_1x_1 - a_2x_2 + x_3$$

$$x_3' = -a_0y(t) + u(t)$$

The state-space equations for the system will then be given by

$$x' = \begin{bmatrix} 0 & 1 & 0 \\ -a_1 & -a_2 & 1 \\ -a_0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x(t)$$

x may also be used in any number of variable transformations, as a matter of mathematical convenience.

However, the variables y and u correspond to physical signals, and may not be arbitrarily selected, redefined, or transformed as x can be.

Solutions for Linear Systems

State Equation Solutions

The solutions in this chapter are heavily rooted in prior knowledge of Differential Equations. Readers should have a prior knowledge of that subject before reading this chapter.

The state equation is a first-order linear differential equation, or (more precisely) a system of linear differential equations. Because this is a first-order equation, we can use results from Differential Equations to find a general solution to the equation in terms of the state-variable *x*. Once the state equation has been solved for *x*, that solution can be plugged into the output equation. The resulting equation will show the direct relationship between the system input and the system output, without the need to account explicitly for the internal state of the system. The sections in this chapter will discuss the solutions to the state-space equations, starting with the easiest case (Time-invariant, no input), and ending with the most difficult case (Time-variant systems).

Solving for x(t) With Zero Input

Looking again at the state equation:

$$x' = Ax(t) + Bu(t)$$

We can see that this equation is a first-order differential equation, except that the variables are vectors, and the coefficients are matrices. However, because of the rules of matrix calculus, these distinctions don't matter. We can ignore the input term (for now), and rewrite this equation in the following form:

$$\frac{dx(t)}{dt} = Ax(t)$$

And we can separate out the variables as such:

$$\frac{dx(t)}{x(t)} = Adt$$

Integrating both sides, and raising both sides to a power of e, we obtain the result:

$$x(t) = e^{At+C}$$

Where C is a constant. We can assign $D = e^C$ to make the equation easier, but we also know that D will then be the initial conditions of the system. This becomes obvious if we plug the value zero into the variable *t*. The final solution to this equation then is given as:

$$x(t) = x(t_0)e^{At}$$

We call the matrix exponential e^{At} the **state-transition matrix**, and calculating it, while difficult at times, is crucial to analyzing and manipulating systems. We will talk more about calculating the matrix exponential below.

Solving for x(t) With Non-Zero Input

If, however, our input is non-zero (as is generally the case with any interesting system), our solution is a little bit more complicated. Notice that now that we have our input term in the equation, we will no longer be able to separate the variables and integrate both sides easily.

$$x'(t) = Ax(t) + Bu(t)$$

We subtract to get the x(t) on the left side, and then we do something curious; we premultiply both sides by the inverse state transition matrix:

$$e^{-At}x'(t) - e^{-At}Ax(t) = e^{-At}Bu(t)$$

The rationale for this last step may seem fuzzy at best, so we will illustrate the point with an example:

Example: Take the derivative of the following with respect to time:

$$e^{-At}x(t)$$

The product rule from differentiation reminds us that if we have two functions multiplied together:

$$f(t)g(t)$$

and we differentiate with respect to t, then the result is:

$$f(t)g'(t) + f'(t)g(t)$$

If we set our functions accordingly:

$$\begin{matrix} f(t) = e^{-At} & f'(t) = -Ae^{-At} \\ g(t) = x(t) & g'(t) = x'(t) \end{matrix}$$

Then the output result is:

$$e^{At}x(t) - e^{-At}Ax(t)$$

If we look at this result, it is the same as from our equation above.

Using the result from our example, we can condense the left side of our equation into a derivative:

$$\frac{d(e^{-At}x(t))}{dt} = e^{-At}Bu(t)$$

Now we can integrate both sides, from the initial time (t_0) to the current time (t), using a dummy variable τ, we will get closer to our result. Finally, if we premultiply by e^{At}, we get our final result:

$$x(t) = e^{At-t_0}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau$$

[General State Equation Solution]

If we plug this solution into the output equation, we get:

[General Output Equation Solution]

$$y(t) = Ce^{At-t_0}x(t_0) + C\int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau + Du(t)$$

This is the general Time-Invariant solution to the state space equations, with non-zero input. These equations are important results, and students who are interested in a further study of control systems would do well to memorize these equations.

Solving for x[n]

Similar to the continuous time systems above, we can find a general solution to the discrete time difference equations.

$$x[n] = A^n x[0] + \sum_{m=0}^{n-1} A^{n-1-m}Bu[n]$$

[General State Equation Solution]

[General Output Equation Solution]

$$y[n] = CA^n x[0] + \sum_{m=0}^{n-1} CA^{n-1-m}Bu[n] + Du[n]$$

State-Transition Matrix

The state transition matrix, e^{At}, is an important part of the general state-space solutions for the time-invariant cases listed above. Calculating this matrix exponential function is one of the very first things that should be done when analyzing a new system, and the results of that calculation will tell important information about the system in question.

More information about **matrix exponentials** can be found in:
Matrix Exponentials

The matrix exponential can be calculated directly by using a Taylor-Series expansion:

$$e^{At} = \sum_{n=0}^{\infty} \frac{(At)^n}{n!}$$

Also, we can attempt to diagonalize the matrix A into a **diagonal matrix** or a **Jordan Cannonical matrix**. The exponential of a diagonal matrix is simply the diagonal elements individually raised to that exponential. The exponential of a Jordan cannonical matrix is slightly more complicated, but there is a useful pattern that can be exploited to find the solution quickly. Interested readers should read the relevant passages in Engineering Analysis

More information about **diagonal matrices** and **Jordan-form matrices** can be found in:
Diagonalization
Matrix Functions

The state transition matrix, and matrix exponentials in general are very important tools in control engineering.

General Time Variant Solution

The state-space equations can be solved for time-variant systems, but the solution is significantly more complicated then the time-invariant case. Our state equation is given as follows:

$$x'(t) = A(t)x(t) + B(t)u(t)$$

We can say that the general solution to time-variant state-equation is defined as:

$$x(t) = \phi(t, t_0)x(t_0) + \int_{t_0}^{t} \phi(\tau, t_0)B(\tau)u(\tau)d\tau$$ **[Time-Variant General Solution]**

The function φ is called the **state-transition matrix**, because it (like the matrix exponential from the time-invariant case) controls the change for states in the state equation. However, unlike the time-invariant case, we cannot define this as a simple exponential. In fact, φ can't be defined in general, because it will actually be a different function for every system. However, the state-transition matrix does follow some basic properties that we can use to determine the state-transition matrix.

In a time-invariant system, the general solution is obtained when the state-transition matrix is determined. For that reason, the first thing (and the most important thing) that we need to do here is find that matrix. We will discuss the solution to that matrix below.

State Transition Matrix

The state transtion matrix φ satisfies the following relationships:

$$\frac{\partial \phi(t, t_0)}{\partial t} = A(t)\phi(t, t_0)$$

$$\phi(\tau, \tau) = I$$

And φ also must have the following properties:

Note:
The state transition matrix φ is a matrix function of two variables (we will say *t* and τ). Once the form of the matrix is solved, we will plug in the initial time, t_0 in place of the variable τ. Because of the nature of this matrix, and the properties that it must satisfy, this matrix typically is composed of exponential or sinusoidal functions. The exact form of the state-

1.	$\phi(t_2, t_1)\phi(t_1, t_0) = \phi(t_2, t_0)$
2.	$\phi^{-1}(t, \tau) = \phi(\tau, t)$
3.	$\phi^{-1}(t, \tau)\phi(t, \tau) = I$
4.	$\frac{d\phi(t, t_0)}{dt} = A(t)$

transition matrix is dependant on the system itself, and the form of the system's differential equation. There is no single "template solution" for this matrix.

If the system is time-invariant, we can define φ as:

$$\phi(t, t_0) = e^{A(t-t_0)}$$

The reader can verify that this solution for a time-invariant system satisfies all the properties listed above. However, in the time-variant case, there are many different functions that may satisfy these requirements, and the solution is dependant on the structure of the system. The state-transition matrix must be determined before analysis on the time-varying solution can continue. We will discuss some of the methods for determining this matrix below.

Time-Variant, Zero Input

As the most basic case, we will consider the case of a system with zero input. If the system has no input, then the state equation is given as:

$$x'(t) = A(t)x(t)$$

And we are interested in the response of this system in the time interval T = (a, b). The first thing we want to do in this case is find a **fundamental matrix** of the above equation. The fundamental matrix is related

Fundamental Matrix

Given the equation:

$$x'(t) = A(t)x(t)$$

Here, *x* is an *n* × 1 vector, and A is an *n* × *n* matrix.

The solutions to this equation form an *n*-dimensional vector space in the interval T = (a, b). Any set of *n* linearly-independent solutions {x_1, x_2, ..., x_n} to the equation above is called a **fundamental set** of solutions.

A **fundamental matrix** is formed by creating a matrix out of the *n* fundamental vectors. We will denote the fundamental matrix with a script capital X:

$$\mathcal{X} = [x_1 x_2 \cdots x_n]$$

Readers who have a background in Linear Algebra may recognize that the fundamental set is a **basis set** for the solution space. Any basis set that spans the entire solution space is a valid fundamental set.

The fundamental matrix will satisfy the state equation:

$$\mathcal{X}(t) = A(t)\mathcal{X}(t)$$

Also, *any matrix that solves this equation can be a fundamental matrix* if and only if the determinant of the matrix is non-zero for all time *t* in the interval T. The determinant must be non-zero, because we are going to use the inverse of the fundamental matrix to solve for the state-transition matrix.

State Transition Matrix

Once we have the fundamental matrix of a system, we can use it to find the state transition matrix of the system:

$$\phi(t, t_0) = \mathcal{X}(t)\mathcal{X}^{-1}(t_0)$$

The inverse of the fundamental matrix exists, because we specify in the definition above that it must have a non-zero determinant, and therefore must be non-singular. The reader should note that this is only one possible method for determining the state transtion matrix, and we will discuss other methods below.

Example: 2-Dimensional System

Given the following fundamental matrix, Find the state-transition matrix.

$$\mathcal{X}(t) = \begin{bmatrix} e^{-t} & \frac{1}{2}e^{t} \\ 0 & e^{-t} \end{bmatrix}$$

The state-transition matrix is given by:

$$\phi(t, t_0) = \begin{bmatrix} e^{-t+t_0} & \frac{1}{2}[e^{t+t_0} - e^{-t+3t_0} \\ 0 & e^{t-t_0} \end{bmatrix}$$

Other Methods

There are other methods for finding the state transition matrix besides having to find the fundamental matrix.

Method 1

If A(t) is triangular (upper or lower triangular), the state transition matrix can be determined by sequentially integrating the individual rows of the state equation.

Method 2

If for every τ and t, the state matrix commutes as follows:

$$A(t)\left[\int_\tau^t A(\zeta)d\zeta\right] = \left[\int_\tau^t A(\zeta)d\zeta\right]A(t)$$

Then the state-transition matrix can be given as:

$$\phi(t, \tau) = e^{\int_\tau^t A(\zeta)d\zeta}$$

It will be left as an excercise for the reader to prove that if A(t) is time-invariant, that the equation in **method 2** above will reduce to the state-transition matrix $e^{At-\tau}$.

Time-Variant, Non-zero Input

Eigenvalues and Eigenvectors

Eigenvalues and Eigenvectors

The eigenvalues and eigenvectors of the system matrix play a key role in determining the response of the system. It is important to note that only square matrices have eigenvalues and eigenvectors associated with them. Non-square matrices cannot be analyzed using the methods below.

The word "eigen" is from the German for "characteristic", and so this chapter could also be called "Characteristic values and characteristic vectors", although that is more verbose, and less well-known of a description of the topics discussed in this chapter. Eigenvalues and Eigenvectors have a number of properties that make them valuable tools in analysis, and they also have a number of valuable relationships with the matrix from which they are derived. Computing the eigenvalues and the eigenvectors of the system matrix is one of the most important things that should be be done when beginning to analyze a system matrix, second only to calculating the matrix exponential of the system matrix.

The eigenvalues and eigenvectors of the system determine the relationship between the individual system state variables (the members of the *x* vector), the response of the system to inputs, and the stability of the system. Also, the eigenvalues and eigenvectors can be used to calculate the matrix exponential of the system matrix (through spectral decomposition). The remainder of this chapter will discuss eigenvalues and eigenvectors, and the ways that they affect their respective systems.

Characteristic Equation

The characteristic equation of the system matrix A is given as:

$$Av = \lambda v$$

[Matrix Characteristic Equation]

Where λ are scalar values called the **eigenvalues**, and *v* are the corresponding **eigenvectors**. To solve for the eigenvalues of a matrix, we can take the following determinant:

$$|A - \lambda I| = 0$$

To solve for the eigenvectors, we can then add an additional term, and solve for *v*:

$$|A - \lambda I|v = 0$$

Another value worth finding are the **left eigenvectors** of a system, defined as *w* in the modified characteristic equation:

$$wA = \lambda w$$

[Left-Eigenvector Equation]

For more information about eigenvalues, eigenvectors, and left eigenvectors, read the appropriate sections in the following books:

- Linear Algebra
- Engineering Analysis

Diagonalization

If the matrix A has a complete set of distinct eigenvalues, the matrix can be **diagonalized**. A diagonal matrix is a matrix that only has entries on the diagonal, and all the rest of the entries in the matrix are zero. We can define a **transformation matrix**, T, that satisfies the diagonalization transformation:

$$A = TDT^{-1}$$

Which in turn will satisfy the relationship:

$$e^At = Te^{Dt}T^{-1}$$

The left-hand side of the equation may look more complicated, but because D is a diagonal matrix here (not to be confused with the feed-forward matrix from the output equation), the calculations are much easier.

We can define the transition matrix, and the inverse transition matrix in terms of the eigenvectors and the left eigenvectors:

$$T = \begin{bmatrix} v_1 & v_2 & v_3 & \cdots & v_n \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} w_1' \\ w_2' \\ w_3' \\ \vdots \\ w_n' \end{bmatrix}$$

Exponential Matrix Decomposition

For more information about spectral decomposition, see:
Spectral Decomposition

A matrix exponential can be decomposed into a sum of the eigenvectors, eigenvalues, and left eigenvalues, as follows:

$$e^{At} = \sum_{i=0}^{n} e^{\lambda_i t} v_i w_i' x(0)$$

Notice that this equation only holds in this form if the matrix A has a complete set of n distinct eigenvalues. Since w'_i is a row vector, and x(0) is a column vector of the initial system states, we can combine those two into a scalar coefficient α:

$$e^{At} = \sum_{i=0}^{n} \alpha_i e^{\lambda_i t} v_i$$

Since the state transition matrix determines how the system responds to an input, we can see that the system eigenvalues and eigenvectors are a key part of the system response. Let us plug this decomposition into the general solution to the state equation:

[State Equation Spectral Decomposition]

$$x(t) = \sum_{i=0}^{n} \alpha_i e^{\lambda_i t} v_i x(0) + \sum_{i=0}^{n} \int_0^t e^{\lambda_i (t-\tau)} v_i w_i' B u(\tau) d\tau$$

We will talk about this equation in the following sections.

Decoupling

If a system can be designed such that the following relationship holds true:

$$w_i' B = 0$$

then the system response from that particular eigenvalue will not be affected by the system input *u*, and we say that the system has been **decoupled**. Such a thing is difficult to do in practice. For people who are familiar with linear algebra, the left-eigenvector of the matrix A must be in the *null space* of the matrix B.

Condition Number

With every matrix there is associated a particular number called the **condition number** of that matrix. The condition number tells a number of things about a matrix, and it is worth calculating. The condition number, *k*, is defined as:

$$k = \frac{\|w_i\| \|v_i\|}{|w_i' v_i|}$$

[Condition Number]

Systems with smaller condition numbers are better, for a number of reasons:

1. Large condition numbers lead to a large transient response of the system
2. Large condition numbers make the system eigenvalues more sensitive to changes in the system.

We will discuss the issue of **eigenvalue sensitivity** more in a later section.

Stability

We will talk about stability at length in later chapters, but is a good time to point out a simple fact concerning the eigenvalues of the system. Notice that if the eigenvalues of the system matrix A are *postive*, or (if they are complex) that they have positive real parts, that the system state (and therefore the system output, scaled by the C matrix) will approach infinity as time *t* approaches infinity. In essence, if the eigenvalues are positive, the system will not satisfy the condition of BIBO stability, and will therefore become *unstable*.

Another factor that is worth mentioning is that a manufactured system *never exactly matches the system model*, and there will always been inaccuracies in the specifications of the component parts used, *within a certain tolerance*. As such, the system matrix will be slightly different from the mathematical model of the system (although good systems will not be severly different), and therefore the eigenvalues and eigenvectors of the system will not be the same values as those derived from the model. These facts give rise to several results:

1. Systems with high *condition numbers* may have eigenvalues that differ by a large amount from those derived from the mathematical model. This means that the system response of the physical system may be very different from the intended response of the model.
2. Systems with high condition numbers may become *unstable* simply as a result of inaccuracies in the component parts used in the manufacturing process.

For those reasons, the system eigenvalues and the condition number of the system matrix are highly important variables to consider when analyzing and designing a system. We will discuss the topic of stability in more detail in later chapters.

Non-Unique Eigenvalues

The decomposition above only works if the matrix A has a full set of n distinct eigenvalues (and corresponding eigenvectors). If A does not have n distinct eigenvectors, then a set of **generalized eigenvectors** need to be determined. The generalized eigenvectors will produce a similar matrix that is in **jordan cannonical form**, not the diagonal form we were using earlier.

Generalized Eigenvectors

Generalized eigenvectors can be generated using the following equation:

$$(A - \lambda I)v_{n+1} = v_n$$

[Generalized Eigenvector Generating Equation]

if *d* is the number of times that a given eigenvalue is repeated, and *p* is the number of unique eigenvectors derived from those eigenvalues, then there will be *q* = *d* - *p* generalized eigenvectors. Generalized eigenvectors are developed by plugging in the regular eigenvectors into the equation above (v_n). Some regular eigenvectors might not produce any non-trivial generalized eigenvectors. Generalized eigenvectors may also be plugged into the equation above to produce additional generalized eigenvectors. It is important to note that the generalized eigenvectors form an ordered series, and they must be kept in order during analysis or the results will not be correct.

Examples: Repeated Eigenvalues

Example 1: We have a 5 × 5 matrix A with eigenvalues $\lambda = 1, 1, 1, 2, 2$. For $\lambda = 1$, there is 1 distinct eigenvector *a*. For $\lambda = 2$ there is 1 distinct eigenvector *b*. From *a*, we generate the generalized eigenvector *c*, and from *c* we can generate vector *d*. From the eigevector *b*, we generate the generalized eigevector *e*. In order our eigenvectors are listed as:

[*a c d b e*]

Notice how *c* and *d* are listed in order after the eigenvector that they are generated from, *a*. Also, we could reorder this as:

[*b e a c d*]

because the generalized eigenvectors are listed in order after the regular eigenvector that they are generated from. Regular eigenvectors can be listed in any order.

Example 2: We have a 4 × 4 matrix A with eigenvalues $\lambda = 1, 1, 1, 2$. For $\lambda = 1$ we have two eigevectors, *a* and *b*. For $\lambda = 2$ we have an eigenvector *c*.

We need to generate a fourth eigenvector, *d*. The only eigenvalue that needs another eigenvector is $\lambda = 1$, however there are already two eigevectors associated with that eigenvalue, and only one of them will generate a non-trivial generalized eigenvector. To figure out which one works, we need to plug both vectors into the generating equation:

$$(A - \lambda I)|_{\lambda=1} d = a$$
$$(A - \lambda I)|_{\lambda=1} d = b$$

If *a* generates the correct vector *d*, we will order our eigenvectors as:

[*a d b c*]

but if *b* generates the correct vector, we can order it as:

[*a b d c*]

Jordan Cannonical Form

For more information about **Jordan Cannonical Form**, see:
Matrix Forms

If a matrix has a complete set of distinct eigenvectors, the transition matrix T can be defined as the matrix of those eigenvectors, and the resultant transformed matrix will be a diagonal matrix. However, if the eigenvectors are not unique, and there are a number of generalized eigenvectors associated with the matrix, the transition matrix T will consist of the ordered set of the regular eigenvectors and generalized eigenvectors. The regular eigenvectors that did not produce any generalized eigenvectors (if any) should be first in the order, followed by the eigenvectors that did produce generalized eigenvectors, and the generalized eigenvectors that they produced (in appropriate sequence).

Once the T matrix has been produced, the matrix can be transformed by it and it's inverse:

$$A = T^{-1}JT$$

The J matrix will be a **jordan block matrix**. The format of the jordan block matrix will be as follows:

$$J = \begin{bmatrix} D & 0 & \cdots & 0 \\ 0 & J_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J_n \end{bmatrix}$$

Where D is the diagonal block produced by the regular eigenvectors that are not associated with generalized eigenvectors (if any). The J_n blocks are standard jordan blocks with a size corresponding to the number of eigenvectors/generalized eigenvectors in each sequence. In each J_n block, the eigenvalue associated with the regular eigenvector of the sequence is on the main diagonal, and there are 1's in the super-diagonal.

System Response

Equivalence Transformations

If we have a non-singular n × n matrix P, we can define a transformed vector "x bar" as:

$$\bar{x} = Px$$

We can transform the entire state-space equation set as follows:

$$\bar{x}'(t) = \bar{A}\bar{x}(t) + \bar{B}\bar{u}(t)$$
$$\bar{y}(t) = \bar{C}\bar{x}(t) + \bar{D}u(t)$$

Where:

$\bar{A} = PAP^{-1}$
$\bar{B} = PB$
$\bar{C} = CP^{-1}$
$\bar{D} = D$

We call the matrix P the **equivalence transformation** between the two sets of equations.

It is important to note that the **eigenvalues** of the matrix A (which are of primary importance to the system) do not change under the equivalence transformation. The eigenvectors of A, and the eigenvectors of $\bar{A}$ are related by the matrix P.

MIMO Systems

Multi-Input, Multi-Output

Systems with more then one input and/or more then one output are known as **Multi-Input Multi-Output** systems, or they are frequently known by the abbreviation **MIMO**. This is in contrast to systems that have only a single input and a single output (SISO), like we have been discussing previously.

State-Space Representation

MIMO systems that are lumped and linear can be described easily with state-space equations. To represent multiple inputs we expand the input u(t) into a vector **u**(t) with the desired number of inputs. Likewise, to represent a system with multiple outputs, we expand y(t) into **y**(t), which is a vector of all the outputs. For this method to work, the outputs must be linearly dependant on the input vector and the state vector.

$$\mathbf{x}'(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

Let's say that we have 2 outputs, y1 and y2, and 2 inputs, u1 and u2. These are related in our system through the following system of differential equations:

$$y_1'' + a_1y_1' + a_0(y_1 + y_2) = u_1(t)$$

$$y_2' + a_2(y_2 - y_1) = u_2(t)$$

now, we can assign our state variables as such, and produce our first-order differential equations:

$$x_1 = y_1$$

$$x_4 = y_2$$

$$x_1' = y_1' = x_2$$

$$x_2' = -a_1x_2 - a_0(x_1 + x_4) + u_1(t)$$

$$x_4' = -a_2(x_4 - x_1) + u_2(t)$$

And finally we can assemble our state space equations:

$$x' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -a_0 & -a_1 & 0 & a_0 \\ 0 & 0 & 0 & 0 \\ a_2 & 0 & 0 & -a_2 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix}y_1\\y_2\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}x(t)$$

When we have multiple inputs or outputs, it is frequently common to use capital letters to denote vectors. For instance, we can say that Y is the vector of all outputs, and U is the vector of all inputs.

Transfer Function Matrix

If the system is LTI and Lumped, we can take the Laplace Transform of the state-space equations, as follows:

$$\mathcal{L}[\mathbf{x}'(t)] = \mathcal{L}[A\mathbf{x}(t)] + \mathcal{L}[B\mathbf{u}(t)]$$
$$\mathcal{L}[\mathbf{y}(t)] = \mathcal{L}[C\mathbf{x}(t)] + \mathcal{L}[D\mathbf{u}(t)]$$

Which gives us the result:

$$s\mathbf{X}'(s) - \mathbf{x}(0) = A\mathbf{X}(s) + B\mathbf{U}(s)$$
$$\mathbf{Y}(s) = C\mathbf{X}(s) + D\mathbf{U}(s)$$

Where x(0) is the initial conditions of the system state vector. If the system is relaxed, we can ignore this term, but for completeness we will continue the derivation with it.

We can separate out the variables in the state equation as follows:

$$s\mathbf{X}'(s) - A\mathbf{X}(s) = \mathbf{x}(0) + B\mathbf{U}(s)$$

Then factor out an **X**(s):

$$\mathbf{X}(s)[sI - A] = \mathbf{x}(0) + B\mathbf{U}(s)$$

And then we can multiply both sides by the inverse of [sI-A] to give us our state equation:

$$\mathbf{X}(s) = [sI - A]^{-1}\mathbf{x}(0) + [sI - A]^{-1}B\mathbf{U}(s)$$

Now, if we plug in this value for **X**(s) into our output equation, above, we get a more complicated equation:

$$\mathbf{Y}(s) = C([sI - A]^{-1}\mathbf{x}(0) + [sI - A]^{-1}B\mathbf{U}(s)) + D\mathbf{U}(s)$$

And we can distribute the matrix **C** to give us our answer:

$$\mathbf{Y}(s) = C[sI - A]^{-1}\mathbf{x}(0) + C[sI - A]^{-1}B\mathbf{U}(s) + D\mathbf{U}(s)$$

Now, if the system is relaxed, and therefore **x**(0) is 0, the first term of this equation becomes 0. In this case, we can factor out a **U**(s) from the remaining two terms:

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

We can make the following substitution to obtain the **Transfer Function Matrix**, or more simply, the **Transfer Matrix**, **H**(s):

$$C[sI - A]^{-1}B + D = \mathbf{H}(s)$$

[Transfer Matrix]

And rewrite our output equation in terms of the transfer matrix as follows:

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

[Transfer Matrix Description]

If Y(s) and X(s) are 1 × 1 vectors (a SISO system), then we have our external description:

$$Y(s) = H(s)X(s)$$

Now, since X(s) = **X**(s), and Y(s) = **Y**(s), then **H**(s) must be equal to H(s). These are simply two different ways to describe the same exact equation, the same exact system.

Dimensions

If our system has **q** inputs, and **r** outputs, our transfer function matrix will be an r × q matrix.

Relation to Transfer Function

For SISO systems, the Transfer Function matrix will reduce to the transfer function as would be obtained by taking the Laplace transform of the system response equation.

For MIMO systems, with n inputs and m outputs, the transfer function matrix will contain n × m transfer functions, where each entry is the transfer function relationship between each individual input, and each individual output.

Through this derivation of the transfuer function matrix, we have shown the equivalency between the the Laplace methods and the State-Space method for representing systems. Also, we have shown how the Laplace method can be generalized to account for MIMO systems. Through the rest of this book, we will use the Laplace and State Space methods interchangably, opting to use one or the other where appropriate.

Zero-State and Zero-Input

If we have our complete system response equation from above:

$$\mathbf{Y}(s) = C[sI - A]^{-1}\mathbf{x}(0) + (C[sI - A]^{-1}B + D)\mathbf{U}(s)$$

We can separate this into two separate parts:

- $C[sI - A]^{-1}\mathbf{x}(0)$ The **Zero-Input Response**.

- $(C[sI - A]^{-1}B + D)\mathbf{U}(s)$ The **Zero-State Response**.

These are named because if there is no input to the system (zero-input), then the output is the response of the system to the initial system state. If there is no state to the system, then the output is the response of the system to the system input. The complete response is the sum of the system with no input, and the input with no state.

Discrete MIMO Systems

In the discrete case, we end up with similar equations, except that the **x**(0) initial conditions term is preceeded by an additional *z* variable:

$$\mathbf{X}(z) = [zI - A]^{-1}z\mathbf{x}(0) + [zI - A]^{-1}B\mathbf{U}(z)$$
$$\mathbf{Y}(z) = C[zI - A]^{-1}z\mathbf{x}(0) + C[zI - A]^{-1}B\mathbf{U}(z) + D\mathbf{U}(z)$$

If **x**(0) is zero, that term drops out, and we can derive a Transfer Function Matrix in the z domain as well:

$$\mathbf{Y}(z) = (C[zI - A]^{-1}B + D)\mathbf{U}(z)$$

$$C[zI - A]^{-1}B + D = \mathbf{H}(z)$$

[Transfer Matrix]

$$\mathbf{Y}(z) = \mathbf{H}(z)\mathbf{U}(z)$$

[Transfer Matrix Description]

Pulse Response

For digital systems, it is frequently a good idea to write the **pulse response** equation, from the state-space equations:

$$x[k + 1] = Ax[k] + Bu[k]$$

$$y[k] = Cx[k] + Du[k]$$

We can combine these two equations into a single difference equation using the coefficient matrices A, B, C, and D. To do this, we find the ratio of the system output vector, Y, to the system input vector, U:

$$\frac{Y(z)}{U(z)} = H(z) = C(zI - A)^{-1}B + D$$

So the system response to a digital system can be derived from the pulse response equation by:

$$Y(z) = H(z)U(z)$$

And we can set U(z) to a step input through the following z transform:

$$u(t) \Leftrightarrow U(z) = \frac{z}{z-1}$$

Plugging this into our pulse response we get our step response:

$$Y(z) = (C(zI - A)^{-1}B + D)\left(\frac{z}{z-1}\right)$$

$$\mathbf{Y}(z) = \mathbf{H}(z)\left(\frac{z}{z-1}\right)$$

[Pulse Response]

System Realization

Realization

Realization is the process of taking a mathematical model of a system (either in the Laplace domain or the State-Space domain), and creating a physical system. Some systems are not realizable.

An important point to keep in mind is that the Laplace domain representation, and the state-space representations are equivalent, and both representations describe the same physical systems. We want, therefore, a way to convert between the two representations, because each one is well suited for particular methods of analysis.

The state-space representation, for instance, is preferable when it comes time to move the system design from the drawing board to a constructed physical device. For that reason, we call the process of converting a system from the Laplace representation to the state-space representation "realization".

Realization Conditions

- A transfer function G(s) is realizable if and only if the system can be described by a finite-dimensional state-space equation.
- **{A B C D}**, an ordered set of the 4 system matrices, is called a **realization** of the system G(s).
- A system G is realizable if and only if the transfer matrix **G**(s) is a proper rational matrix. In other words, every entry in the matrix **G**(s) (only 1 for SISO systems) is a rational polynomial, and if the degree of the denominator is higher or equal to the degree of the numerator.

We've already covered the method for realizing a SISO system, the remainder of this chapter will talk about the general method of realizing a MIMO system.

Realizing the Transfer Matrix

We can decompose a transfer matrix **G**(s) into a *strictly proper* transfer matrix:

$$\mathbf{G}(s) = \mathbf{G}(\infty) + \mathbf{G}_{sp}(s)$$

Where $G_{sp}(s)$ is a strictly proper transfer matrix. Also, we can use this to find the value of our D matrix:

$$D = \mathbf{G}(\infty)$$

We can define d(s) to be the lowest common denominator polynomial of all the entries in **G**(s):

$$d(s) = s^r + a_1 s^{r-1} + \cdots + a_{r-1}s + a_r$$

Remember, *q* is the number of inputs, *p* is the number of internal system states, and *r* is the number of outputs.

Then we can define $\mathbf{G}_{sp}$ as:

$$\mathbf{G}_{sp}(s) = \frac{1}{d(s)}N(s)$$

Where

$$N(s) = N_1 s^{r-1} + \cdots + N_{r-1} s + N_r$$

And the N_i are p × q constant matrices.

If we remember our method for converting a transfer function to a state-space equation, we can follow the same general method, except that the new matrix A will be a block matrix, where each block is the size of the transfer matrix:

$$A = \begin{bmatrix} -a_1 I_p & -a_2 I_p & \cdots & -a_{r-1} I_p & -a_r I_p \\ I_p & 0 & \cdots & 0 & 0 \\ 0 & I_p & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_p & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} I_p \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} I_p & 0 & 0 & \cdots & 0 \end{bmatrix}$$

System Representation

Systems can be represented graphically in a number of ways. Block diagrams and signal-flow diagrams are powerful tools that can be used to manipulate systems, and convert them easily into transfer functions or state-space equations. The chapters in this section will discuss how systems can be described visually, and will also discuss how systems can be interconnected with each other.

Gain

This page of the Control Systems book is a stub. You can help by expanding this section.

What is Gain?

Gain is a proportional value that shows the relationship between the magnitude of the input to the magnitude of the output signal at steady state. Many systems contain a method by which the gain can be altered, providing more or less "power" to the system. However, increasing gain or decreasing gain beyond a particular safety zone can cause the system to become unstable.

Consider the given second-order system:

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

We can include an arbitrary gain term, K in this system that will represent an amplification, or a power increase:

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

Example: Gain

Here are some good examples of arbitrary gain values being used in physical systems:

Volume Knob
: On your stereo there is a volume knob that controls the gain of your amplifier circuit. Higher levels of volume (turning the volume "up") corresponds to higher amplification of the sound signal.

Gas Pedal
: The gas pedal in your car is an example of gain. Pressing harder on the gas pedal causes the engine to receive more gas, and causes the engine to output higher RPMs.

Brightness Buttons
: Most computer monitors come with brightness buttons that control how bright the screen image is. More brightness causes more power to be outputed to the screen.

Responses to Gain

As the gain to a system increases, generally the rise-time decreases, the percent overshoot increases, and the settling time increases. Although, these relationships are not always the same. A **critically damped system**, for example, may decrease in rise time while not experiancing any effects of percent overshoot or settling time.

Gain and Stability

If the gain increases to a high enough extent, some systems can become unstable. We will examine this effect in the chapter on **Root Locus**.

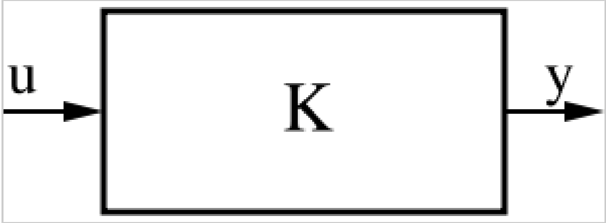
Conditional Stability

Systems that are stable for some gain values, and unstable for other values are called **conditionally stable** systems. The stability is conditional upon the the value of the gain, and often times the threshold where the system becomes unstable is important to find.

Block Diagrams

Block Diagram Representation

When designing or analyzing a system, often it is useful to model the system graphically. Block Diagrams are a useful and simple method for analyzing a system graphically. A "block" looks on paper exactly how it sounds:



A basic block diagram of system K, with input u(t) and output y(t).

If system K has a time-domain impulse response K(t), we can express y(t) as:

$$y(t) = u(t) * K(t)$$

Where the asterisk (*) denotes convolution. If system K has a Laplace-domain transfer function K(s), we show the relationship between the input and the output as:

$$Y(s) = U(s)K(s)$$

And if K is a state matrix, we can show the relationship between the input and output vectors as:

$$y = Kx$$

Systems in Series

When two or more systems are in series, they can be combined into a single representative system, with a transfer function that is the sum of the individual systems.

If we have two systems, F and G, we can put them in series with one another so that the output of system F is the input to system G. Now, we can analyze them depending on whether we are using our classical or modern methods.

Series Transfer Functions

If two or more systems are in series with one another, the total transfer function of the series is the product of all the individual system transfer functions.

Series State Space

If we have two systems in series (say system F and system G), where the output of F is the input to system G, we can write out the state-space equations for each individual system.

System 1:

$$\begin{aligned}x_F' &= A_F x_F + B_F u \\ y_F &= C_F x_F + D_F u\end{aligned}$$

System 2:

$$\begin{aligned}X_G' &= A_G x_G + B_G y_F \\ Y_G &= C_G x_G + D_G y_F\end{aligned}$$

And we can write substitute these equations together form the complete response of system H, that has input u, and output y_G:

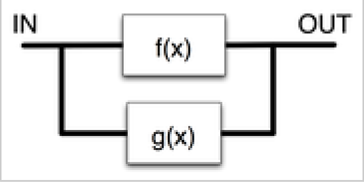
$$\begin{bmatrix}x_G \\ x_F\end{bmatrix} = \begin{bmatrix}A_G & B_G C_F \\ 0 & A_F\end{bmatrix}\begin{bmatrix}x_G \\ x_F\end{bmatrix} + \begin{bmatrix}B_G D_F \\ B_F\end{bmatrix} u$$

[Series state equation]

$$\begin{bmatrix}y_G \\ y_F\end{bmatrix} = \begin{bmatrix}C_G & D_G C_F \\ 0 & C_F\end{bmatrix}\begin{bmatrix}x_G \\ x_F\end{bmatrix} + \begin{bmatrix}D_G D_F \\ D_F\end{bmatrix} u$$

[Series output equation]

Systems in Parallel

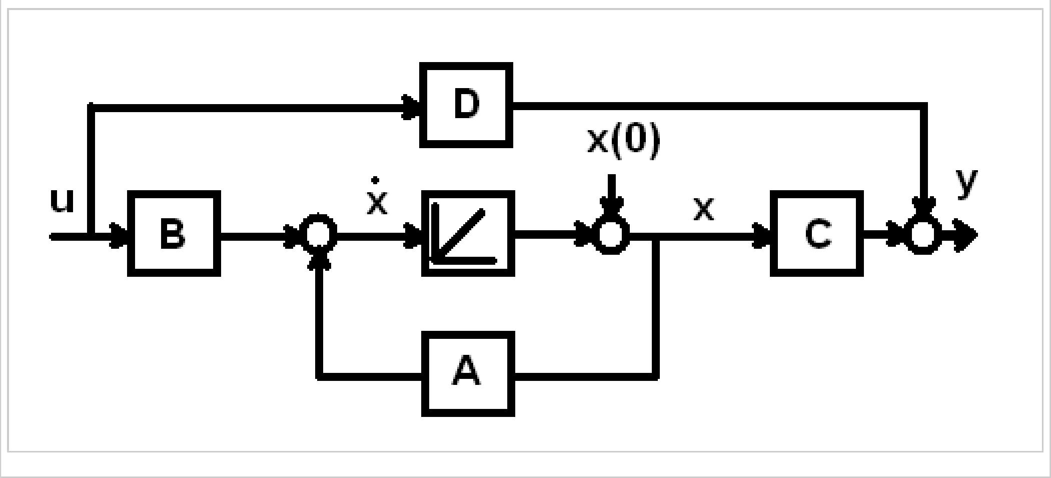


system f(x) in parallel with system g(x)

In practice, it is not common to see systems arranged in parallel. However, if you replace the node on the left with an adder, that combination is very common.

State Space Model

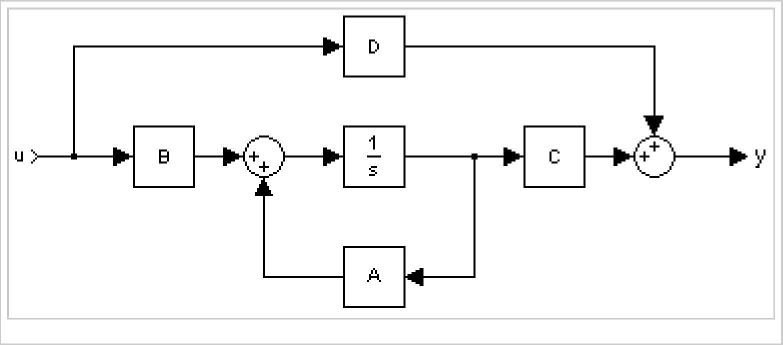
The state-space equations, with non-zero A, B, C, and D matrices conceptually model the following system:



In this image, the strange-looking block in the center is either an integrator, and can be represented in the transfer domain as:

$$\frac{1}{s} \text{ or } \frac{1}{z}$$

Depending on the time characteristics of the system. If we only consider continuous-time systems, we can replace the funny block in the center with an integrator:



In the Laplace Domain

The state space model of the above system, if A, B, C, and D are transfer functions A(s), B(s), C(s) and D(s) of the individual subsystems, and if U(s) and Y(s) represent a single input and output, can be written as follows:

$$\frac{Y(s)}{U(s)} = B(s)\left(\frac{1}{s - A(s)}\right)C(s) + D(s)U(s)$$

We will explain how we got this result, and how we deal with feedforward and feedback loop structures in the next chapter.

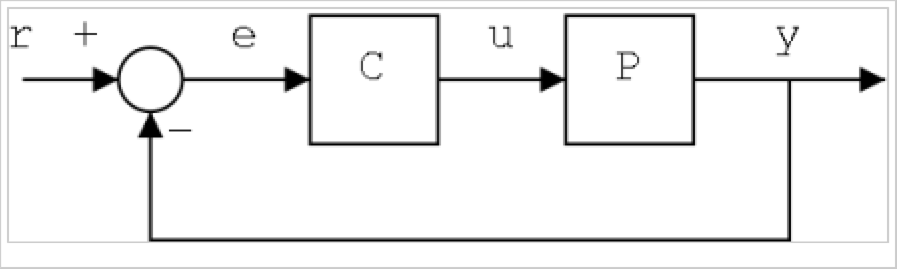
Adders and Multipliers

Some systems may have dedicated summation or multiplication devices, that automatically add or multiply the transfer functions of multiple systems together

Feedback Loops

A feedback loop is a common and powerful tool when designing a control system. Feedback loops take the system output into consideration, which enables them to perform better

Basic Feedback Structure



This is a basic feedback structure. Here, we are using the output value of the system to help us prepare the next output value. In this way, we can create systems that correct errors. Here we see a feedback loop with a value of one. we call this a **unity feedback**.

Here is a list of some relevant vocabulary, that will be used in the following sections:

Plant
: The term "Plant" is a carry-over term from chemical engineering to refer to the main system process. The plant is the preexisting system that does not (without the aid of a controller or a compensator) meet the given specifications. Plants are usually given "as is", and are not changeable. In the picture above, the plant is denoted with a P.

Controller
: A controller, or a "compensator" is an additional system that is added to the plant to control the operation of the plant. The system can have multiple compensators, and they can appear anywhere in the system: Before the pick-off node, after the adder, before or after the plant, and in the feedback loop. In the picture above, our compensator is denoted with a C.

Adder
: An adder is a symbol on a system diagram, (denoted above with parenthesis) that conceptually adds two or more input signals, and produces a single sum output signal.

Pick-off node
: A pickoff node is simply a fancy term for a split in a wire.

Forward Path
: The forward path in the feedback loop is the path after the adder, that travels through the plant and towards the system output.

Reverse Path
: The reverse path is the path after the pick-off node, that loops back to the beginning of the system. This is also known as the "feedback path".

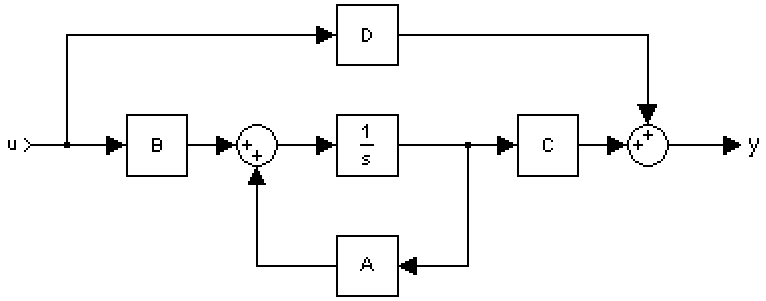
Unity feedback
: When the multiplicative value of the feedback path is 1.

Negative vs Positive Feedback

It turns out that negative feedback is almost always the most useful type of feedback. When we subtract the value of the output from the value of the input (our desired value), we get a value called the **error signal**. The error signal shows us how far off our output is from our desired input.

Example: State-Space Equation

In the previous chapter, we showed you this picture:



Now, we will derive the I/O relationship into the state-space equations. If we examine the inner-most feedback loop, we can see that the forward path has an integrator system, $\frac{1}{s}$, and the feedback loop has the matrix value A. If we take the transfer function only of this loop, we get:

$$T_{inner}(s) = \frac{\frac{1}{s}}{1 - \frac{1}{s}A} = \frac{1}{s - A}$$

Pre-multiplying by the factor B, and post-multiplying by C, we get the transfer function of the entire lower-half of the loop:

$$T_{lower}(s) = B\left(\frac{1}{s - A}\right)C$$

We can see that the upper path (D) and the lower-path T_{lower} are added together to produce the final result:

$$T_{total}(s) = B\left(\frac{1}{s - A}\right)C + D$$

Now, for an alternate method, we can assume that **x'** is the value of the inner-feedback loop, right before the integrator. This makes sense, since the integral of **x'** should be **x** (which we see from the diagram that it is. Solving for **x'**, with an input of **u**, we get:

$$x' = Ax + Bu$$

This is because the value coming from the feedback branch is equal to the value **x** times the feedback loop matrix A, and the value coming from the left of the adder is the input **u** times the matrix B.

If we keep things in terms of **x** and **u**, we can see that the system output is the sum of **u** times the feed-forward value D, and the value of **x** times the value C:

$$y = Cx + Du$$

These last two equations are precisely the state-space equations of our system.

Feedback Loop Transfer Function

We can solve for the output of the system by using a series of equations:

$$E(s) = X(s) - Y(s)$$

$$Y(s) = G(s)E(s)$$

and when we solve for Y(s) we get:

$$Y(s) = X(s)\frac{KGp(s)}{1 + Gp(s)}$$

[Feedback Transfer Function]

The reader is encouraged to use the above equations to derive the result by themselves.

The function E(s) is known as the **error signal**. The error signal is the difference between the system output (Y (s)), and the system input (X(s)). Notice that the error signal is now the direct input to the system G(s). X(s) is now called the **reference input**. The purpose of the negative feedback loop is to make the system output equal to the system input, by identifying large differences between X(s) and Y(s) and correcting for them. Here is a simple example of reference inputs and feedback systems:

There is an elevator in a certain building with 5 floors. Pressing button "1" will take you to the first floor, and pressing button "5" will take you to the fifth floor, etc. For reasons of simplicity, only one button can be pressed at a time.

Pressing a particular button is the reference input of the system. Pressing "1" gives the system a reference input of 1, pressing "2" gives the system a reference input of 2, etc. The elevator system then, tries to make the output (the physical floor location of the elevator) match the reference input (the button pressed in the elevator). The error signal, e(t), represents the difference between the reference input x(t), and the physical location of the elevator at time t, y(t).

Let's say that the elevator is on the first floor, and the button "5" is pressed at time t_0. The reference input then becomes a step function:

$$x(t) = 5u(t - t_0)$$

Where we are measuring in units of "floors". At time t_0, the error signal is:

$$e(t_0) = x(t_1) - y(t_1) = 5 - 1 = 4$$

Which means that the elevator needs to travel upwards 4 more floors. At time t_1, when the elevator is at the second floor, the error signal is:

$$e(t_1) = x(t_1) - y(t_1) = 5 - 2 = 3$$

Which means the elevator has 3 more floors to go. Finally, at time t_4, when the elevator reaches the top, the error signal is:

$$e(t_4) = x(t_4) - y(t_4) = 5 - 5 = 0$$

And when the error signal is zero, the elevator stops moving. In essence, we can define three cases:

- e(t) is positive: In this case, the elevator goes up one floor, and checks again.
- e(t) is zero: The elevator stops.
- e(t) is negative: The elevator goes down one floor, and checks again.

Open Loop vs Closed Loop

```
X(s)           +
-------| K |--->( )----->| Gp(s) |-------+----> Y(s)
                 ^ -                     |
  break here---> |                       |
                 +-------| Gb(s) |-------+
```

Let's say that we have the generalized system shown above. The top part, Gp(s) represents all the systems and all the controllers on the forward path. The bottom part, Gb(s) represents all the feedback processing elements of the system. The letter "K" in the beginning of the system is called the **Gain**. We will talk about the gain more in later chapters. We can define the **Closed-Loop Transfer Function** as follows:

$$H_{cl}(s) = \frac{KGp(s)}{1 + Gp(s)Gb(s)}$$ **[Closed-Loop Transfer Function]**

If we "open" the loop, and break the feedback node, we get the **Open-Loop Transfer Function**, as such:

$$H_{ol}(s) = KGp(s)Gb(s)$$ **[Open-Loop Transfer Function]**

We can redefine the closed-loop transfer function in terms of the open-loop transfer function:

$$H_{cl}(s) = \frac{KGp(s)}{1 + H_{ol}(s)}$$

These results are important, and they will be used without further explanation or derivation throughout the rest of the book.

Placement of a Controller

There are a number of different places where we could place an additional controller:

1. In front of the system, before the feedback loop.
2. Inside the feedback loop, in the forward path, before the plant.
3. In the forward path, after the plant.
4. In the feedback loop, in the reverse path.
5. After the feedback loop.

Each location has certain benefits and problems, and hopefully we will get a chance to talk about all of them.

Sampler Systems

Let's say that we introduce a sampler into our system:

```
X(s)                     / E*(s)
--------| K |--->( )---/  ---->| Gr(s) |--->| G(s) |--------+----> Y(s)
                  ^                                         |
                - |                                         |
                  +-----------------------------------------+
```

Notice that after the sampler, we must introduce a reconstruction circuit (described elsewhere) so that we may continue to keep the input, output, and plant in the laplace domain. Notice that we denote the reconstruction circuit with the symbol: Gr(s).

Now, Let's show the transfer function of this equation:

$$E^*(s) = \mathcal{L}^*[X(s) - Y(s)] = X^*(s) - Y^*(s)$$
$$Y(s) = E^*(s)G_r(s)G(s)$$

Now, this is going to get a little tricky, so follow along:

$$Y(s) = X^*(s)G_r(s)G(s) - Y^*(s)G_r(s)G(s)$$

And we convert into the star domain because each term on the right-hand side of this equation has a star-domain term:

$$Y^*(s) = X^*(s)G_r^*(s)G^*(s) - Y^*(s)G_r^*(s)G^*(s)$$

And next we can change variables into the Z-domain:

$$Y(z) = X(z)G_r(z)G(z) - Y(z)G_r(z)G(z)$$

And we can solve for Y(z):

$$Y(z) = \frac{X(z)G_r(z)G(z)}{1 + G_r(z)G(z)}$$

The preceeding was a particularly simple example. However, the reader is encouraged to solve for the transfer function for a system with a sampler (and it's associated reconstructor) in the following places:

1. Before the feedback system
2. In the forward path, after the plant
3. In the reverse path
4. After the feedback loop

Second-Order Systems

Damping Ratio

Natural Frequency

System Sensitivity

Signal Flow Diagrams

This page needs some pictures! if you have images of signal-flow graphs that you would be willing to upload/donate, it would be appreciated.

Signal Flow Diagrams

Signal Flow Diagrams are another method for visually representing a system. Signal Flow Diagrams are especially useful, because they allow for particular methods of analysis, such as **Mason's Gain Formula**.

Signal flow diagrams typically use curved lines to represent wires *and systems*, instead of using lines at right-angles, and boxes, respectively. Every curved line is considered to have a multiplier value, which can be a constant gain value, or an entire transfer function. Signals travel from one end of a line to the other, and lines that are placed in series with one another have their total multiplier values multiplied together (just like in block diagrams).

Signal flow diagrams help us to identify structures called "loops" in a system, which can be analyzed individually to determine the complete response of the system.

Mason's Gain Formula

Mason's rule is a rule for determining the gain of a system. Mason's rule can be used with block diagrams, but it is most commonly (and most easily) used with signal flow diagrams.

Forward Paths

A **forward path** is a path in the signal flow diagram that connects the input to the output without touching any single node or path more then once. A single system can have multiple forward paths.

Loops

A **loop** is a structure in a signal flow diagram that leads back to itself. A loop does not contain the beginning and ending points, and the end of the loop is the same node as the beginning of a loop.

Loops are said to touch if they share a node or a line in common.

The **Loop gain** is the total gain of the loop, as you travel from one point, around the loop, back to the starting point.

Delta Values

The Delta value of a system, denoted with a greek Δ is computed as follows:

$$\Delta = 1 - A + B - C + D - E...$$

Where:

- A is the sum of all individual loop gains
- B is the sum of the products of all the pairs of touching loops
- C is the sum of the products of all the sets of 3 touching loops
- D is the sum of the products of all the sets of 4 touching loops
- et cetera.

If the given system has no pairs of loops that touch, for instance, B and all additional letters after B will be zero.

Mason's Rule

If we have computed our delta values (above), we can then use **Mason's Gain Rule** to find the complete gain of the system:

$$M = \frac{y_{out}}{y_{in}} = \sum_{k=1}^{N} \frac{M_k \Delta_{\ k}}{\Delta}$$

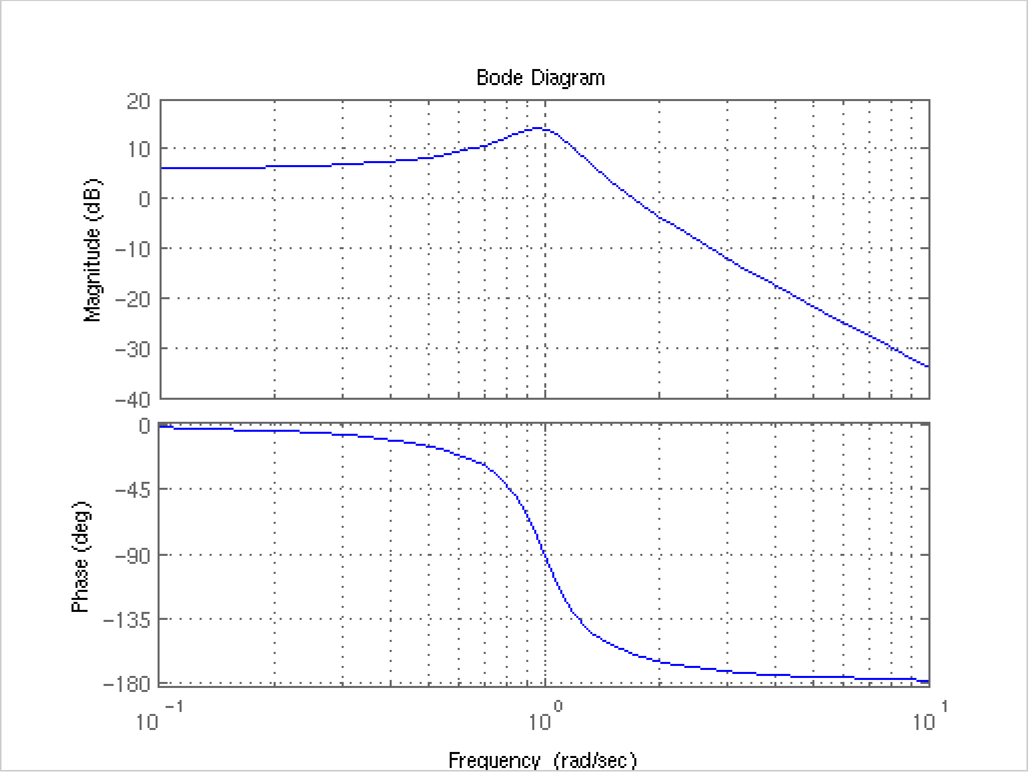
[Mason's Rule]

Where M is the total gain of the system, represented as the ratio of the output gain (y_{out}) to the input gain (y_{in}) of the system. M_k is the gain of the kth forward path, and Δ_k is the loop gain of the kth loop.

Bode Plots

Bode Plots

A Bode Plot is a useful tool that shows the gain and phase response of a given LTI system for different frequencies. Bode Plots are generally used with the Fourier Transform of a given system.



An example of a Bode magnitude and phase plot set. The Magnitude plot is typically on the top, and the Phase plot is typically on the bottom of the set.

The frequency of the bode plots are plotted against a logarithmic frequency axis. Every tickmark on the frequency axis represents a power of 10 times the previous value. For instance, on a standard Bode plot, the values of the markers go from (0.1, 1, 10, 100, 1000, ...) Because each tickmark is a power of 10, they are referred to as a **decade**. Notice that the "length" of a decade increases as you move to the right on the graph.

The bode Magnitude plot measures the system Input/Output ratio in special units called **decibels**. The Bode phase plot measures the phase shift in degrees (typically, but radians are also used).

Decibels

A **Decibel** is a ratio between two numbers on a logarithmic scale. A Decibel *is not itself a number, and cannot be treated as such in normal calculations*. To express a ratio between two numbers (A and B) as a decibel we apply the following formula:

$$Db = 20 \log \left(\frac{A}{B} \right)$$

Where Db is the decibel result.

Or, if we just want to take the decibels of a single number C, we could just as easily write:

$$Db = 20 \log(C)$$

Frequency Response Notations

If we have a system transfer function T(s), we can separate it into a numerator polynomial N(s) and a denominator polynomial D(s). We can write this as follows:

$$T(s) = \frac{N(s)}{D(s)}$$

To get the magnitude gain plot, we must first transition the transfer function into the frequency response by using the change of variables:

$$s = j\omega$$

From here, we can say that our frequency response is a composite of two parts, a real part R and an imaginary part X:

$$T(j\omega) = R(\omega) + jX(\omega)$$

We will use these forms below.

Straight-Line Approximations

The Bode magnitude and phase plots can be quickly and easily approximated by using a series of straight lines. These approximate graphs can be generated by following a few short, simple rules (listed below). Once the straight-line graph is determined, the actual Bode plot is a smooth curve that follows the straight lines, and travels through the **breakpoints**.

Break Points

If the frequency response is in pole-zero form:

$$T(j\omega) = \frac{\prod_n |j\omega + z_n|}{\prod_m |j\omega + p_m|}$$

We say that the values for all z_n and p_m are called **break points** of the Bode plot. These are the values where the Bode plots experiance the largest change in direction.

Break points are sometimes also called "break frequencies", "cuttoff points", or "corner points".

Bode Gain Plots

Bode Gain Plots, or **Bode Magnitude Plots** display the ratio of the system gain at each input frequency. It is important to note that a system's response may change depending on the frequency characteristics of the input. The Laplace transform does not account for frequency as a factor in determining system output, but the **Fourier Transform** does.

Bode Gain Calculations

The magnitude of the transfer function T is defined as:

$$|T(j\omega)| = \sqrt{R^2 + X^2}$$

However, it is frequently difficult to transition a function that is in "numerator/denominator" form to "real+imaginary" form. Luckily, our decibel calculation comes in handy. Let's say we have a frequency response defined as a fraction with numerator and denominator polynomials defined as:

$$T(j\omega) = \frac{\prod_n |j\omega + z_n|}{\prod_m |j\omega + p_m|}$$

If we convert both sides to decibels, the logarithms from the decibel calculations convert multiplication of the arguments into additions, and the divisions into subtractions:

$$Gain = \sum_n 20\log(j\omega + z_n) - \sum_m 20\log(j\omega + p_m)$$

And calculating out the gain of each term and adding them together will give the gain of the system at that frequency.

Bode Gain Approximations

The slope of a straight line on a Bode magnitude plot is measured in units of **dB/Decade**, because the units on the vertical axis are dB, and the units on the horizontal axis are decades.

The value ω = 0 is infinitely far to the left of the bode plot (because a logarithmic scale never reaches zero), so finding the value of the gain at &omega = 0 essentially sets that value to be the gain for the Bode plot from all the way on the left of the graph up till the first break point. The value of the slope of the line at ω = 0 is 0dB/Decade.

From each pole break point, the slope of the line decreases by 20dB/Decade. The line is straight until it reaches the next break point. From each zero break point the slope of the line increases by 20dB/Decade. Double, triple, or higher amounts of repeat poles and zeros affect the gain by multiplicative amounts. Here are some examples:

- 2 poles: -40dB/Decade
- 10 poles: -200dB/Decade
- 5 zeros: +100dB/Decade

Bode Phase Plots

Bode phase plots are plots of the phase shift to an input waveform dependant on the frequency characteristics of the system input. Again, the Laplace transform does not account for the phase shift characteristics of the system, but the Fourier Transform can. The phase of a complex function, in "real+imaginary" form is given as:

$$\angle T(j\omega) = \tan^{-1}\left(\frac{X}{R}\right)$$

Bode Proceedure

Given a frequency response in pole-zero form:

$$T(j\omega) = A\frac{\prod_n |j\omega + z_n|}{\prod_m |j\omega + p_m|}$$

Where A is a non-zero constant (can be negative or positive).

Here are the steps involved in sketching the approximate Bode magnitude plots:

Bode Magnitude Plots

Step 1
: Factor the transfer function into pole-zero form.

Step 2
: Find the frequency response from the Transfer function.

Step 3
: Use logarithms to separate the frequency response into a sum of decibel terms

Step 5
: Use $\omega = 0$ to find the starting magnitude.

Step 4
: The locations of every pole and every zero are called **break points**. At a zero breakpoint, the slope of the line increases by 20dB/Decade. At a pole, the slope of the line decreases by 20dB/Decade.

Step 6
: At a zero breakpoint, the value of the actual graph differs from the value of the staright-line graph by 3dB. A zero is +3dB over the straight line, and a pole is -3dB below the straight line.

Step 5
: Sketch the actual bode plot as a smooth-curve that follows the straight lines of the previous point, and travels through the breakpoints.

Here are the steps to drawing the Bode phase plots:

Bode Phase Plots

Step 1
: If A is positive, start your graph (with zero slope) at 0 degrees. If A is negative, start your graph with zero slope at 180 degrees (or -180 degrees, they are the same thing).

Step 2

For every zero, slope the line **up** at 45 degrees per decade when $\omega = \frac{z_n}{10}$ (1 decade before the break frequency). Multiple zeros means the slope is steeper.

Step 3

For every pole, slope the line **down** at 45 degrees per decade when $\omega = \frac{p_m}{10}$ (1 decade before the break frequency). Multiple poles means the slope is steeper.

Step 4

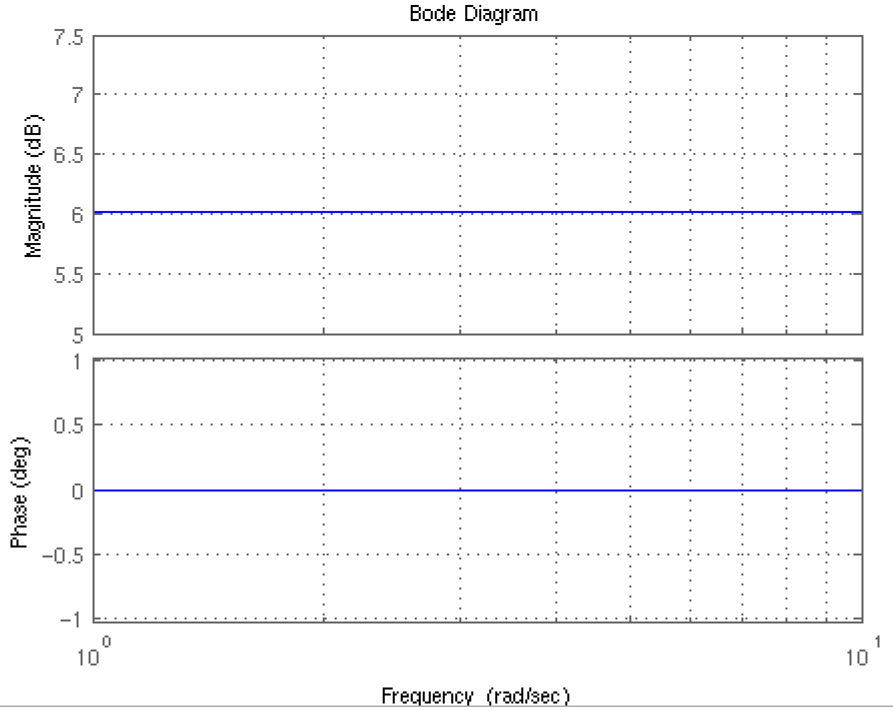
Flatten the slope again when the phase has changed by 90 degrees (for a zero) or -90 degrees (for a pole) (or larger values, for multiple poles or multiple zeros.

Examples

Example: Constant Gain

Draw the bode plot of an amplifier system, with a constant gain increase of 6dB.

Because the gain value is constant, and is not dependant on the frequency, we know that the value of the magnitude graph is constant at all places on the graph. There are no break points, so the slope of the graph never changes. We can draw the graph as a straight, horizontal line at 6dB:



Example: Integrator

Draw the bode plot of a perfect integrator system given by the transfer function:

$$T(s) = \frac{2}{s}$$

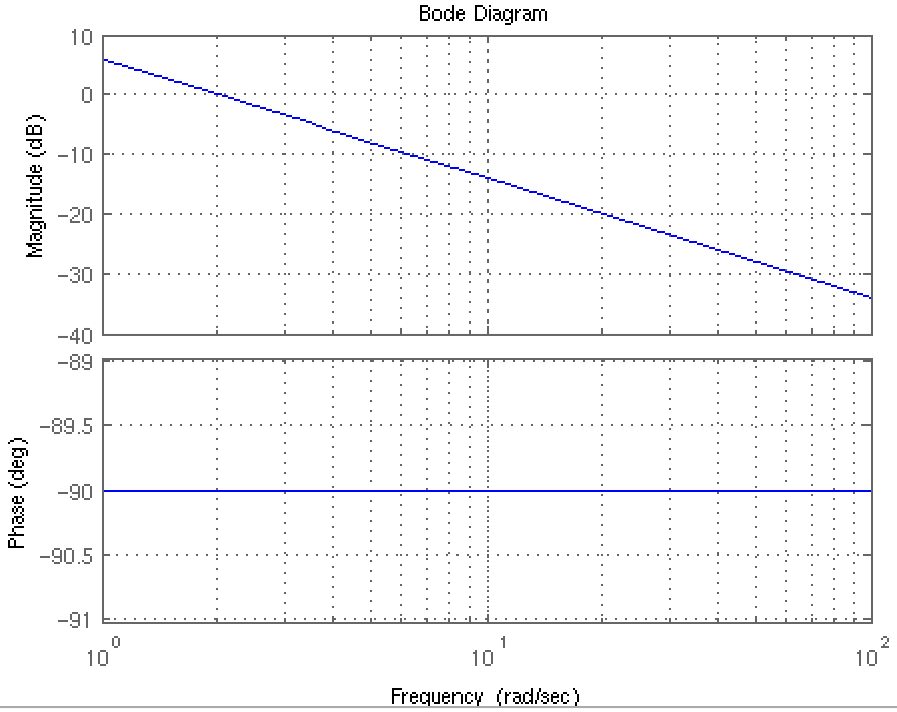
First, we find the frequency response of the system, by a change of variables:

$$T(j\omega) = \frac{2}{j\omega}$$

Then we convert our magnitude into logarithms:

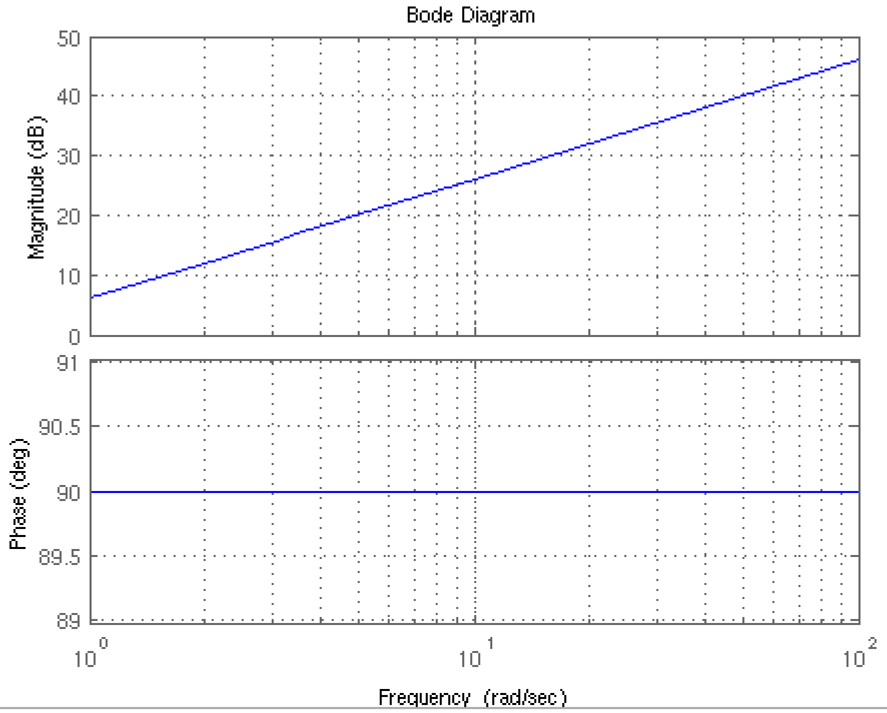
$$Gain = 20\log(2) - 20\log(j\omega)$$

Notice that the location of the break point for the pole is located at ω → 0, which is all the way to the left of the graph. Also, we notice that inserting 0 in for ω gives us an undefined value (which approaches negative infinity, as the limit). We know, because there is a single pole at zero, that the graph to the right of zero (which is everywhere) has a slope of -20dB/Decade. We can determine from our magnitude caluculation by plugging in ω → 1 that the second term drops out, and the magnitude at that point is 3dB. We now have the slope of the line, and a point that it intersects, and we can draw the graph:



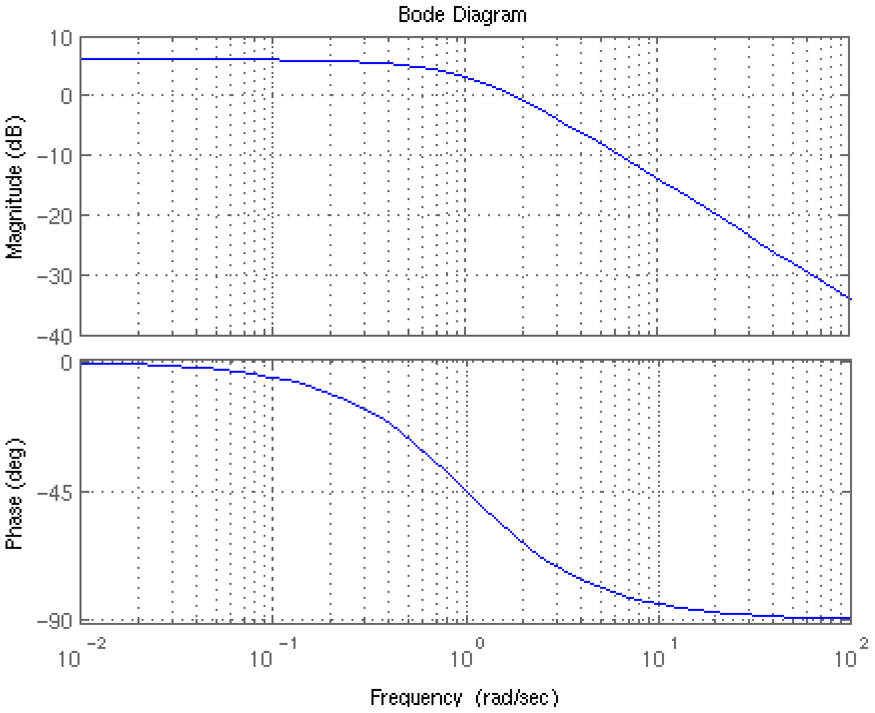
Example: Differentiator

$$T(j\omega) = 2j\omega$$



Example: 1 Break Point

$$T(j\omega) = \frac{2}{s-1}$$



Further Reading

- Circuit Theory/Bode Plots
- Wikipedia:Bode plots

Nichols Charts

This page of the Control Systems book is a stub. You can help by expanding this section.

Nichols Charts

This page will talk about the use of Nichols charts to analyze frequency-domain characteristics of control systems.

Stability

System stability is an important topic, because unstable systems may not perform correctly, and may actually be harmful to people. There are a number of different methods and tools that can be used to determine system stability, depending on whether you are in the state-space, or the complex domain.

Stability

BIBO Stability

When a system becomes unstable, the output of the system approaches infinity (or negative infinity), which often poses a security problem for people in the immediate vicinity. Also, systems which become unstable often incur a certain amount of physical damage, which can become costly. This chapter will talk about system stability, what it is, and why it matters.

A system is defined to be **BIBO Stable** if every bounded input to the system results in a bounded output. This means that so long as we don't input infinity to our system, we won't get infinity output.

Determining BIBO Stability

We can prove mathematically that a system f is BIBO stable if an arbitrary input x is bounded by two finite but large arbitrary constants M and -M:

$$-M < x \le M$$

We apply the input x, and the arbitrary boundries M and -M to the system to produce three outputs:

$$y_x = f(x)$$
$$y_M = f(M)$$
$$y_{-M} = f(-M)$$

Now, all three outputs should be finite for all possible values of M and x, and they should satisfy the following relationship:

$$y_{-M} \le y_x \le y_M$$

If this condition is satisfied, then the system is BIBO stable.

Example

Consider the system:

$$h(t) = \frac{2}{t}$$

We can apply our test, selecting an arbitrarily large finite constant M, and an arbitrary input x such that -M < x < M.

As M approaches infinity (but does not reach infinity), we can show that:

$$y_{-M} = \lim_{M \to \infty} \frac{2}{M} = 0$$

And:

$$y_M = \lim M \to \infty \frac{2}{M} \approx \infty$$

So now, we can write out our inequality:

$$y_{-M} \le y_x \le y_M$$
$$0 \le x < \infty$$

And this inequality should be satisfied for all possible values of x. However, we can see that when x is zero, we have the following:

$$y_x = \lim_{x \to 0} \frac{2}{x} = \infty$$

Which means that x is between -M and M, but the value y_x is not between y_{-M} and y_M. Therefore, this system is not stable.

Poles and Stability

When the poles of the closed-loop transfer function of a given system are located in the right-half of the S-plain (RHP), the system becomes unstable. When the poles of the system are located in the left-half plane (LHP), the system is shown to be stable. A number of tests deal with this particular facet of stability: The **Routh-Hurwitz Criteria**, the **Root-Locus**, and the **Nyquist Stability Criteria** all test whether there are poles of the transfer function in the RHP. We will learn about all these tests in the upcoming chapters.

Transfer Functions Revisited

Let us remember our generalized feedback-loop transfer function, with a gain element of K, a forward path Gp(s), and a feedback of Gb(s). We write the transfer function for this system as:

$$H_{cl}(s) = \frac{KGp(s)}{1 + H_{ol}(s)}$$

Where H_{cl} is the closed-loop transfer function, and H_{ol} is the open-loop transfer function. Again, we define the open-loop transfer function as the product of the forward path and the feedback elements, as such:

$$H_{ol}(s) = KGp(s)Gb(s)$$

Now, we can define F(s) to be the **characteristic equation**. F(s) is simply the denominator of the closed-loop transfer function, and can be defined as such:

$$F(s) = 1 + H_{ol} = D(s)$$

[Characteristic Equation]

We can say conclusively that the roots of the characteristic equation are the poles of the transfer function. Now, we know a few simple facts:

1. The locations of the poles of the closed-loop transfer function determine if the system is stable or not
2. The zeros of the characteristic equation are the poles of the closed-loop transfer function.
3. The characteristic equation is always a simpler equation then the the closed-loop transfer function.

These functions combined show us that we can focus our attention on the characteristic equation, and find the roots of that equation.

State-Space and Stability

Determining whether a state-space system is stable is a little bit more tricky, but there are some tests that we can perform to show whether a system is stable. There are methods that use the eigenvalues of the system matrix to show whether the system is stable, and then there is the **Lyapunov Method** that determines whether a system matrix is stable or not. We will learn about these methods in the upcoming chapters.

Marginal Stablity

When the poles of the system in the complex S-Domain exist on the complex frequency axis (the horizontal axis), the system exhibits oscillatory characteristics, and is said to be marginally stable. A marginally stable system may become unstable under certain circumstances, and may be perfectly stable under other circumstances. It is impossible to tell by inspection whether a marginally stable system will become unstable or not.

Routh-Hurwitz Criterion

Stability Criteria

The Routh-Hurwitz stability criterion is a necessary and sufficient criterion to prove the stability of an LTI system.

Necessary

Conditions that are necessary must be satisfied for a system to be stable, but conditions that satisfy these conditions might not all be stable. Necessary conditions may return "false positives", but will never return "false negatives".

Sufficient

Sufficient conditions are conditions that if met show the system to be definatively stable. Sufficient conditions may not be necessary, and they may return false negatives.

The Routh-Hurtwitz criteria is both necessary and sufficient: A system must pass the RH test, and once it passes the test, it is definately stable.

Routh-Hurwitz Criteria

The Routh-Hurwitz criteria is comprised of three separate tests that must be satisfied. If any test fails, the system is not stable. Also, if any single test fails, any further tests need not be performed. For this reason, the tests are arranged in order from the easiest to determine to the hardest to determine.

The Routh Hurwitz test is performed on the denominator of the transfer function, the **characteristic equation**. For instance, in a closed-loop transfer function with G(s) in the forward path, and H(s) in the feedback loop, we have:

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

If we simplify this equation, we will have an equation with a numerator N(s), and a denominator D(s):

$$T(s) = \frac{N(s)}{D(s)}$$

The Routh-Hurwitz criteria will focus on the denominator polynomial D(s).

Routh-Hurwitz Tests

Here are the three tests of the Routh-Hurwitz Criteria. For convenience, we will use N as the order of the polynomial (the value of the highest exponent of s in D(s)). The equation D(s) can be represented generally as follows:

$$D(s) = a_0 + a_1 s + a_2 s^2 + \cdots + a_N s^N$$

Rule 1

All the coefficients a_i must be present (non-zero)

Rule 2

All the coefficients a_i must be positive

Rule 3

If **Rule 1** and **Rule 2** are both satisfied, then form a **Routh array** from the coefficients a_i. There is one pole in the right-hand s-plane for ever sign change of the members in the first column of the Routh array (any sign changes, therefore, mean the system is unstable).

We will explain the Routh array below.

The Routh Array

The Routh array is formed by taking all the coefficients a_i of D(s), and staggering them in array form. The final columns for each row should contain zeros:

$$\begin{array}{c|cccc|} s^N & a_N & a_{N-2} & \cdots & 0 \\ s^{N-1} & a_{N-1} & a_{N-3} & \cdots & 0 \end{array}$$

Therefore, if N is odd, the top row will be all the odd coefficients. If N is even, the top row will be all the even coefficients. We can fill in the remainder of the Routh Array as follows:

$$\begin{array}{c|cccc|} s^N & a_N & a_{N-2} & \cdots & 0 \\ s^{N-1} & a_{N-1} & a_{N-3} & \cdots & 0 \\ & b_{N-1} & b_{N-3} & \cdots & \\ & c_{N-1} & c_{N-3} & \cdots & \\ s^0 & \cdots & & & \end{array}$$

Now, we can define all our b, c, and other coefficients, until we reach row s^0. To fill them in, we use the following formulae:

$$b_{N-1} = \frac{1}{a_{N-1}} \begin{vmatrix} a_N & a_{N-2} \\ a_{N-1} & a_{N-3} \end{vmatrix}$$

And

$$b_{N-3} = \frac{1}{a_{N-1}} \begin{vmatrix} a_N & a_{N-4} \\ a_{N-1} & a_{N-5} \end{vmatrix}$$

For each row that we are computing, we call the left-most element in the row directly above it the **pivot element**. For instance, in row b, the pivot element is a_{N-1}, and in row c, the pivot element is b_{N-1} and so on and so forth until we reach the bottom of the array.

To obtain any element, we take the determinant of of the following matrix, and divide by the pivot element:

$$\begin{vmatrix} k & l \\ m & n \end{vmatrix}$$

Where:

- **k** is the left-most element two rows above the current row.
- **l** is the pivot element.
- **m** is the element two rows up, and one column to the left of the current element.
- **n** is the element one row up, and one column to the left of the current element.

In terms of **k l m n**, our equation is:

$$v = \frac{(lm) - (kn)}{l}$$

Example: Calculating C_{N-3}

To calculate the value C_{N-3}, we must determine the values for **k l m** and **n**:

- **k** is the left-most element two rows up: a_{N-1}
- **l** the pivot element, is the left-most element one row up: b_{N-1}
- **m** is the element from one-column to the right, and up two rows: a_{N-5}
- **n** is the element one column right, and one row up: b_{N-5}

Plugging this into our equation gives us the formula for C_{N-3}:

$$\frac{1}{b_{N-1}}\begin{vmatrix} a_{N-1} & b_{N-1} \\ a_{N-5} & b_{N-5} \end{vmatrix} = \frac{b_{N-1}a_{N-5} - a_{N-1}b_{N-5}}{b_{N-1}}$$

Example: Stable Third Order System

We are given a system with the following characteristic equation:

$$D(s) = s^3 + 2s^2 + 4s + 3$$

Using the first two requirements, we see that all the coefficients are non-zero, and all of the coefficients are positive. We will proceed then to construct the Routh-Array:

$$\begin{array}{l|ccc|}
s^3 & 1 & 4 & 0 \\
s^2 & 2 & 3 & 0 \\
s^1 & b_{N-1} & b_{N-3} & 0 \\
s^0 & c_{N-1} & c_{N-3} & 0
\end{array}$$

And we can calculate out all the coefficients:

$$b_{N-1} = \frac{(2)(4) - (1)(3)}{2} = \frac{5}{2}$$
$$b_{N-3} = \frac{(2)(0) - (0)(1)}{2} = 0$$
$$c_{N-1} = \frac{(3)\left(\frac{5}{2}\right) - (2)(0)}{\frac{5}{2}} = 3$$
$$c_{N-3} = \frac{(2)(0) - \left(\frac{5}{3}\right)(0)}{\frac{5}{2}} = 0$$

And filling these values into our Routh Array, we can determine whether the system is stable:

$$\begin{array}{l|ccc|}
s^3 & 1 & 4 & 0 \\
s^2 & 2 & 3 & 0 \\
s^1 & \frac{5}{2} & 0 & 0 \\
s^0 & 3 & 0 & 0
\end{array}$$

From this array, we can clearly see that all of the signs of the first column are positive, there are no sign changes, and therefore there are no poles of the characteristic equation in the RHP.

Special Case: Row of All Zeros

If, while calculating our Routh-Hurwitz, we obtain a row of all zeros, we do not stop, but can actually learn more information about our system. If we obtain a row of all zeros, we can replace the zeros with a value ε, that we define as being an infinitely small positive number. We can use the value of epsilon in our equations, and when we are done constructing the Routh Array, we can take the limit as epsilon approaches 0 to determine the final format ouf our Routh array.

If we have a row of all zeros, the row directly above it is known as the **Auxiliary Polynomial**, and can be very helpful. The roots of the auxiliary polynomial give us the precise locations of complex conjugate roots that lie on the jω axis. However, one important point to notice is that if there are repeated roots on the jω axis, the system is actually **unstable**. Therefore, we must use the auxiliary polynomial to determine whether the roots are repeated or not.

Special Case: Zero in the First Column

In this special case, there is a zero in the first column of the Routh Array, but the other elements of that row are non-zero. Like the above case, we can replace the zero with a small variable epsilon (ε) and use that variable to continue our calculations. After we have constructed the entire array, we can take the limit as epsilon approaches zero to get our final values.

RH in Digital Systems

Because of the differences in the Z and S domains, the Routh-Hurwitz criteria can not be used directly with digital systems. This is because digital systems and continuous-time systems have different regions of stability. However, there are some methods that we can use to analyze the stability of digital systems. Our first option (and arguably not a very good option) is to convert the digital system into a continuous-time representation using the **bilinear transform**. The bilinear transform converts an equation in the Z domain into an equation in the W domain, that has properties similar to the S domain. Another possibility is to use **Jury's Stability Test**. Jury's test is a procedure similar to the RH test, except it has been modified to analyze digital systems in the Z domain directly.

Bilinear Transform

One common, but time-consuming, method of analyzing the stability of a digital system in the z-domain is to use the bilinear transform to convert the transfer function from the z-domain to the w-domain. The w-domain is similar to the s-domain in the following ways:

- Poles in the right-half plane are unstable
- Poles in the left-half plane are stable
- Poles on the imaginary axis are partially stable

The w-domain is warped with respect to the s domain, however, and except for the relative position of poles to the imaginary axis, they are not in the same places as they would be in the s-domain.

Remember, however, that the Routh-Hurwitz criterion can tell us whether a pole is unstable or not, and nothing else. Therefore, it doesn't matter where exactly the pole is, so long as it is in the correct half-plane. Since we know that stable poles are in the left-half of the w-plane and the s-plane, and that unstable poles are on the right-hand side of both planes, we can use the Routh-Hurwitz test on functions in the w domain exactly like we can use it on functions in the s-domain.

Other Mappings

There are other methods for mapping an equation in the Z domain into an equation in the S domain, or a similar domain. We will discuss these different methods in the **Appendix**.

Jury's Test

Jury's test is a test that is similar to the Routh-Hurwitz criterion, except that it can be used to analyze the stability of an LTI digital system in the Z domain. To use Jury's test to determine if a digital system is stable, we must check our z-domain characteristic equation against a number of specific rules and requirements. If the function fails any requirement, it is not stable. If the function passes all the requirements, it is stable. Jury's test is a necessary and sufficient test for stability in digital systems.

Again, we call D(z) the **characteristic polynomial** of the system. It is the denominator polynomial of the Z-domain transfer function. Jury's test will focus exclusively on the Characteristic polynomial. To perform Jury's test, we must perform a number of smaller tests on the system. If the system fails any test, it is unstable.

Jury Tests

Given a characteristic equation in the form:

$$D(z) = a_0 + a_1 z + a_2 z^2 + \cdots + a_N Z^N$$

The following tests determine whether this system has any poles outside the unit circle (the instability region). These tests will use the value N as being the degree of the characteristic polynomial.

The system must pass all of these tests to be considered stable. If the system fails any test, you may stop immediately: you do not need to try any further tests.

Rule 1
: If z is 1, the system output must be positive:

$$D(1) > 0$$

Rule 2
: If z is -1, then the following relationship must hold:

$$(-1)^N D(-1) > 0$$

Rule 3
: The absolute value of the constant term (a_0) must be less then the value of the highest coefficient (a_N):

$$|a_0| < a_N$$

If **Rule 1** **Rule 2** and **Rule 3** are satisified, construct the **Jury Array** (discussed below).

Rule 4
: Once the Jury Array has been formed, all the following relationships must be satisifed until the end of the array:

$$|b_0| > |b_{N-1}|$$
$$|c_0| > |b_{N-2}|$$
$$|d_0| > |b_{N-3}|$$

And so on until the last row of the array. If all these conditions are satisifed, the system is stable.

While you are constructing the Jury Array, you can be making the tests of **Rule 4**. If the Array fails **Rule 4** at any point, you can stop calculating the array: your system is unstable. We will discuss the construction of the Jury Array below.

The Jury Array

The Jury Array is constructed by first writing out a row of coefficients, and then writing out another row with the same coefficients in reverse order. For instance, if your polynomial is a third order system, we can write the First two lines of the Jury Array as follows:

z^0	z^1	z^2	z^3
a_0	a_1	a_2	a_3
a_3	a_2	a_1	a_0

Now, once we have the first row of our coefficients written out, we add another row of coefficients (we will use **b** for this row, and **c** for the next row, as per our previous convention), and we will calculate the values of the lower rows from the values of the upper rows. Each new row that we add will have one fewer coefficient then the row before it:

a_0	a_1	a_2	a_3
a_3	a_2	a_1	a_0
b_0	b_1	b_2	
b_2	b_1	b_0	
c_0	c_1		
c_1	c_0		

Once we get to a row with 2 members, we can stop constructing the array.

To calculate the values of the odd-number rows, we can use the following formulae. The even number rows are equal to the previous row in reverse order. We will use k as an arbitrary subscript value. These formulae are reusable for all elements in the array:

$$b_k = \begin{vmatrix} a_0 & a_{N-k} \\ a_N & a_k \end{vmatrix}$$

$$c_k = \begin{vmatrix} b_0 & b_{N-1-k} \\ b_{N-1} & b_k \end{vmatrix}$$

$$d_k = \begin{vmatrix} c_0 & c_{N-2-k} \\ c_{N-2} & c_k \end{vmatrix}$$

This pattern can be carried on to all lower rows of the array, if needed.

Example: Calculating e_5

Give the equation for member e_5 of the jury array (assuming the original polynomial is sufficiently large to require an e_5 member).

Going off the pattern we set above, we can have this equation for a member e:

$$e_k = \begin{vmatrix} d_0 & d_{N-R-k} \\ d_{N-R} & d_k \end{vmatrix}$$

Where we are using R as the subtractive element from the above equations. Since row c had R → 1, and row d had R → 2, we can follow the pattern and for row e set R → 3. Plugging this value of R into our equation above gives us:

$$e_k = \begin{vmatrix} d_0 & d_{N-3-k} \\ d_{N-3} & d_k \end{vmatrix}$$

And since we want e_5 we know that k is 5, so we can substitute that into the equation:

$$e_5 = \begin{vmatrix} d_0 & d_{N-3-5} \\ d_{N-3} & d_5 \end{vmatrix} = \begin{vmatrix} d_0 & d_{N-8} \\ d_{N-3} & d_5 \end{vmatrix}$$

When we take the determinant, we get the following equation:

$$e_5 = d_{N-8}d_{N-3} - d_0d_5$$

Further Reading

We will discuss the bilinear transform, and other methods to convert between the Laplace domain and the Z domain in the appendix:

- Z Transform Mappings

Root Locus

The Problem

Consider a system like a radio. The radio has a "volume" knob, that controls the amount of gain of the system. High volume means more power going to the speakers, low volume means less power to the speakers. As the volume value increases, the poles of the transfer function of the radio change, and they might potentially become unstable. We would like to find out *if* the radio becomes unstable, and if so, we would lke to find out what values of the volume cause it to become unstable. Our current methods would require us to plug in each new value for the volume (gain, "K"), and solve the open-loop transfer function for the roots. This process can be a long one. Luckily, there is a method called the **root-locus** method, that allows us to graph the locations of all the poles of the system for all values of gain, K.

Root-Locus

As we change gain, we notice that the system poles and zeros actually move around in the S-plane. This fact can make life particularly difficult, when we need to solve higher-order equations repeatedly, for each new gain value. The solution to this problem is a technique known as **Root-Locus** graphs. Root-Locus allows you to graph the locations of the poles and zeros *for every value of gain*, by following several simple rules.

Let's say we have a closed-loop transfer function for a particular system:

$$\frac{N(s)}{D(s)} = \frac{KG(s)}{1 + KG(s)H(s)}$$

Where N is the numerator polynomial and D is the denominator polynomial of the transfer functions, respectively. Now, we know that to find the roots of the equation, we must set the denominator to 0, and solve the characteristic equation. In otherwords, the locations of the poles of a specific equation must satisfy the following relationship:

$$D(s) = 1 + KG(s)H(s) = 0$$

from this same equation, we can manipulate the equation as such:

$$1 + KG(s)H(s) = 0$$

$$KG(s)H(s) = -1$$

And finally by converting to polar coordinates:

$$\angle KG(s)H(s) = 180^\circ$$

Now we have 2 equations that govern the locations of the poles of a system for all gain values:

$$1 + KG(s)H(s) = 0$$

[The Magnitude Equation]

$$\angle KG(s)H(s) = 180^\circ$$

[The Angle Equation]

Digital Systems

The same basic method can be used for considering digital systems in the Z-domain:

$$\frac{N(z)}{D(z)} = \frac{KG(z)}{1 + K\overline{GH}(z)}$$

Where N is the numerator polynomial in z, D is the denominator polynomial in z, and $\overline{GH}(z)$ is the open-loop transfer function of the system, in the Z domain.

The denominator D(z), by the definition of the characteristic equation is equal to:

$$D(z) = 1 + K\overline{GH}(z) = 0$$

We can manipulate this as follows:

$$1 + K\overline{GH}(z) = 0$$
$$K\overline{GH}(z) = -1$$

We can now convert this to polar coordinates, and take the angle of the polynomial:

$$\angle K\overline{GH}(z) = 180^\circ$$

We are now left with two important equations:

$$1 + K\overline{GH}(z) = 0$$

[The Magnitude Equation]

$$\angle K\overline{GH}(z) = 180^\circ$$

[The Angle Equation]

If you will compare the two, the Z-domain equations are nearly identical to the S-domain equations, and act exactly the same. For the remainder of the chapter, we will only consider the S-domain equations, with the understanding that digital systems operate in nearly the same manner.

The Root-Locus Procedure

In the transform domain (see note at right), when the gain is small

Note:

the poles start at the poles of the open-loop transfer function. When gain becomes infinity, the poles move to overlap the zeros of the system. This means that on a root-locus graph, all the poles move towards a zero. Only one pole may move towards one zero, and this means that there must be the same number of poles as zeros.

In this section, the rules for the S-Plain and the Z-plain are the same, so we won't refer to the differences between them.

If there are fewer zeros then poles in the transfer function, there are a number of implicit zeros located at infinity, that the poles will approach.

First thing, we need to convert the magnitude equation into a slightly more convenient form:

$$KG(s)H(s) + 1 = 0 \to G(s)H(s) = \frac{-1}{K}$$

Now, we can assume that G(s)H(s) is a fraction of some sort, with a numerator and a denominator that are both polynomials. We can express this equation using arbitrary functions a(s) and b(s), as such:

$$\frac{a(s)}{b(s)} = \frac{-1}{K}$$

Note:
We generally use capital letters for functions in the frequency domain, but **a (s)** and **b(s)** are unimportant enough to be lower-case.

We will refer to these functions a(s) and b(s) later in the procedure.

We can start drawing the root-locus by first placing the roots of b(s) on the graph with an 'X'. Next, we place the roots of a(s) on the graph, and mark them with an 'O'.

poles are marked on the graph with an 'X' and zeros are marked with an 'O' by common convention. These letters have no particular meaning

Next, we examine the real-axis. starting from the left-hand side of the graph and traveling to the right, we draw a root-locus line on the real-axis at every point to the left of an odd number of poles on the real-axis. This may sound tricky at first, but it becomes easier with practice.

double poles or double zeros count as two.

Now, a root-locus line starts at every pole. Therefore, any place that two poles appear to be connected by a root locus line on the real-axis, the two poles actually move towards each other, and then they "break away", and move off the axis. The point where the poles break off the axis is called the **breakaway point**. From here, the root locus lines travel towards the nearest zero.

It is important to note that the s-plane is symmetrical about the real axis, so whatever is drawn on the top-half of the S-plane, must be drawn in mirror-image on the bottom-half plane.

Once a pole breaks away from the real axis, they can either travel out towards infinity (to meet an implict zero), or they can travel to meet an explict zero, or they can re-join the real-axis to meet a zero that is located on the real-axis. If a pole is traveling towards infinity, it always follows an asymptote. The number of asymptotes is equal to the number of implict zeros at infinity.

Root Locus Rules

Here is the complete set of rules for drawing the root-locus graph. We will use p and z to denote the number of poles and the number of zeros of the open-loop transfer function, respectively. We will use P_i and Z_i to denote the location of the *i*th pole and the *i*th zero, respectively. Likewise, we will use ψ_i and ρ_i to denote the angle from a given point to the *i*th pole and zero, respectively. All angles are given in radians (π denotes π radians).

There are 11 rules that, if followed correctly, will allow you to create a correct root-locus graph.

Rule 1
: There is one branch of the root-locus for every root of b(s).

Rule 2
: The roots of b(s) are the **poles** of the open-loop transfer function. Mark the roots of b(s) on the graph with an X.

Rule 3
: The roots of a(s) are the **zeros** of the open-loop transfer function. Mark the roots of b(s) on the graph with an O. There should be a number of O's less then or equal to the number of X's. There is a number of zeros p - z located at infinity. These zeros at infinity are called "implicit zeros". All branches of the root-locus will move from a pole to a zero (some branches, therefore, may travel towards infinity).

Rule 4
: A point on the real axis is a part of the root-locus if it is to the left of an odd number of poles or zeros.

Rule 5
: The gain at any point on the root locus can be determined by the inverse of the absolute value of the magnitude equation.

$$\left|\frac{b(s)}{a(s)}\right| = |K|$$

Rule 6
: The root-locus diagram is symmetric about the real-axis. All complex roots are conjugates.

Rule 7
: Two roots that meet on the real-axis will break away from the axis at certain break-away points. If we set s → σ (no imaginary part), we can use the following equation:

$$K = -\frac{a(\sigma)}{b(\sigma)}$$

And differentiate to find the local maximum:

$$\frac{dK}{d\sigma} = \frac{d}{d\sigma}\frac{a(\sigma)}{b(\sigma)}$$

Rule 8
: The breakaway lines of the root locus are separated by angles of $\frac{\pi}{\alpha}$, where α is the number of poles intersecting at the breakaway point.

Rule 9
: The breakaway root-loci follow asymptotes that intersect the real axis at angles φ_ω given by:

$$\phi_\omega = \frac{\pi + 2N\pi}{p - z}, \quad N = 0, 1, ...p - z - 1$$

The origin of these asymptotes, OA, is given as the sum of the pole locations, minus the sum of the zero locations, divided by the difference between the number of poles and zeros:

$$OA = \frac{\sum_p P_i - \sum_z Z_i}{p - z}$$

The OA point should lie on the real axis.

Rule 10

The branches of the root locus cross the imaginary axis at points where the **angle equation** value is π (180^o).

Rule 11

The angles that the root locus branch makes with a complex-conjugate pole or zero is determined by analyzing the **angle equation** at a point infinitessimally close to the pole or zero. The angle of departure, φ_d is given by the following equation:

$$\sum_p \psi_i + \sum_z \rho_i + \phi_d = \pi$$

The angle of arrival, φ_a, is given by:

$$\sum_z \rho_i + \sum_p \psi_i + \phi_a = \pi$$

We will explain these rules in the rest of the chapter.

Root Locus Equations

Here are the two major equations:

S-Domain Equations	Z-Domain Equations
$1 + KG(s)H(s) = 0$	$1 + K\overline{GH}(z) = 0$
$\angle KG(s)H(s) = 180^o$	$\angle K\overline{GH}(z) = 180^o$

[Root Locus Equations]

Note that the sum of the angles of all the poles and zeros must equal to 180.

Number of Asymptotes

If the number of explicit zeros of the system is denoted by Z (uppercase z), and the number of poles of the system is given by P, then the number of asymptotes (N_a) is given by:

$$N_a = P - Z$$

[Number of Asymptotes]

The angles of the symptotes are given by:

$$\phi_k = (2k+1)\frac{\pi}{P-Z}$$ **[Angle of Asymptotes]**

for values of $k = [0, 1, ...N_a]$.

The angles for the asymptotes are measured from the positive real-axis

Asymptote Intersection Point

The asymptotes intersect the real axis at the point:

$$\sigma_0 = \frac{\sum_P - \sum_Z}{P-Z}$$ **[Origin of Asymptotes]**

Where $\sum_P$ is the sum of all the locations of the poles, and $\sum_Z$ is the sum of all the locations of the explicit zeros.

Breakaway Points

The breakaway points are located at the roots of the following equation:

$\frac{G(s)H(s)}{ds} = 0$ or $\frac{\overline{GH}(z)}{dz} = 0$ **[Breakaway Point Locations]**

the breakaway point equation can be difficult to solve, so many times the actual location is approximated.

Root Locus and Stability

The root locus proceedure should produce a graph of where the poles of the system are for all values of gain K. When any or all of the roots of D are in the unstable region, the system is unstable. When any of the roots are in the marginally stable region, the system is marginally stable (oscillatory). When all of the roots of D are in the stable region, then the system is stable.

It is important to note that a system that is stable for gain K_1 may become unstable for a different gain K_2. Some systems may have poles that cross over from stable to unstable multiple times, giving multiple gain values for which the system is unstable.

Here is a quick refresher:

Region	S-Domain		Z-Domain	
Stable Region	Left-Hand S Plane	$\sigma > 0$	Inside the Unit Circle	$\lvert z\rvert < 1$
Marginally Stable Region	The vertical axis	$\sigma = 0$	The Unit Circle	$\lvert z\rvert = 1$

Unstable Region	Right-Hand S Plane	$\sigma < 0$	Outside the Unit Circle,	$\|z\| > 1$

Examples

Example 1: First-Order System

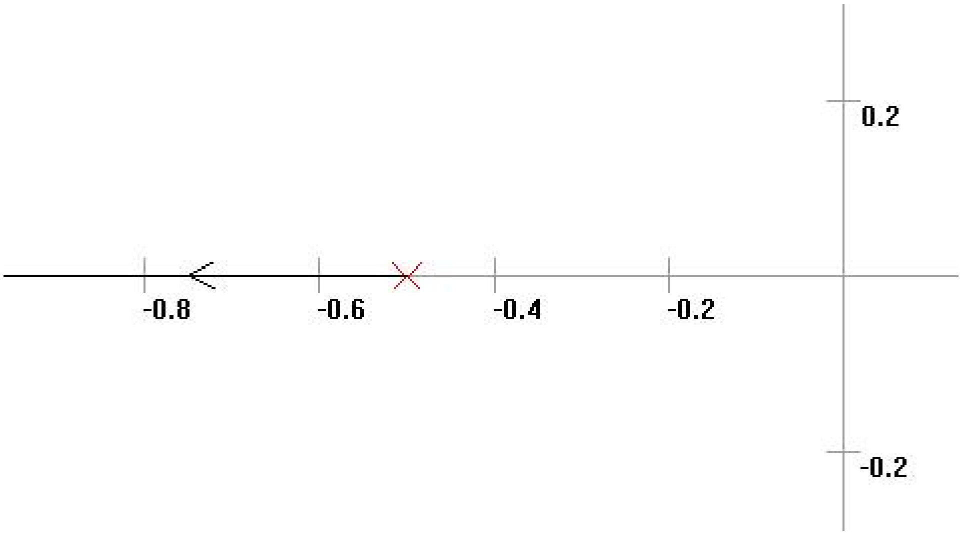
Find the root-locus of the closed-loop system:

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

If we look at the characteristic equation, we can quickly solve for the single pole of the system:

$$D(s) = 1 + 2s = 0$$
$$s = -\frac{1}{2}$$

We plot that point on our root-locus graph, and everything on the real axis to the left of that single point is on the root locus (from the rules, above). Therefore, the root locus of our system looks like this:



From this image, we can see that for *all values of gain* this system is stable.

Example 2: Third Order System

We are given a system with three real poles, shown by the transfer function:

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

Is this system stable?

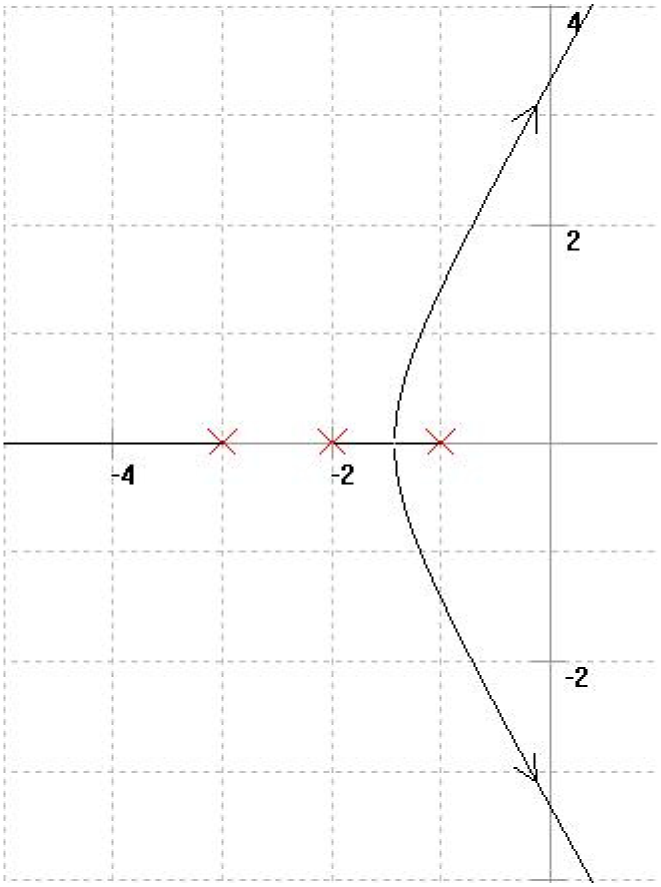
To answer this question, we can plot the root-locus. First, we draw the poles on the graph at locations -1, -2, and -3. The real-axis between the first and second poles is on the root-locus, as well as the real axis to the left of the third pole. We know also that there is going to be breakaway from the real axis at some point. The origin of asymptotes is located at:

$$OA = \frac{-1 + -2 + -3}{3} = -2,$$

and the angle of the asymptotes is given by:

$$\phi = 180$$

We know that the breakaway occurs between the first and third poles, so we will estimate the exact breakaway point. Drawing the root-locus gives us the graph below.



We can see that for low values of gain the system is stable, but for higher values of gain, the system becomes unstable.

Example: Complex-Conjugate Zeros

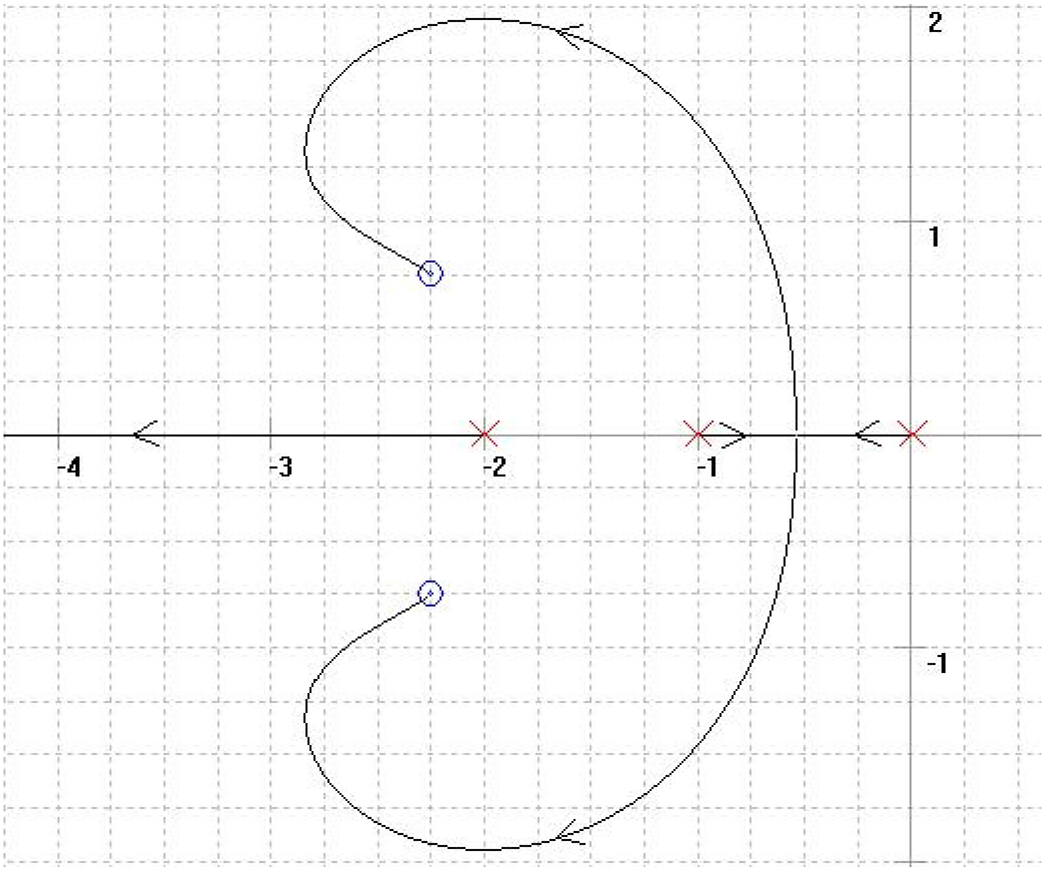
Find the root-locus graph for the following system transfer function:

$$T(s) = K\frac{s^2 - 4.5s + 5.625}{s(s+1)(s+2)}$$

If we look at the denominator, we have poles at the origin, -1, and -2. Following **Rule 4**, we know that the real-axis between the first two poles, and the real axis after the third pole are all on the root-locus. We also know that there is going to be a breakaway point between the first two poles, so that they can approach the complex conjugate zeros. If we use the quadratic equation on the numerator, we can find that the zeros are located at:

$$s = (2.25 + j0.75), (2.25 - j0.75)$$

If we draw our graph, we get the following:



We can see from this graph that the system is stable for all values of K.

Example: Root-Locus Using MATLAB/Octave

Use MATLAB, Octave, or another piece of mathematical simulation software to produce the root-locus graph for the following system:

$$T(s) = K\frac{s+2}{(s+3)(s^2+2s+2)}$$

First, we must multiply through in the denominator:

$$D(s) = s^3 + 5s^2 + 8s + 6$$

Now, we can generate the coefficient vectors from the numerator and denominator:

```
num = [0 0 1 2];
den = [1 5 8 6];
```

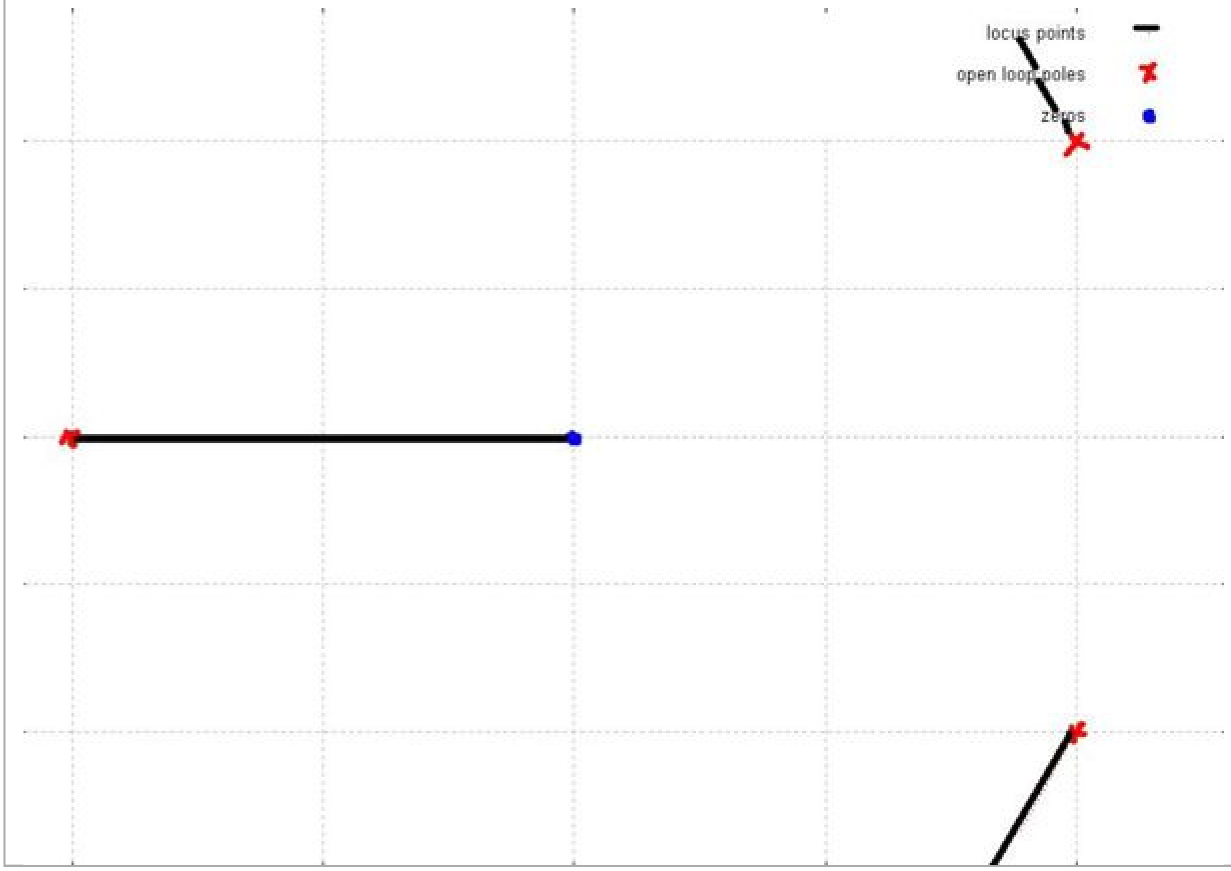
Next, we can feed these vectors into the **rlocus** command:

```
rlocus(num, den);
```

Note:In Octave, we need to create a system structure first, by typing:

```
sys = tf(num, den);
rlocus(sys);
```

Either way, we generate the following graph:



Nyquist Criterion

Nyquist Stability Criteria

The **Nyquist Stability Criteria** is a test for system stability, just like the Routh-Hurwitz test, or the Root-Locus Methodology. However, the Nyquist Criteria can also give us additional information about a system. Routh-Hurwitz and Root-Locus can tell us where the poles of the system are for particular values of gain. By altering the gain of the system, we can determine if any of the poles move into the RHP, and therefore become unstable. The Nyquist Criteria, however, can tell us things about the *frequency characteristics* of the system. For instance, some systems with constant gain might be stable for low-frequency inputs, but become unstable for high-frequency inputs.

> Here is an example of a system responding differently to different frequency input values: Consider an ordinary glass of water. If the water is exposed to ordinary sunlight, it is unlikely to heat up too much. However, if the water is exposed to higher-frequency microwave radiation (from inside your microwave oven, for instance), the water can quickly heat up to a boil.

Also, the Nyquist Criteria can tell us things about the phase of the input signals, the time-shift of the system, and other important information.

Contours

A **contour** is a complicated mathematical construct, but luckily we only need to worry ourselves with a few points about them. We will denote contours with the greek letter Γ (gamma). Contours are lines, drawn on a graph, that follow certain rules:

1. The contour must close (it must form a complete loop)
2. The contour may not cross directly through a pole of the system.
3. Contours must have a direction (clockwise or counterclockwise, generally).
4. A contour is called "simple" if it has no self-intersections. We only consider simple contours here.

Once we have such a contour, we can develop some important theorems about them, and finally use these theorems to derive the **Nyquist stability criterion**.

Argument Principal

Here is the argument principal, that we will use to derive the stability criterion. Do not worry if you do not understand all the terminology, we will walk through it:

> The Argument Principle
>
> If we have a contour, Γ, drawn in one plane (say the complex laplace plane, for instance), we can map that contour into another plane, the F(s) plane, by transforming the contour with the function F(s). The resultant contour, $\Gamma_{F(s)}$ will circle the origin point of the F(s) plane N times, where N is equal to the difference between Z and P (the number of zeros and poles of the function F(s), respectively).

When we have our contour, Γ, we transform it into $\Gamma_{F(s)}$ by plugging every point of the contour into the function F(s), and taking the resultant value to be a point on the transformed contour.

Example: First Order System

Let's say, for instance, that Γ is a unit square contour in the complex s plane. The verticies of the square are located at points E, F, G, H, as follows:

$$I = 1 + j1$$
$$J = 1 - j1$$
$$K = -1 - j1$$
$$L = -1 + j1$$

we must also specify the direction of our contour, and we will say (arbitrarily) that it is a clockwise contour (travels from I to J to K to L). We will also define our tranform function, F(s), to be the following:

$$F(s) = s2 + 1$$

We can factor the denominator of F(s), and we can show that there is one zero at s → -0.5, and no poles. Plotting this root on the same graph as our contour, we see clearly that it lies within the contour. Since s is a complex variable, defined with real and imagnary parts as:

$$s = \sigma + j\omega$$

We know that F(s) must also be complex. We will say, for reasons of simplicy, that the axes in the F(s) plane are u and v, and are related as such:

$$F(s) = u + jv = 2(\sigma + j\omega) + 1$$

From this relationship, we can define u and v in terms of σ and ω:

$$u = 2\sigma + 1$$
$$v = 2\omega$$

Now, to transform Γ, we will plug every point of the contour into F(s), and the resultant values will be the points of $\Gamma_{F(s)}$. We will solve for complex values u and v, and we will start with the verticies, because they are the simplest examples:

$$u + jv = F(I) = 3 + j2$$
$$u + jv = F(J) = 3 - j2$$
$$u + jv = F(K) = -1 + j2$$
$$u + jv = F(L) = -1 - j2$$

We can take the lines in between the vertices as a function of s, and plug the entire function into the transform. Luckily, because we are using straight lines, we can simplify very much:

- Line from I to J: $\sigma = 1, u = 3, v = \omega$
- Line from J to K: $\omega = -1, u = 2\sigma + 1, v = -1$
- Line from K to L: $\sigma = -1, u = -1, v = \omega$
- Line from L to I: $\omega = 1, u = 2\sigma + 1, v = 1$

And when we graph these functions, from virtex to virtex, we see that the resultant contour in the F(s) plane is a square, but not centered at the origin, and larger in size. Notice how the contour encircles the origin of the F(s) plane one time. This will be important later on.

Example:Second-Order System

Let's say that we have a slightly more complicated mapping function:

$$F(s) = \frac{s + 0.5}{2s^2 + 2s + 1}$$

We can see clearly that F(s) has a zero at s → -0.5, and a complex conjugate set of poles at s → -0.5 + j0.5 and s → -0.5 - j0.5. We will use the same unit square contour, Γ, from above:

$$I = 1 + j1$$
$$J = 1 - j1$$
$$K = -1 - j1$$
$$L = -1 + j1$$

We can see clearly that the poles and the zero of F(s) lie within Γ. Setting F(s) to u + jv and solving, we get the following relationships:

$$u + jv = F(\sigma + j\omega) = \frac{(\sigma + 0.5) + j(\omega)}{(2\sigma^2 - 2\omega^2 + 2\sigma + 1) + j(2\sigma\omega + \omega)}$$

This is a little difficult now, because we need to simplify this whole expression, and separate it out into real and imaginary parts. There are two methods to doing this, neither of which is short or easy enough to demonstrate here to entirety:

1. We convert the numerator and denominator polynomials into a polar representation in terms of r and θ, then perform the division, and then convert back into rectangular format.
2. We plug each segment of our contour into this equation, and simplify numerically.

The Nyquist Contour

The nyquist contour, the contour that makes the entire nyquist criterion work, must encircle the entire right half of the complex s plane. Remember that if a pole to the closed-loop transfer function (or equivalently a zero of the characteristic equation) lies in the right-half of the s plane, the system is an unstable system.

To satisfy this requirement, the nyquist contour takes the shape of an infinite semi-circle that encircles the entire right-half of the s plane.

Nyquist Criteria

Let us first introduce the most important equation when dealing with the Nyquist criterion:

$$N = Z - P$$

Where:

- N is the number of encirclements of the (-1, 0) point.
- Z is the number of zeros of the characteristic equation.
- P is the number of poles of the characteristic equation.

With this equation stated, we can now state the **Nyquist Stability Criterion**:

Nyquist Stability Criterion

> A feedback control system is stable, if and only if the contour $\Gamma_{F(s)}$ in the F(s) plane does not encircle the (-1, 0) point when P is 0.
>
> A feedback control system is stable, if and only if the contour $\Gamma_{F(s)}$ in the F(s) plane encircles the (-1, 0) point a number of times equal to the number of poles of F(s) enclosed by Γ.

In other words, if P is zero then N must equal zero. Otherwise, N must equal P. Essentially, we are saying that Z must always equal zero, because Z is the number of zeros of the characteristic equation (and therefore the number of poles of the closed-loop transfer function) that are in the right-half of the s plane.

Keep in mind that we don't necessarily know the locations of all the zeros of the characteristic equation. So if we find, using the nyquist criterion, that the number of poles is not equal to N, then we know that there must be a zero in the right-half plane, and that therefore the system is unstable.

Nyquist ↔ Bode

A careful inspection of the Nyquist plot will reveal a surprising relationship to the Bode plots of the system. If we use the Bode phase plot as the angle θ, and the Bode magnitude plot as the distance r, then it becomes apparent that the Nyquist plot of a system is simply the polar representation of the Bode plots.

To obtain the Nyquist plot from the Bode plots, we take the phase angle and the magnitude value at each frequency ω. We convert the magnitude value from decibels back into gain ratios. Then, we plot the ordered pairs (r, θ) on a polar graph.

Nyquist in the Z Domain

The Nyquist Criteria can be utilized in the digital domain in a similar manner as it is used with analog systems. The primary difference in using the criteria is that the shape of the Nyquist contour must change to encompass the unstable region of the Z plane. Therefore, instead of an infinitesimal semi-circle, the Nyquist contour for digital systems is a counter-clockwise unit circle. By changing the shape of the contour, the same N = P - Z equation holds true, and the resulting Nyquist graph will typically look identical to one from an analog system, and can be interpreted in the same way.

State-Space Stability

State-Space Stability

Here are some definitions:

Unstable

A system is said to be unstable if the system response approaches infinity as time approaches infinity. If our system is G(t), then, we can say a system is unstable if:

$$\lim_{t\to\infty} \|G(t)\| = \infty$$

Asymptotically Stable

A system is said to be **asymptotically stable** if the system response approaches 0 as time approaches infinity. Put mathematically:

$$\lim_{t\to\infty} \|G(t)\| = 0$$

Eigenvalues and Stability

An LTI system is stable (asymptotically stable, see above) if all the eigenvalues of A have negative real parts. Consider the following state equation:

$$x' = Ax(t) + Bu(t)$$

We can take the Laplace Transform of both sides of this equation, using initial conditions of $x_0 = 0$:

$$sX(s) = AX(s) + BU(s)$$

Subtract AX(s) from both sides:

$$sX(s) - AX(s) = BU(s)$$
$$(sI - A)X(s) = BU(s)$$

Assuming (sI - A) is nonsingular, we can multiply both sides by the inverse:

$$X(s) = (sI - A)^{-1}BU(s)$$

Now, if we remember our formula for finding the matrix inverse from the adjoint matrix:

$$A^{-1} = \frac{\operatorname{adj}(A)}{|A|}$$

We can use that definition here:

$$X(s) = \frac{\mathrm{adj}(sI - A)BU(s)}{|(sI - A)|}$$

Let's look at the denominator (which we will now call D(s)) more closely. To be stable, the following condition must be true:

$$D(s) = |(sI - A)| = 0$$

And if we substitute λ for s, we see that this is actually the characteristic equation of matrix A! This means that the values for s that satisfy the equation (to poles of our transfer function) are precisely the eigenvalues of matrix A. In the S domain, it is required that all the poles of the system be located in the left-half plane, and therefore all the eigenvalues of A must have negative real parts.

Positive Definiteness

These terms are important, and will be used in further discussions on this topic.

- f(x) is **positive definate** if f(x) > 0 for all x.
- f(x) is **positive semi-definate** if $f(x) \ge 0$ for all x, and f(x) = 0 only if x = 0.
- f(x) is **negative definate** if f(x) < 0 for all x.
- f(x) is **negative semi-definate** if $f(x) \le 0$ for all x, and f(x) = 0 only if x = 0.

A matrix X is positive definate if all it's principle minors are positive. Also, a matrix X is positive definite if all it's eigenvalues have positive real parts. These two methods may be used interchangeably.

Positive definiteness is a very important concept. So much so that the Lyapunov stability test depends on it. The other categorizations are not as important, but are included here for completeness.

Lyapunov Stability

First, let us define **Equilibrium State**, and other terms.

Equilibrium State

The state $\hat{x}$ is the equilibrium state of the following system:

$$x' = f(x)$$

If $f(\hat{x}) = 0$.

Zero State

The zero state of a system is the value for which f(0) = 0.

Lyapunov's Equation

For linear systems, we can use the **Lyapunov Equation**, below, to determine if a system is stable. We will state the Lyapunov Equation first, and then state the **Lyapunov Stability Theorem**.

$$MA + A^T M = -N$$

Lyapunov Stability Theorem

An LTI system $x' = Ax$ is stable if there exists a matrix M that satisfies the **Lyapunov Equation** where Where N is positive definite, and M is positive definite.

Notice that for the Lyapunov Equation to be satisfied, the matrices must be compatible sizes. In fact, matrices A, M, and N must all be square matrices of equal size.

Controllers and Compensators

There are a number of preexisting devices for use in system control, such as lead and lag compensators, and powerful PID controllers. PID controllers are so powerful that many control engineers may use no other method of system control! The chapters in this section will discuss some of the common types of system compensators and controllers.

Controllability and Observability

System Interaction

In the world of control engineering, there are a slew of systems available that need to be controlled. The task of a control engineer is to design controller and compensator units to interact with these pre-existing systems. However, some systems simply cannot be controlled (or, more often, cannot be controlled in specific ways). The concept of **controllability** refers to the ability of a controller to arbitrarily alter the functionality of the system plant.

The state-variable of a system, *x*, which represents the internal workings of the system that can be separate from the regular input-output relationship of the system also needs to be measured or *observed*. The term **observability** describes whether the internal state variables of the system can be externally measured.

Controllability

Observability

The state-variables of a system might not be able to be measured for any of the following reasons:

1. The location of the particular state variable might not be physically accessible (a capacitor or a spring, for instance).
2. There are no appropriate instruments to measure the state variable, or the state-variable might be measured in units for which there does not exist any measurement device.
3. The state-variable is a derived "dummy" variable that has no physical meaning.

If things cannot be directly observed, for any of the reasons above, it can be necessary to calculate or **estimate** the values of the internal state variables, using only the input/output relation of the system, and the output history of the system from the starting time. In other words, we must ask whether or not it is possible to determine what the inside of the system (the internal system states) is like, by only observing the outside performance of the system (input and output)? We can provide the following formal definition of mathematical observability:

An initial state, $x(t_0)$ is **observable** if it can be determined from the system output y(t) that has been observed through the time interval $t_0 < t < t_f$. If the initial state cannot be so determined, the system is **unobservable**.

A system is said to be **observable** if all the possible initial states of the system can be observed. Systems that fail this criteria are said to be **unobservable**.

The observability of the system is dependant only on the system states and the system output, so we can simplify our state equations to remove the input terms:

$$x'(t) = Ax(t)$$
$$y(t) = Cx(t)$$

Therefore, we can show that the observability of the system is dependant only on the coefficient matrices A and C. We can show precisely how to determine whether a system is observable, using only these two matrices. If we have the matrix Q:

$$Q = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{p-1} \end{bmatrix}$$

Remember that matrix A has dimensions $p \times p$, and matrix C has dimensions $r \times p$.

we can show that the system is observable if and only if the Q matrix has a rank of *p*. Notice that the Q matrix has the dimensions *pr* &times *p*.

System Specifications

This page of the Control Systems book is a stub. You can help by expanding this section.

System Specification

There are a number of different specifications that might need to be met by a new system design. In this chapter we will talk about some of the specifications that systems use, and some of the ways that engineers analyze and quantify systems.

Steady-State Accuracy

Sensitivity

The **sensitivity** of a system is a parameter that is specified in terms of a given output and a given input. The sensitivity measures how much change is caused in the output by small changes to the reference input. Sensitive systems have very large changes in output in response to small changes in the input. The sensitivity of system H to input X is denoted as:

$$S_H^X(s)$$

Disturbance Rejection

All physically-realized systems have to deal with a certain amount of noise and disturbance. The ability of a system to ignore the noise is known as the **disturbance rejection** of the system.

Control Effort

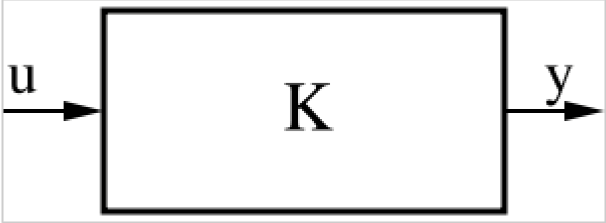
The control effort is the amount of energy or power necessary for the controller to perform it's duty.

Controllers

Controllers

There are a number of different types of control systems that have already been designed and studied extensively. These controllers are the P, PI, PD, and PID controllers.

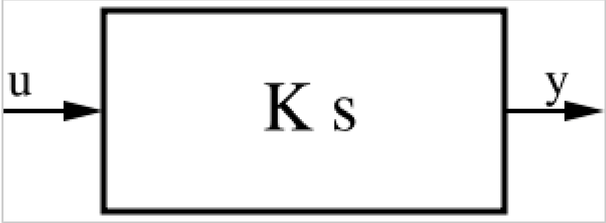
Proportional Controllers



A Proportional controller block diagram

Proportional controllers are simply gain values. These are essentially multiplicative coefficients, usually denoted with a K.

Derivative Controllers



A Proportional-Derivative controller block diagram

In the Laplace domain, we can show the derivative of a signal using the following notation:

$$D(s) = \mathcal{L}\left\{f'(t)\right\} = sF(s) - f(0)$$

Since most systems that we are considering have zero initial condition, this simplifies to:

$$D(s) = \mathcal{L}\left\{f'(t)\right\} = sF(s)$$

The derivative controllers are implemented to account for future values, by taking the derivative, and controlling based on where the signal is going to be in the future. Derivative controllers should be used with care, because even small amount of high-frequency noise can cause very large derivatives, which appear like amplified noise. Also, Derivative controllers are difficult to implement perfectly in hardware or software, so frequently solutions involving only integral controllers or proportional controllers are preferred over using derivative controllers.

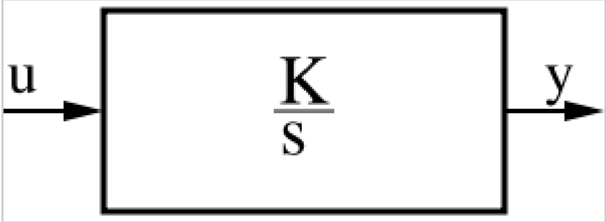
Z-Domain Derivatives

We won't derive this equation here, but suffice it to say that the following equation in the Z-domain performs the same function as the Laplace-domain derivative:

$$D(z) = \frac{z-1}{Tz}$$

Where T is the sampling time of the signal.

Integral Controllers



A Proportional-Integral Controller block diagram

To implemenent an Integral in a Laplace domain transfer function, we use the following:

$$\mathcal{L}\left\{\int_0^t f(t)\,dt\right\} = \frac{1}{s}F(s)$$

Integral controllers of this type add up the area under the curve for past time. In this manner, a PI controller (and eventually a PID) can take account of the past performance of the controller, and correct based on past errors.

Z-Domain Integral

The integral controller can be implemented in the Z domain using the following equation:

$$D(z) = \frac{z+1}{z-1}$$

PID Controllers

What is PID control? PID can be described as a set of rules with which precise regulation of a closed-loop control system is obtained. Closed loop control means a method in which a real-time measurement of the process being controlled is constantly fed back to the controlling device to ensure that the value which is desired is, in fact, being realized. The mission of the controlling device is to make the measured value, usually known as the PROCESS VARIABLE, equal to the desired value, usually known as the SETPOINT. The very best way of accomplishing this task is with the use of the control algorithm we know as PID.

In its basic form, PID involves three mathematical control functions working together: Proportional-Integral-Derivative. The most important of these, Proportional Control, determines the magnitude of the difference between the SETPOINT and the PROCESS VARIABLE (known as ERROR), and then applies appropriate proportional changes to the CONTROL VARIABLE to eliminate ERROR. Many control systems will, in fact, work quite well with only Proportional Control. Integral Control examines the offset of SETPOINT and the PROCESS VARIABLE over time and corrects it when and if necessary. Derivative Control monitors the rate of change of the PROCESS VARIABLE and consequently makes changes to the OUTPUT VARIABLE to accomodate unusual changes.

Each of the three control functions is governed by a user-defined parameter. These parameters vary immensely from one control system to another, and, as such, need to be adjusted to optimize the precision of control. The process of determining the values of these parameters is known as PID Tuning.

PID Tuning, although considered "black magic" by many, really is, of course, always a well-defined technical process. There are several different methods of PID Tuning available, any of which will tune any system. Certain PID Tuning methods require more equipment than others, but usually result in more accurate results with less effort.

PID Transfer Function

The transfer function for a standard PID controller is an addition of the Proportional, the Integral, and the Differential controller transfer functions (hence the name, PID). Also, we give each term a gain constant, to control the weight that each factor has on the final output:

$$D(s) = K_p + \frac{K_i}{s} + K_d s$$

[PID]

Where the coefficients control the weight of each part of the controller.

PID Tuning

The process of selecting the various coefficient values to make a PID controller perform correctly is called **PID Tuning**.

Digital PID

In the Z domain, the PID controller has the following transfer function:

$$D(z) = K_p + K_i \frac{T}{2}\left[\frac{z+1}{z-1}\right] + K_d\left[\frac{z-1}{Tz}\right]$$

[Digital PID]

And we can convert this into a cannonical equation by manipulating the above equation to obtain:

$$D(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

Where:

$$a_0 = K_p + \frac{K_i T}{2} + \frac{K_d}{T}$$

$$a_1 = -K_p + \frac{K_i T}{2} + \frac{-2K_d}{T}$$

$$a_2 = \frac{K_d}{T}$$

$$b_1 = -1$$

$$b_2 = 0$$

Once we have the Z-domain transfer function of the PID controller, we can convert it into the digital time domain:

$$y[n] = x[n]a_0 + x[n-1]a_1 + x[n-2]a_2 - y[n-1]b_1 - y[n-2]b_2$$

And finally, from this difference equation, we can create a digital filter structure to implement the PID.

For more information about digital filter structures, see Digital Signal Processing

Bang-Bang Controllers

Despite the low-brow sounding name of the Band-Bang controller, it is a very useful tool that is only really available using digital methods. A better name perhaps for a bang-bang controller is an on/off controller, where a digital system makes decisions based on target and threshold values, and decides whether to turn the controller on and off.

Consider the example of a household furnace. The oil in a furnace burns at a specific temperature: It can't burn hotter or cooler. To control the temperature in your house then, the thermostat control unit decides when to turn the furnace on, and when to turn the furnace off. This on/off control scheme is a bang-bang controller.

Compensators

This page of the Control Systems book is a stub. You can help by expanding this section.

Compensation

There are a number of different compensation units that can be employed to help fix certain system metrics that are outside of a proper operating range. Most commonly, the phase characteristics are in need of compensation, especially if the magnitude response is to remain constant.

Phase Compensation

Occasionally, it is necessary to alter the phase characteristics of a given system, without altering the magnitude characteristics. To do this, we need to alter the frequency response in such a way that the phase response is altered, but the magnitude response is not altered. To do this, we implement a special variety of controllers known as **phase compensators**. They are called compensators because they help to improve the phase response of the system.

There are two general types of compensators: **Lead Compensators**, and **Lag Compensators**. If we combine the two types, we can get a special **Lead-Lag Compensator** system.

When designing and implementing a phase compensator, it is important to analyze the effects on the gain and phase margins of the system, to ensure that compensation doesnt cause the system to become unstable.

Phase Lead

The transfer function for a lead-compensator is as follows:

$$T_{lead}(s) = \frac{s - z}{s - p}$$ **[Lead Compensator]**

To make the compensator work correctly, the following property must be satisfied:

$$|z| < |p|$$

And both the pole and zero location should be close to the origin, in the RHP. Because there is only one pole and one zero, they both should be located on the real axis.

Phase lead compensators help to shift the poles of the transfer function to the left, which is beneficial for stability purposes.

Phase Lag

The transfer function for a lag compensator is the same as the lead-compensator, and is as follows:

$$T_{lead}(s) = \frac{s - z}{s - p}$$

However, in the lag compensator, the location of the pole and zero should be swapped:

$$|p| < |z|$$

Both the pole and the zero should be close to the origin, on the real axis.

The Phase lag compensator helps to improve the steady-state error of the system. The poles of the lag compensator should be very close together to help prevent the poles of the system from shifting right, and therefore reducing system stability.

Phase Lead-Lag

The transfer function of a **lead-lag compensator** is simply a multiplication of the lead and lag compensator transfer functions, and is given as:

$$T_{lead-lag}(s) = \frac{(s - z_1)(s - z_2)}{(s - p_1)(s - p_2)}.$$ **[Lead-Lag Compensator]**

Where typically the following relationship must hold true:

$$|p_1| > |z_1| > |z_2| > |p_2|$$

Appendicies

Appendix: Physical Models

Physical Models

This page will serve as a refresher for various different engineering disciplines on how physical devices are modeled. Models will be displayed in both time-domain and Laplace-domain input/output characteristics. The only information that is going to be displayed here will be the ones that are contributed by knowledgable contributors.

Electrical Systems

Mechanical Systems

Civil/Construction Systems

Chemical Systems

Appendix: Z Transform Mappings

Z Transform Mappings

There are a number of different mappings that can be used to convert a system from the complex Laplace domain into the Z-Domain. None of these mappings are perfect, and every mapping requires a specific starting condition, and focuses on a specific aspect to reproduce faithfully. One such mapping that has already been discussed is the **bilinear transform**, which, along with prewarping, can faithfully map the various regions in the s-plane into the corresponding regions in the z-plane. We will discuss some other potential mappings in this chapter, and we will discuss the pros and cons of each.

Bilinear Transform

The Bilinear transform converts from the Z-domain to the complex W domain. The W domain is not the same as the Laplace domain, although there are some similarities. Here are some of the similiarities between the Laplace domain and the W domain:

1. Stable poles are in the Left-Half Plane
2. Unstable poles are in the right-half plane
3. Marginally stable poles are on the vertical, imaginary axis

With that said, the bilinear transform can be defined as follows:

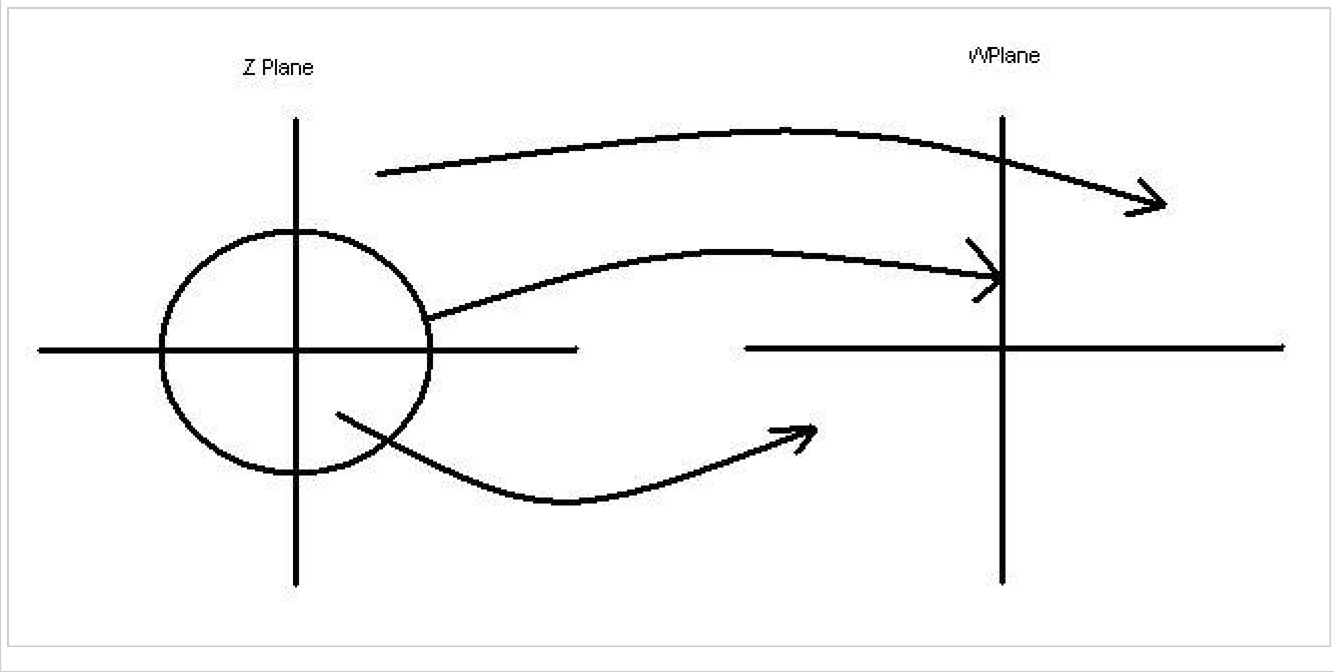
$$w = \frac{2}{T}\frac{z-1}{z+1}$$

[Bilinear Transform]

$$z = \frac{(T/2)+w}{(T/2)-w}$$

[Inverse Bilinear Transform]

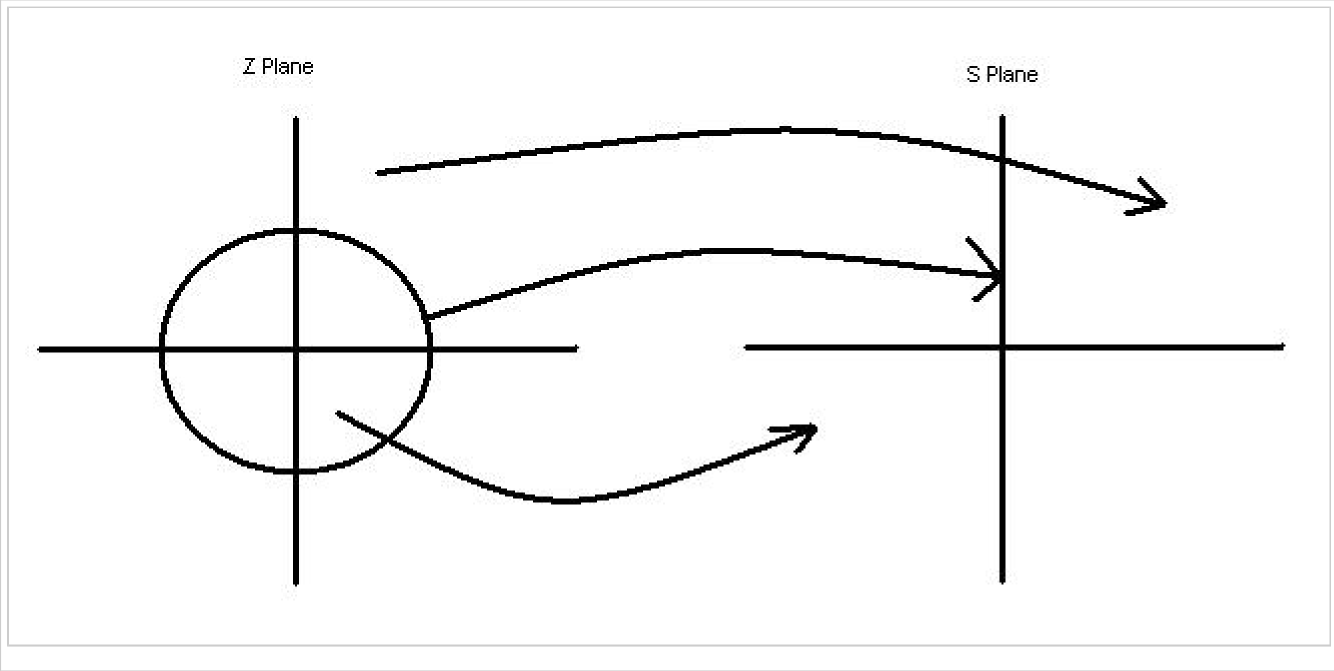
Graphically, we can show that the bilinear transform operates as follows:



Prewarping

The W domain is not the same as the Laplace domain, but if we employ the process of **prewarping** before we take the bilinear transform, we can make our results match more closely to the desired Laplace Domain representation.

Using prewarping, we can show the effect of the bilinear transform graphically:



Matched Z-Transform

If we have a function in the laplace domain that has been decomposed using partial fraction expansion, we generally have an equation in the form:

$$Y(s) = \frac{A}{s+\alpha_1} + \frac{B}{s+\alpha_2} + \frac{C}{s+\alpha_3} + ...$$

And once we are in this form, we can make a direct conversion between the s and z planes using the following mapping:

$$s + \alpha = 1 - z^{-1}e^{\alpha T}$$

[Matched Z Transform]

Pro
: A good direct mapping in terms of s and a single coefficient

Con
: requires the Laplace-domain function be decomposed using partial fraction expansion.

Simpson's Rule

$$s = \frac{3}{T}\frac{z^2}{z^2 + 4z^1 + 1}$$

[Simpson's Rule]

CON
: Essentially multiplies the order of the transfer function by a factor of 2. This makes things difficult when you are trying to physically implement the system.

(w, v) Transform

Given the following system:

$$Y(s) = G(s, z, z^\alpha)X(s)$$

Then:

$$w = \frac{2}{T}\frac{z-1}{z+1}$$

$$v(\alpha) = 1 - \alpha(1 - z^{-1}) + \frac{\alpha(\alpha - 1)}{z}(1 - z^{-1})^2$$

And:

[(w, v) Transform]

$$Y(z) = G(w, z, v(\alpha)) \left[X(z) - \frac{x(0)}{1 + z^{-1}} \right]$$

Pro

Directly maps a function in terms of z and s, into a function in terms of only z.

Con

Requires a function that is already in terms of s, z and α.

Z-Forms

Appendix: Transforms

Laplace Transform

The when we talk about the Laplace transform, we are actually talking about the version of the Laplace transform known as the **unilinear Laplace Transform**. The other version, the **Bilinear Laplace Transform** (not related to the Bilinear Transorm, below) is not used in this book.

The Laplace Transform is defined as:

$$F(s) = \mathcal{L}[f(t)] = \int_0^\infty f(t)e^{-st}dt$$ **[Laplace Transform]**

And the Inverse Laplace Transform is defined as:

$$f(t) = \mathcal{L}^{-1}\left\{F(s)\right\} = \frac{1}{2\pi}\int_{c-i\infty}^{c+i\infty} e^{ft}F(s)\,ds$$ **[Inverse Laplace Transform]**

Table of Laplace Transforms

This is a table of common laplace transforms.

Time Domain	Laplace Domain
$x(t) = \frac{1}{2\pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} X(s)e^{st}ds$	$X(s) = \int_{-\infty}^{\infty} x(t)e^{-st}dt$
$\delta(t)$	1
$\delta(t-a)$	e^{-as}
$u(t)$	$\frac{1}{s}$
$u(t-a)$	$\frac{e^{-as}}{s}$
$tu(t)$	$\frac{1}{s^2}$
$t^n u(t)$	$\frac{n!}{s^{n+1}}$
$\frac{1}{\sqrt{\pi t}}u(t)$	$\frac{1}{\sqrt{s}}$

$e^{at}u(t)$	$\frac{1}{s-a}$
$t^n e^{at}u(t)$	$\frac{n!}{(s-a)^{n+1}}$
$\cos\omega t u(t)$	$\frac{s}{s^2+\omega^2}$
$\sin\omega t u(t)$	$\frac{\omega}{s^2+\omega^2}$
$\cosh\omega t u(t)$	$\frac{s}{s^2-\omega^2}$
$\sinh\omega t u(t)$	$\frac{\omega}{s^2-\omega^2}$
$e^{at}\cos\omega t u(t)$	$\frac{s-a}{(s-a)^2+\omega^2}$
$e^{at}\sin\omega t u(t)$	$\frac{\omega}{(s-a)^2+\omega^2}$
$\frac{1}{2\omega^3}(\sin\omega t - \omega t\cos\omega t)$	$\frac{1}{(s^2+\omega^2)^2}$
$\frac{t}{2\omega}\sin\omega t$	$\frac{s}{(s^2+\omega^2)^2}$
$\frac{1}{2\omega}(\sin\omega t + \omega t\cos\omega t)$	$\frac{s^2}{(s^2+\omega^2)^2}$
e^{At}	$(sI-A)^{-1}$

Properties of the Laplace Transform

This is a table of the most important properties of the laplace transform.

Property	Definition
Linearity	$\mathcal{L}\left\{af(t)+bg(t)\right\} = aF(s)+bG(s)$
Differentiation	$\mathcal{L}\{f'\} = s\mathcal{L}\{f\} - f(0^-)$ $\mathcal{L}\{f''\} = s^2\mathcal{L}\{f\} - sf(0^-) - f'(0^-)$ $\mathcal{L}\left\{f^{(n)}\right\} = s^n\mathcal{L}\{f\} - s^{n-1}f(0^-) - \cdots - f^{(n-1)}(0^-)$
Frequency Division	$\mathcal{L}\{tf(t)\} = -F'(s)$

	$\mathcal{L}\{t^n f(t)\} = (-1)^n F^{(n)}(s)$
Frequency Integration	$\mathcal{L}\left\{\frac{f(t)}{t}\right\} = \int_s^\infty F(\sigma)\, d\sigma$
Time Integration	$\mathcal{L}\left\{\int_0^t f(\tau)\, d\tau\right\} = \mathcal{L}\left\{u(t) * f(t)\right\} = \frac{1}{s}F(s)$
Scaling	$\mathcal{L}\left\{f(at)\right\} = \frac{1}{a}F\left(\frac{s}{a}\right)$
Initial value theorem	$f(0^+) = \lim_{s\to\infty} sF(s)$
Final value theorem	$f(\infty) = \lim_{s\to 0} sF(s)$
Frequency Shifts	$\mathcal{L}\left\{e^{at}f(t)\right\} = F(s-a)$ $\mathcal{L}^{-1}\left\{F(s-a)\right\} = e^{at}f(t)$
Time Shifts	$\mathcal{L}\left\{f(t-a)u(t-a)\right\} = e^{-as}F(s)$ $\mathcal{L}^{-1}\left\{e^{-as}F(s)\right\} = f(t-a)u(t-a)$
Convolution Theorem	$\mathcal{L}\{f(t) * g(t)\} = F(s)G(s)$

Where:

$$f(t) = \mathcal{L}^{-1}\{F(s)\}$$
$$g(t) = \mathcal{L}^{-1}\{G(s)\}$$
$$s = \sigma + j\omega$$

Convergence of the Laplace Integral

Properties of the Laplace Transform

Fourier Transform

The Fourier Transform is used to break a time-domain signal into it's frequency domain components. The Fourier Transform is very closely related to the Laplace Transform, and is only used in place of the Laplace transform when the system is being analyzed in a frequency context.

The Fourier Transform is defined as:

$$F(j\omega) = \mathcal{F}[f(t)] = \int_0^\infty f(t)e^{-j\omega t}dt$$ **[Fourier Transform]**

And the Inverse Fourier Transform is defined as:

$$f(t) = \mathcal{F}^{-1}\left\{F(j\omega)\right\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{-j\omega t}d\omega$$ **[Inverse Fourier Transform]**

Table of Fourier Transforms

This is a table of common fourier transforms.

Time Domain	Fourier Domain
$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\omega$	$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$
1	$2\pi\delta(\omega)$
$-0.5 + u(t)$	$\frac{1}{j\omega}$
$\delta(t)$	1
$\delta(t-c)$	$e^{-j\omega c}$
$u(t)$	$\pi\delta(\omega) + \frac{1}{j\omega}$
$e^{-bt}u(t)$	$\frac{1}{j\omega + b}$
$\cos\omega_0 t$	$\pi[\delta(\omega+\omega_0) + \delta(\omega-\omega_0)]$
$\cos(\omega_0 t + \theta)$	$\pi[e^{-j\theta}\delta(\omega+\omega_0) + e^{j\theta}\delta(\omega-\omega_0)]$
$\sin\omega_0 t$	$j\pi[\delta(\omega+\omega_0) - \delta(\omega-\omega_0)]$
$\sin(\omega_0 t + \theta)$	$j\pi[e^{-j\theta}\delta(\omega+\omega_0) - e^{j\theta}\delta(\omega-\omega_0)]$
$rect(\frac{t}{\tau})$	$\tau sinc\frac{\tau\omega}{2\pi}$
$\tau sinc\frac{\tau t}{2\pi}$	$2\pi p_\tau(\omega)$
$(1 - \frac{2\|t\|}{\tau})p_\tau(t)$	$\frac{\tau}{2}sinc^2\frac{\tau\omega}{4\pi}$
$\frac{\tau}{2}sinc^2(\frac{\tau t}{4\pi})$	

	$2\pi(1-\frac{2\lvert\omega\rvert}{\tau})p_\tau(\omega)$

Note: $sinc(x) = \sin(x)/x$; $p_\tau(t)$ is the rectangular pulse function of width τ

Table of Fourier Transform Properties

This is a table of common properties of the fourier transform.

	Signal	Fourier transform unitary, angular frequency	Fourier transform unitary, ordinary frequency	Remarks
	$g(t) \equiv$ $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} G(\omega)e^{i\omega t}d\omega$	$G(\omega) \equiv$ $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} g(t)e^{-i\omega t}dt$	$G(f) \equiv$ $\int_{-\infty}^{\infty} g(t)e^{-i2\pi ft}dt$	
1	$a\cdot g(t)+b\cdot h(t)$	$a\cdot G(\omega)+b\cdot H(\omega)$	$a\cdot G(f)+b\cdot H(f)$	Linearity
2	$g(t-a)$	$e^{-ia\omega}G(\omega)$	$e^{-i2\pi af}G(f)$	Shift in time domain
3	$e^{iat}g(t)$	$G(\omega-a)$	$G\left(f-\frac{a}{2\pi}\right)$	Shift in frequency domain, dual of 2
4	$g(at)$	$\frac{1}{\lvert a\rvert}G\left(\frac{\omega}{a}\right)$	$\frac{1}{\lvert a\rvert}G\left(\frac{f}{a}\right)$	If $\lvert a\rvert$ is large, then $g(at)$ is concentrated around 0 and $\frac{1}{\lvert a\rvert}G\left(\frac{\omega}{a}\right)$ spreads out and flattens
5	$G(t)$	$g(-\omega)$	$g(-f)$	Duality property of the Fourier transform. Results from swapping "dummy" variables of t and ω.
6	$\frac{d^n g(t)}{dt^n}$	$(i\omega)^n G(\omega)$	$(i2\pi f)^n G(f)$	Generalized derivative property of the Fourier transform

7	$t^n g(t)$	$i^n \dfrac{d^n G(\omega)}{d\omega^n}$	$\left(\dfrac{i}{2\pi}\right)^n \dfrac{d^n G(f)}{df^n}$	This is the dual to 6
8	$(g * h)(t)$	$\sqrt{2\pi} G(\omega) H(\omega)$	$G(f) H(f)$	$g * h$ denotes the convolution of g and h — this rule is the convolution theorem
9	$g(t) h(t)$	$\dfrac{(G * H)(\omega)}{\sqrt{2\pi}}$	$(G * H)(f)$	This is the dual of 8

Convergence of the Fourier Integral

Properties of the Fourier Transform

Z-Transform

The Z-transform is used primarily to convert discrete data sets into a continuous representation. The Z-transform is notationally very similar to the star transform, except that the Z transform does not take explicit account for the sampling period. The Z transform has a number of uses in the field of digital signal processing, and the study of discrete signals in general, and is useful because Z-transform results are extensively tabulated, whereas star-transform results are not.

The Z Transform is defined as:

$$X(z) = \mathcal{Z}[x[n]] = \sum_{i=-\infty}^{\infty} x[n] z^{-n}$$

[Z Transform]

Inverse Z Transform

The inverse Z Transform is a highly complex transformation, and might be inaccessible to students without enough background in calculus. However, students who are familiar with such integrals are encouraged to perform some inverse Z transform calculations, to verify that the formula produces the tabulated results.

$$x[n] = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz$$

[Inverse Z Transform]

Z-Transform Tables

	Signal, $x[n]$	**Z-transform,** $X(z)$	**ROC**

1	$\delta[n]$	1	all z
2	$\delta[n-n_0]$	$\frac{1}{z^{n_0}}$	all z
3	$u[n]$	$\frac{z}{z-1}$	$\|z\| > 1$
4	$a^n u[n]$	$\frac{1}{1-az^{-1}}$	$\|z\| > \|a\|$
5	$na^n u[n]$	$\frac{az^{-1}}{(1-az^{-1})^2}$	$\|z\| > \|a\|$
6	$-a^n u[-n-1]$	$\frac{1}{1-az^{-1}}$	$\|z\| < \|a\|$
7	$-na^n u[-n-1]$	$\frac{az^{-1}}{(1-az^{-1})^2}$	$\|z\| < \|a\|$
8	$\cos(\omega_0 n)u[n]$	$\frac{1-z^{-1}\cos(\omega_0)}{1-2z^{-1}\cos(\omega_0)+z^{-2}}$	$\|z\| > 1$
9	$\sin(\omega_0 n)u[n]$	$\frac{z^{-1}\sin(\omega_0)}{1-2z^{-1}\cos(\omega_0)+z^{-2}}$	$\|z\| > 1$
10	$a^n \cos(\omega_0 n)u[n]$	$\frac{1-az^{-1}\cos(\omega_0)}{1-2az^{-1}\cos(\omega_0)+a^2z^{-2}}$	$\|z\| > \|a\|$
11	$a^n \sin(\omega_0 n)u[n]$	$\frac{az^{-1}\sin(\omega_0)}{1-2az^{-1}\cos(\omega_0)+a^2z^{-2}}$	$\|z\| > \|a\|$

Modified Z-Transform

The Modified Z-Transform is similar to the Z-transform, except that the modified version allows for the system to be subjected to any arbitrary delay, by design. The Modified Z-Transform is very useful when talking about digital systems for which the processing time of the system is not negligible. For instance, a slow computer system can be modeled as being an instantaneous system with an output delay.

The modified Z transform is based off the delayed Z transform:

[Modified Z Transform]

$$X(z,m) = X(z,\Delta)|_{\Delta\to 1-m} = \mathcal{Z}\left\{X(s)e^{-\Delta T s}\right\}|_{\Delta\to 1-m}$$

Star Transform

The Star Transform is a discrete transform that has similarities between the Z transform and the Laplace Transform. In fact, the Star Transform can be said to be nearly analogous to the Z transform, except that the Star transform explicitly accounts for the sampling time of the sampler.

The Star Transform is defined as:

$$F^*(s) = \mathcal{L}^*[f(t)] = \sum_{i=0}^{\infty} f(iT)e^{-siT}$$

[Star Transform]

Star transform pairs can be obtained by plugging $z = e^{sT}$ into the Z-transform pairs, above.

Bilinear Transform

The bilinear transform is used to convert an equation in the Z domain into the arbitrary W domain, with the following properties:

1. roots inside the unit circle in the Z-domain will be mapped to roots on the left-half of the W plane.
2. roots outside the unit circle in the Z-domain will be mapped to roots on the right-half of the W plane
3. roots on the unit circle in the Z-domain will be mapped onto the vertical axis in the W domain.

The bilinear transform can therefore be used to convert a Z-domain equation into a form that can be analyzed using the Routh-Hurwitz criteria. However, it is important to note that the W-domain is not the same as the complex Laplace S-domain. To make the output of the bilinear transform equal to the S-domain, the signal must be prewarped, to account for the non-linear nature of the bilinear transform.

The Bilinear transform can also be used to convert an S-domain system into the Z domain. Again, the input system must be prewarped prior to applying the bilinear transform, or else the results will not be correct.

The Bilinear transform is governed by the folloing variable transformations:

$$z = \frac{(T/2) + w}{(T/2) - w}, \quad w = \frac{2}{T}\frac{z-1}{z+1}$$

[Bilinear Transform]

Where T is the sampling time of the discrete signal.

Frequencies in the w domain are related to frequencies in the s domain through the following relationship:

$$\omega_w = \frac{2}{T}\tan\left(\frac{\omega_s T}{2}\right)$$

This relationship is called the **frequency warping characteristic** of the bilinear transform. To counter-act the effects of frequency warping, we can **pre-warp** the Z-domain equation using the inverse warping charateristic. If the equation is prewarped before it is transformed, the resulting poles of the system will line up more faithfully with those in the s-domain.

$$\omega = \frac{2}{T} \arctan \left(\omega_a \frac{T}{2} \right).$$ **[Bilinear Frequency Prewarping]**

Applying these transformations before applying the bilinear transform actually enables direct conversions between the S-Domain and the Z-Domain. The act of applying one of these frequency warping characteristics to a function before transforming is called **prewarping**.

Wikipedia Resources

- w:Laplace transform
- w:Fourier transform
- w:Z-transform
- w:Star transform
- w:Bilinear transform

System Representations

System Representations

This is a table of times when it is appropriate to use each different type of system representation:

Properties	State-Space Equations	Transfer Function	Transfer Matrix
Linear, Distributed	no	no	no
Linear, Lumped	yes	no	no
Linear, Time-Invariant, Distributed	no	yes	no
Linear, Time-Invariant, Lumped	yes	yes	yes

General Description

General Description	
Time-Invariant, Non-causal	$y(t) = \int_{-\infty}^{\infty} h(t - r)dr$
Time-Invariant, Causal	$y(t) = \int_0^t h(t - r)dr$
Time-Variant, Non-Causal	$y(t) = \int_{-\infty}^{\infty} h(t, r)dr$
Time-Variant, Causal	$y(t) = \int_0^t h(t, r)dr$

State-Space Equations

State-Space Equations	
Time-Invariant	$x'(t) = Ax(t) + Bu(t)$ $y(t) = Cx(t) + Du(t)$
Time-Variant	$x'(t) = A(t)x(t) + B(t)u(t)$ $y(t) = C(t)x(t) + D(t)u(t)$

[Analog State Equations]

State-Space Equations	

[Digital State Equations]

Time-Invariant	$x'[t] = Ax[t] + Bu[t]$ $y[t] = Cx[t] + Du[t]$
Time-Variant	$x'[t] = A[t]x[t] + B[t]u[t]$ $y[t] = C[t]x[t] + D[t]u[t]$

Transfer Functions

Transfer Function
$Y(s) = H(s)X(s)$

[Analog Transfer Function]

Transfer Function
$Y(s) = H(s)X(s)$

[Digital Transfer Function]

Transfer Matrix

Transfer Matrix
$\mathbf{Y}(s) = \mathbf{H}(s)\mathbf{X}(s)$

[Analog Transfer Matrix]

Transfer Matrix
$\mathbf{Y}(z) = \mathbf{H}(z)\mathbf{X}(z)$

[Digital Transfer Matrix]

Matrix Operations

For more about this subject, see:
Linear Algebra
and
Engineering Analysis

Laws of Matrix Algebra

(commutative, distributive, associative)

Conjugate Matrix

Transpose Matrix

Associative Matrix

Determinant

Minors

Cofactors

Rank and Trace

Partitioning

Appendix: MatLab

This page would highly benefit from some screenshots of various systems. Users who have MATLAB or Octave available are highly encouraged to produce some screenshots for the systems here.

MATLAB

MATLAB is a programming language that is specially designed for the manipulation of matricies. Because of it's computational power, MATLAB is a tool of choice for many control engineers to design and simulate control systems. This page is going to discuss using MATLAB for control systems design and analysis.

This page assumes a prior knowledge of the fundamentals of MATLAB. For more information about MATLAB, see MATLAB Programming.

Also, there is an open-source competitor to MATLAB called **Octave**. Octave is similar to MATLAB, but there are also some differences. This page will focus on MATLAB, but another page could be added to focus on Octave. As of Sept 10th, 2006, all the MATLAB commands listed below have been implemented in GNU octave.

This page will use the {{MATLAB CMD}} template to show MATLAB functions that can be used to perform different tasks.

Step Response

First, let's take a look at the classical approach, with the following system:

This operation can be performed using this MATLAB command:
step

$$G(s) = \frac{5s + 10}{s^2 + 4s + 5}$$

This system can effectively be modeled as two vectors of coefficients, NUM and DEN:

```
NUM = [5, 10]
DEN = [1, 4, 5]
```

Now, we can use the MATLAB **step** command to produce the step response to this system:

```
step(NUM, DEN, t);
```

Where t is a time vector. If no results on the left-hand side are supplied by you, the step function will automatically produce a graphical plot of the step response. If, however, you use the following format:

```
[y, x, t] = step(NUM, DEN, t);
```

Then MATLAB will not produce a plot automatically, and you will have to produce one yourself.

Now, let's look at the modern, state-space approach. If we have the matrices A, B, C and D, we can plug these into the step function, as shown:

```
step(A, B, C, D);
```

or, we can optionally include a vector for time, t:

```
step(A, B, C, D, t);
```

Again, if we supply results on the left-hand side of the equation, MATLAB will not automatically produce a plot for us.

If we didn't get an automatic plot, and we want to produce our own, we type:

This operation can be performed using this MATLAB command:
plot

```
[y, x, t] = step(NUM, DEN, t);
```

And then we can create a graph using the **plot** command:

```
plot(t, y);
```

y is the output magnitude of the step response, while x is the internal state of the system from the state-space equations:

$$x' = Ax + Bu$$

$$y = Cx + Du$$

Classical ↔ Modern

MATLAB contains features that can be used to automatically convert to the state-space representation from the Laplace representation. This function, **tf2ss**, is used as follows:

This operation can be performed using this MATLAB command:
tf2ss

```
[A, B, C, D] = tf2ss(NUM, DEN);
```

Where NUM and DEN are the coefficient vectors of the numerator and denominator of the transfer function, respectively.

In a similar vein, we can convert from the Laplace domain back to the state-space representation using the **ss2tf** function, as such:

This operation can be performed using this MATLAB command:

```
[NUM, DEN] = ss2tf(A, B, C, D);
```

ss2tf

Or, if we have more then one input in a vector u, we can write it as follows:

```
[NUM, DEN] = ss2tf(A, B, C, D, u);
```

The u parameter must be provided when our system has more then one input, but it does not need to be provided if we have only 1 input. This form of the equation produces a transfer function for each separate input. NUM and DEN become 2-D matricies, with each row being the coefficients for each different input.

z-Domain Digital Filters

This operation can be performed using this MATLAB command:
filter

Let us now consider a digital system with the following generic transfer function in the Z domain:

$$H(z) = \frac{n(z)}{d(z)}$$

Where n(z) and d(z) are the numerator and denominator polynomials of the transfer function, respectively. The **filter** command can be used to apply an input vector x to the filter. The output, y, can be obtained from the following code:

```
y = filter(n, d, x);
```

The word "filter" may be a bit of a misnomer in this case, but the fact remains that this is the method to apply an input to a digital system. Once we have the output magnitude vector, we can plot it using our plot command:

```
plot(y);
```

This operation can be performed using this MATLAB command:
ones

To get the step response of the digital system, we must first create a step function using the **ones** command:

```
u = ones(1, N);
```

Where N is the number of samples that we want to take in our digital system (not to be confused with "n", our numerator coefficient). Once we have produced our unit step function, we can pass this function through our digital filter as such:

```
y = filter(n, d, u);
```

And we can plot y:

```
plot(y);
```

State-Space Digital Filters

Likewise, we can analyze a digital system in the state-space representation. If we have the following digital state relationship:

$$x[k+1] = Ax[k] + Bu[k]$$

$$y[k] = Cx[k] + Du[k]$$

We can convert automatically to the pulse response using the **ss2tf** function, that we used above:

```
[NUM, DEN] = ss2tf(A, B, C, D);
```

Then, we can filter it with our prepared unit-step sequence vector, u:

```
y = filter(num, den, u)
```

this will give us the step response of the digital system in the state-space representation.

Root Locus Plots

MATLAB supplies a useful, automatic tool for generating the root-locus graph from a transfer function: the **rlocus** command. In the transfer function domain, or the state space domain respectively, we have the following uses of the function:

This operation can be performed using this MATLAB command:
rlocus

```
rlocus(num, den);
```

And:

```
rlocus(A, B, C, D);
```

These functions will automatically produce root-locus graphs of the system. However, if we provide left-hand parameters:

```
[r, K] = rlocus(num, den);
```

Or:

```
[r, K] = rlocus(A, B, C, D);
```

The function won't produce a graph automatically, and you will need to produce one yourself. There is also an optional additional parameter for gain, K, that can be supplied:

```
rlocus(num, den, K);
```

Or:

```
rlocus(A, B, C, D, K);
```

If K is not supplied, MATLAB will supply an automatic gain value for you.

Once we have our values [r, K], we can plot a root locus:

```
plot(r);
```

Digital Root-Locus

Creating a root-locus diagram for a digital system is exactly the same as it is for a continuous system. The only difference is the interpretation of the results, because the stability region for digital systems is different from the stability region for continuous systems. The same **rlocus** function can be used, in the same manner as is used above.

Bode Plots

MATLAB also offers a number of tools for examining the frequency response characteristics of a system, both using bode plots, and using nyquist charts. To construct a bode plot from a transfer function, we use the following command:

This operation can be performed using this MATLAB command:
bode

```
[mag, phase, omega] = bode(NUM, DEN, omega);
```

Or:

```
[mag, phase, omega] = bode(A, B, C, D, u, omega);
```

Where "omega" is the frequency vector where the magnitude and phase response points are analyzed. If we want to convert the magnitude data into decibels, we can use the following conversion:

```
magdb = 20 * log10(mag);
```

This conversion should be known well enough by now that it doesnt require explanation.

When talking about bode plots in decibels, it makes the most sense (and is the most common occurance) to also use a logarithmic frequency scale. To create such a logarithmic sequence in omega, we use the **logspace** command, as such:

This operation can be performed using this MATLAB command:
logspace

```
omega = logspace(a, b, n);
```

This command produces n points, spaced logarithmicly, from 10^a up to 10^b.

If we use the bode command without left-hand arguments, MATLAB will produce a graph of the bode phase and magnitude plots automatically.

Nyquist Plots

In addition to the bode plots, we can create nyquist charts by using the **nyquist** command. The nyquist command operates in a similar manner to the bode command (and other commands that we have used so far):

This operation can be performed using this MATLAB command:
nyquist

```
[real, imag, omega] = nyquist(NUM, DEN, omega);
```

Or:

```
[real, imag, omega] = nyquist(A, B, C, D, u, omega);
```

Here, "real" and "imag" are vectors that contain the real and imaginary parts of each point of the nyquist diagram. If we don't supply the right-hand arguments, the nyquist command automatically produces a nyquist plot for us.

Further Reading

- Ogata, Katsuhiko, "Solving Control Engineering Problems with MATLAB", Prentice Hall, New Jersey, 1994. ISBN 0130459070
- MATLAB Programming.
- http://octave.sourceforge.net/

Glossary
Resources
Licensing

Glossary and List of Equations

The following is a listing of some of the most important terms from the book, along with a short definition or description.

A, B, C

Acceleration Error

The amount of steady state error of the system when stimulated by a unit parabolic input.

Acceleration Error Constant

A system metric that determines that amount of acceleration error in the system.

Adaptive Control

A branch of control theory where controller systems are able to change their response characteristics over time, as the input characteristics to the system change.

Additivity

A system is additive if a sum of inputs results in a sum of outputs.

Analog System

A system that is continuous in time and magnitude.

Block Diagram

A visual way to represent a system that displays individual system components as boxes, and connections between systems as arrows.

Bode Plots

A set of two graphs, a "magnitude" and a "phase" graph, that are both plotted on logscale paper. The magnitude graph is plotted in decibels versus frequency, and the phase graph is plotted in degrees versus frequency. Used to analyze the frequency characteristics of the system.

Bounded Input, Bounded Output

BIBO. If the input to the system is finite, then the output must also be finite. A condition for **stability**.

Causal

A system is causal if the output of the system does not depend on future inputs. All physical systems must be causal.

Classical Approach

See **Classical Controls**.

Classical Controls

A control methodology that uses the transform domain to analyze and manipulate the **Input-Output** characteristics of a system.

Compensator

A Control System that augments the shortcomings of another system.

Condition Number

Continuous-Time

A system or signal that is defined at all points t.

Control System

A system or device that manages the behavior of another system or device.

Controller

See **Control System**.

Convolution

A complex operation on functions defined by the integral of the two functions multiplied together, and time-shifted.

Convolution Integral

The integral form of the convolution operation.

D, E, F

Damping Ratio
    A constant that determines the damping properties of a system.
Digital
    A system that is both **discrete-time**, and **quantized**.
Discrete magnitude
    See **quantized**.
Discrete time
    A system or signal that is only defined at specific points in time.
Distributed
    A system is distributed if it has both an infinite number of states, and an infinite number of state variables. See **Lumped**.
Dynamic
    A system is called dynamic if it doesnt have memory. See **Instantaneous**, **Memory**.

Eigenvalues
    Solutions to the characteristic equation of a matrix.
Eigenvectors
Euler's Formula
    An equation that relates complex exponentials to complex sinusoids.
External Description
    A description of a system that relates the input of the system to the output, without explicitly accounting for the internal states of the system.

Feedback
    The output of the system is passed through some sort of processing unit H, and that result is fed into the plant as an input.
Final Value Theorem
    A theorem that allows the steady-state value of a system to be determined from the transfer function.
Frequency Response
    The response of a system to sinusoids of different frequencies. The Fourier Transform of the impulse response.
Fourier Transform
    An integral transform, similar to the Laplace Transform, that analyzes the frequency characteristics of a system.

G, H, I

Game Theory
    A branch of study that is related to control engineering, and especially **optimal control**. Multiple competing entities, or "players" attempt to minimize their own cost, and maximize the cost of the opponents.
Gain
    A constant multipler in a system that is typically implemented as an amplifier or attenuator. Gain can be changed, but is typically not a function of time.
General Description
    An external description of a system that relates the system output to the system input, the system response, and a time constant through integration.

Hendrik Wade Bode
    Electrical Engineer, did work in control theory and communications. Is primarily remembered in control

engineering for his introduction of the **bode plot**.

Harry Nyquist

Electrical Engineer, did extensive work in controls and information theory. Is remembered in this book primarily for his introduction of the **Nyquist Stability Criterion**.

Homogeniety

A system is homogenious if a scaled input results in an equally scaled output.

Hybrid Systems

Systems which have both analog and digital components.

Impulse

A function denoted δ(t), that is the derivative of the unit step.

Impulse Response

The system output when the system is stimulated by an impulse input. The Inverse Laplace Transform of the transfer function of the system.

Initial Conditions

The conditions of the system at time $t = t_0$, where t_0 is the first time the system is stimulated.

Initial Value Theorem

A theorem that allows the initial conditions of the system to be determined from the Transfer function.

Input-Output Description

See **external description**.

Instantaneous

A system is instantaneous if the system doesnt have memory, and if the current output of the system is only dependant on the current input. See **Dynamic**, **Memory**.

Integrators

A system pole at the origin of the S-plane. Has the effect of integrating the system input.

Inverse Fourier Transform

An integral transform that converts a function from the frequency domain into the time-domain.

Inverse Laplace Transform

An integral transform that converts a function from the S-domain into the time-domain.

Inverse Z-Transform

An integral transform that converts a function from the Z-domain into the discrete time domain.

J, K, L

Laplace Transform

An integral transform that converts a function from the time domain into a complex frequency domain.

Laplace Transform Domain

A complex domain where the Laplace Transform of a function is graphed. The imaginary part of **s** is plotted along the vertical axis, and the real part of **s** is plotted along the horizontal axis.

Left Eigenvectors

Linear

A system that satisfies the **superposition principle**. See **Additive** and **Homogenious**.

Linear Time-Invariant

LTI. See **Linear**, and **Time-Invariant**.

Lumped

A system with a finite number of states, or a finite number of state variables.

M, N, O

Memory

A system has memory if it's current output is dependant on previous and current inputs.

MIMO

A system with multiple inputs and multiple outputs.

Modern Approach
: see **modern controls**

Modern Controls
: A control methodology that uses the state-space representation to analyze and manipulate the **Internal Description** of a system.

Modified Z-Transform
: A version of the Z-Transform, expanded to allow for an arbitrary processing delay.

Natural Frequency
: The fundamental frequency of the system, the frequency for which the system's frequency response is largest.

Negative Feedback
: A feedback system where the output signal is subtracted from the input signal, and the difference is input to the plant.

The Nyquist Criteria
: A necessary and sufficient condition of stability that can be derived from **bode plots**.

Nonlinear Control
: A branch of control engineering that deals exclusively with non-linear systems. We do not cover nonlinear systems in this book.

Oliver Heaviside
: Electrical Engineer, Introduced the Laplace Transform as a tool for control engineering.

Optimal Control
: A branch of control engineering that deals with the minimization of system cost.

Order
: The order of a polynomial is the highest exponent of the independant variable in that exponent. The order of a system is the order of the Transfer Function's denominator polynomial.

Output equation
: An equation that relates the current system input, and the current system state to the current system output.

P, Q, R

Parabolic
: A parabolic input is defined by the equation $\frac{1}{2}t^2u(t)$.

Partial Fraction Expansion
: A method by which a complex fraction is decomposed into a sum of simple fractions.

Percent Overshoot
: PO, the amount by which the step response overshoots the reference value, in percentage of the reference value.

Plant
: A central system which has been provided, and must be analyzed or controlled.

Pole
: A value for s that causes the denominator of the transfer function to become zero, and therefore causes the transfer function itself to approach infinity.

Pole-Zero Form
: The transfer function is factored so that the locations of all the poles and zeros are clearly evident.

Position Error
: The amount of steady-state error of a system stimulated by a unit step input.

Position Error Constant
: A constant that determines the position error of a system.

Positive Feedback
: A feedback system where the system output is added to the system input, and the sum is input into the plant.

Pulse Response
: The response of a digital system to a unit step input, in terms of the transfer matrix.

Ramp
: A ramp is defined by the function $tu(t)$.

Reconstructors
: A system that converts a digital signal into an analog signal.

Reference Value
: The target input value of a feedback system.

Relaxed
: A system is relaxed if the initial conditions are zero.

Rise Time
: The amount of time it takes for the step response of the system to reach within a certain range of the reference value. Typically, this range is 80%.

Robust Control
: A branch of control engineering that deals with systems subject to external and internal noise and disruptions.

Quantized
: A system is quantized if it can only output certain discrete values.

S, T, U, V

Samplers
: A system that converts an analog signal into a digital signal.

Sampled-Data Systems
: See *Hybrid Systems'*.

Sampling Time
: In a discrete system, the sampling time is the amount of time between samples.

S-Domain
: The domain of the Laplace Transform of a signal or system.

Settling Time
: The amount of time it takes for the system's oscillatory response to be damped to within a certain band of the steady-state value. That band is typically 10%

Signal Flow Diagram
: A method of visually representing a system, using arrows to represent the direction of signals in the system.

Star Transform
: A version of the Laplace Transform that acts on discrete signals. This transform is implemented as an infinite sum.

State Equation
: An equation that relates the future states of a system with the current state and the current system input.

State Transition Matrix

State-Space Equations
: A set of equations, typically written in matrix form, that relates the input, the system state, and the output. Consists of the state equation and the ouput equation.

State-Variable
: A vector that describes the internal state of the system.

Stability
: The system output cannot approach infinity as time approaches infinity. See **BIBO**, **Lyapunov Stability**.

Step Response
: The response of a system when stimulated by a unit-step input.

Steady State
: The output value of the system as time approaches infinity.

Steady State Error

At steady state, the amount by which the system output differs from the reference value.

Superposition

A system satisfies the condition of superposition if it is both additive and homogenious.

System Type

The number of ideal integrators in the system.

Time-Invariant

A system is time-invariant if an input time-shifted by an arbitrary delay produces an output shifted by that same delay.

Transfer Function

The ratio of the system output to it's input, in the S-domain. The Laplace Transform of the function's impulse response.

Transfer Function Matrix

The Laplace transform of the state-space equations of a system, that provides an external description of a MIMO system.

Unit Step

An input defined by $u(t)$

Unity Feedback

A feedback system where the feedback loop element H has a transfer function of 1.

Velocity Error

The amount of steady-state error when the system is stimulated by a ramp input.

Velocity Error Constant

A constant that determines that amount of velocity error in a system.

W, X, Y, Z

Zero

A value for s that causes the numerator of the transfer function to become zero, and therefore causes the transfer function itself to become zero.

Zero Input Response

Zero State Response

Z-Transform

An integral transform that is related to the Laplace transform through a change of variables. The Z-Transform is used primarily with digital systems.

List of Equations

The following is a list of the important equations from the text, arranged by subject.

Fundamental Equations

$$e^{j\omega} = \cos(\omega) + j\sin(\omega)$$

[Euler's Formula]

$$(a * b)(t) = \int_{-\infty}^{\infty} a(\tau)b(t-\tau)d\tau$$

[Convolution]

$$\mathcal{L}[f(t) * g(t)] = F(s)G(s)$$
$$\mathcal{L}[f(t)g(t)] = F(s) * G(s)$$

[Convolution Theorem]

$$|A - \lambda I| = 0$$
$$Av = \lambda v$$
$$wA = \lambda w$$

[Characteristic Equation]

$$dB = 20\log(C)$$

[Decibels]

Basic Inputs

$$u(t) = \begin{cases} 0, & t < 0 \\ 1, & t \ge 0 \end{cases}$$

[Unit Step Function]

$$r(t) = tu(t)$$

[Unit Ramp Function]

$$p(t) = \frac{1}{2}t^2u(t)$$

[Unit Parabolic Function]

Error Constants

$$K_p = \lim_{s \to 0} G(s)$$

[Position Error Constant]

$$K_p = \lim_{z \to 1} G(z)$$

$$K_v = \lim_{s \to 0} sG(s)$$
$$K_p = \lim_{z \to 1} (z-1)G(z)$$

[Velocity Error Constant]

$$K_a = \lim_{s \to 0} s^2 G(s)$$
$$K_p = \lim_{z \to 1} (z-1)^2 G(z)$$

[Acceleration Error Constant]

System Descriptions

$$y(t) = \int_{-\infty}^{\infty} g(t,r)x(r)dr$$

[General System Description]

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

[Convolution Description]

$$Y(s) = H(s)X(s)$$

[Transfer Function Description]

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

[State-Space Equations]

$$C[sI - A]^{-1}B + D = \mathbf{H}(s)$$
$$C[zI - A]^{-1}B + D = \mathbf{H}(z)$$

[Transfer Matrix]

$$\mathbf{Y}(s) = \mathbf{H}(s)\mathbf{U}(s)$$
$$\mathbf{Y}(z) = \mathbf{H}(z)\mathbf{U}(z)$$

[Transfer Matrix Description]

$$M = \frac{y_{out}}{y_{in}} = \sum_{k=1}^{N} \frac{M_k \Delta\ _k}{\Delta}$$

[Mason's Rule]

Feedback Loops

$$H_{cl}(s) = \frac{KGp(s)}{1 + KGp(s)Gb(s)}$$

$$H_{ol}(s) = KGp(s)Gb(s)$$

[Open-Loop Transfer Function]

$$F(s) = 1 + H_{ol}$$

[Characteristic Equation]

Transforms

$$F(s) = \mathcal{L}[f(t)] = \int_0^\infty f(t) e^{-st} dt$$

[Laplace Transform]

$$f(t) = \mathcal{L}^{-1}\left\{F(s)\right\} = \frac{1}{2\pi}\int_{c-i\infty}^{c+i\infty} e^{st} F(s)\, ds$$

[Inverse Laplace Transform]

$$F(j\omega) = \mathcal{F}[f(t)] = \int_0^\infty f(t) e^{-j\omega t} dt$$

[Fourier Transform]

$$f(t) = \mathcal{F}^{-1}\left\{F(j\omega)\right\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega) e^{-j\omega t} d\omega$$

[Inverse Fourier Transform]

$$F^*(s) = \mathcal{L}^*[f(t)] = \sum_{i=0}^{\infty} f(iT) e^{-siT}$$

[Star Transform]

$$X(z) = \mathcal{Z}\left\{x[n]\right\} = \sum_{i=-\infty}^{\infty} x[n] z^{-n}$$

[Z Transform]

$$x[n] = Z^{-1}\{X(z)\} = \frac{1}{2\pi j}\oint_C X(z) z^{n-1} dz$$

[Inverse Z Transform]

$$X(z,m) = \mathcal{Z}(x[n],m) = \sum_{n=-\infty}^{\infty} x[n+m-1]z^{-n}$$ **[Modified Z Transform]**

Transform Theorems

$$x(\infty) = \lim_{s\to 0} sX(s)$$
$$x[\infty] = \lim_{z\to 1}(z-1)X(z)$$ **[Final Value Theorem]**

$$x(0) = \lim_{s\to\infty} sX(s)$$ **[Initial Value Theorem]**

Root Locus

$$1 + KG(s)H(s) = 0$$
$$1 + K\overline{GH}(z) = 0$$ **[The Magnitude Equation]**

$$\angle KG(s)H(s) = 180^\circ$$
$$\angle K\overline{GH}(z) = 180^\circ$$ **[The Angle Equation]**

Lyapunov Stability

$$MA + A^TM = -N$$ **[Lyapunov Equation]**

Controllers and Compensators

$$D(s) = K_p + \frac{K_i}{s} + K_d s$$ **[PID]**
$$D(z) = K_p + K_i\frac{T}{2}\left[\frac{z+1}{z-1}\right] + K_d\left[\frac{z-1}{Tz}\right]$$

Resources and Further Reading

Wikibooks

A number of wikibooks exist on topics that are (a) prerequisites to this book (b) companion peices to and references for this book, and (c) of further interest to people who have completed reading this book. Below will be a listing of such books, ordered according to the categories listed above.

Prerequisite Books

- Linear algebra
- Linear Algebra with Differential Equations
- Complex Numbers
- Calculus
- Signals and Systems

Companion Books

- Engineering Analysis
- Engineering Tables
- Analog and Digital Conversion

Books for Further Reading

- Digital Signal Processing
- Communication Systems

Wikiversity

The **Wikiversity** project also contains a number of collaborative learning efforts in the field of control systems, and related subjects. As best as possible, we will attempt to list those efforts here.

- v:Automatic Control Engineering

Software

Root Locus

Root-Locus is a free program that was used to create several of the images in this book. That software can be obtained from the following web address:

- http://www.geocities.com/aseldawy/root_locus.html

MATLAB

MATLAB is copyright *The Mathworks*, with all rights reserved. For more information about MATLAB, or to purchase a copy, visit:

http://www.themathworks.com

All MATLAB code appearing in this book has been released under the terms of the GFDL by the authors.

For further reading about MATLAB, there is a wikibook available:

- MATLAB Programming

Books

The following books were used as reference works in the creation of this wikibook.

- Phillips and Nagle, "Digital Control System Analysis and Design", 3rd Edition, Prentice Hall, 1995. ISBN 013309832X
- Brogan, William L, "Modern Control Theory", 3rd Edition, 1991. ISBN 0135897637
- Dorf and Bishop, "Modern Control Systems", 10th Edition, Prentice Hall, 2005. ISBN 0131277650
- Chen, Chi-Tsong, "Linear System Theory and Design", 3rd Edition, 1999. ISBN 0195117778

Licensing

License

The text of this wikibook is released under the terms of the **GNU Free Documentation License version 1.2**. The particular version of that license that is being used can be found at:

> http://en.wikibooks.org/wiki/Wikibooks:GNU_Free_Documentation_License

The text of that license will also be appended to the end of the **printable version** of this wikibook.

Images used in this document may not be released under the GFDL, and the licenses used with each image in this book will be listed in a table below. Some contributors may cross-license their contributions under the GFDL and another compatable license. Some contributions have been released into the public domain.

Images

The individual images used in this wikibook are released under a variety of different licenses, including the GFDL, and Creative-Commons licenses. Some images have been released into the public domain. The following table will list the images used in this book, along with the license under which the image is released, and any additional information about the images that is needed under the terms of the applicable licenses.

Image and Information	License
Image:Pierre-Simon-Laplace (1749-1827).jpg http://commons.wikimedia.org/wiki/Image:Pierre-Simon-Laplace_(1749-1827).jpg Uploaded by commons:User:Luestling Used on Control Systems/Introduction	Public Domain
Image:Joseph Fourier.jpg http://commons.wikimedia.org/wiki/Image:Joseph_Fourier.jpg Uploaded by commons:User:Rh-Kiriki Used on Control Systems/Introduction	Public Domain
Image:Oliver Heaviside.jpg http://commons.wikimedia.org/wiki/Image:Oliver_Heaviside.jpg Uploaded by commons:User:Bemoeial2 Used on Control Systems/Introduction	Public Domain
Image:System Metrics Diagram.JPG http://en.wikibooks.org/wiki/Image:System_Metrics_Diagram.JPG Uploaded by User:Whiteknight Used on Control Systems/System Metrics	Public Domain
Image:Series-RL.png http://commons.wikimedia.org/wiki/Image:Series-RL.png Uploaded by commons:User:Severino	GFDL

Image:Zeroorderhold.impulseresponse.svg http://commons.wikimedia.org/wiki/Image:Zeroorderhold.impulseresponse.svg Uploaded by commons:User:Rbj	Public Domain
Image:Zeroorderhold.signal.svg http://commons.wikimedia.org/wiki/Image:Zeroorderhold.signal.svg Uploaded by commons:User:Rbj	Public Domain
Image:Predictivefirstorderhold.impulseresponse.svg http://commons.wikimedia.org/wiki/Image:Predictivefirstorderhold.impulseresponse.svg Uploaded by commons:User:Rbj	Public Domain
Image:Predictivefirstorderhold.signal.svg http://commons.wikimedia.org/wiki/Image:Predictivefirstorderhold.signal.svg Uploaded by commons:User:Rbj	Public Domain
Image:Firstorderhold.impulseresponse.svg http://commons.wikimedia.org/wiki/Image:Firstorderhold.impulseresponse.svg Uploaded by commons:User:Rbj	Public Domain
Image:P-controller-symbol-2.svg http://commons.wikimedia.org/wiki/Image:P-controller-symbol-2.svg Uploaded by: commons:User:Netnet	Public Domain
Image:Block diagram.png http://commons.wikimedia.org/wiki/Image:Block_diagram.png Uploaded by: commons:User:Hellisp	Public Domain
Image:Blockdiagrammzustandsraum.PNG http://commons.wikimedia.org/wiki/Image:Blockdiagrammzustandsraum.PNG Uploaded by: commons:User:Ma-Lik	GFDL and Creative Commons Attribution ShareAlike 2.5
Image:Typical State Space model.png http://commons.wikimedia.org/wiki/Image:Typical_State_Space_model.png Uploaded by: en:User:Cburnett	Public Domain
Image:Simple feedback control loop.png http://commons.wikimedia.org/wiki/Image:Simple_feedback_control_loop.png Uploaded by: commons:User:Ikiwaner	GFDL
Image:Bode-pt2.png http://commons.wikimedia.org/wiki/Image:Bode-pt2.png Uploaded by: commons:User:Netnet	Public Domain
Image:Bode-p.png http://commons.wikimedia.org/wiki/Image:Bode-p.png	Public Domain

Uploaded by: commons:User:Netnet	
Image:Bode-i.png http://commons.wikimedia.org/wiki/Image:Bode-i.png Uploaded by: commons:User:Netnet	Public Domain
Image:Bode-d.png http://commons.wikimedia.org/wiki/Image:Bode-d.png Uploaded by: commons:User:Netnet	Public Domain
Image:Bode-pt1.png http://commons.wikimedia.org/wiki/Image:Bode-pt1.png Uploaded by: commons:User:Netnet	Public Domain

Authors

The primary authors of this wikibook are listed below:

- Andrew Whitworth (Whiteknight)

Additional contributors that are not deemed "primary" can be found in the history records of the individual pages on the wikibooks website.

License: GFDL

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

> **A.** Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
> **B.** List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
> **C.** State on the Title page the name of the publisher of the Modified Version, as the publisher.
> **D.** Preserve all the copyright notices of the Document.
> **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
> **F.** Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
> **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
> **H.** Include an unaltered copy of this License.
> **I.** Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from

Retrieved from "http://en.wikibooks.org/wiki/Control_Systems/Print_version"

Categories: Books with print version | Control Systems | Engineering | Control Systems Stub | RenderPNG | Mathematics

- This page was last modified 16:38, 27 October 2006.

- Privacy policy
- About Wikibooks
- Disclaimers