Control Systems

December 26, 2019

Contents

1 Preface

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[1]
- Linear Algebra[2]
- Signals and Systems[3]
- Digital Signal Processing[4]

---

[1] https://en.wikibooks.org/wiki/Calculus
[2] https://en.wikibooks.org/wiki/Linear%20Algebra
[3] https://en.wikibooks.org/wiki/Signals%20and%20Systems
[4] https://en.wikibooks.org/wiki/Digital%20Signal%20Processing

2 Introduction

2.1 This Wikibook

This book was written at **Wikibooks**, a free online community where people write open-content textbooks. Any person with internet access is welcome to participate in the creation and improvement of this book. Because this book is continuously evolving, there are no finite "versions" or "editions" of this book. Permanent links to known good versions of the pages may be provided.

2.2 What are Control Systems?

w:Control system[1] w:Control engineering[2]

The study and design of automatic **Control Systems**, a field known as **control engineering,** has become important in modern technical society. From devices as simple as a toaster or a toilet, to complex machines like space shuttles and power steering, control engineering is a part of our everyday life. This book introduces the field of control engineering and explores some of the more advanced topics in the field. Note, however, that control engineering is a very large field, and this book serves as a foundation of control engineering and introduction to selected advanced topics in the field. Topics in this book are added at the discretion of the authors, and represent the available expertise of our contributors.

Control systems are components that are added to other components, to increase functionality, or to meet a set of design criteria. For 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.

This simple example can be complex to both users and designers of the motor system. It may seem obvious that the motor should start at a higher voltage, so that it accelerates faster. Then we can reduce the supply back down to 10 volts once it reaches ideal speed.

This is clearly a simplistic example, but it illustrates an important point: we can add special "Controller units" to preexisting systems, to improve performance and meet new system specifications.

---

1   https://en.wikipedia.org/wiki/Control%20system
2   https://en.wikipedia.org/wiki/Control%20engineering

Here are some formal definitions of terms 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.

Control System is a conceptual framework for designing systems with capabilities of regulation and/or tracking to give a desired performance. For this there must be a set of signals measurable to know the performance, another set of signals measurable to influence the evolution of the system in time and a third set which is not measurable but disturb the evolution.

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.

There are essentially two methods to approach the problem of designing a new control system: the **Classical Approach**, and the **Modern Approach**.

2.3 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 than the techniques labeled "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 its 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 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. 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.

2.4 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 than 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.

2.5 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:

Algebra[3]

Calculus[4]

The reader should have a good understanding of differentiation and integration. Partial differentiation, multiple integration, and functions of multiple variables will be used occasionally, but the students are not necessarily required to know those subjects well. These advanced calculus topics could better be treated as a co-requisite instead of a pre-requisite.

Linear Algebra[5]

State-space system representation draws heavily on linear algebra techniques. Students should know how to operate on matrices. Students should understand basic matrix operations (addition, multiplication, determinant, inverse, transpose). Students would also benefit from a prior understanding of Eigenvalues and Eigenvectors, but those subjects are covered in this text.

Ordinary Differential Equations[6]

All linear systems can be described by a linear ordinary differential equation. It is beneficial, therefore, for students to understand these equations. Much of this book describes methods to analyze these equations. Students should know what a differential equation is, and they should also know how to find the general solutions of first and second order ODEs.

Engineering Analysis[7]

This book reinforces many of the advanced mathematical concepts used in the Engineering Analysis[8] book, and we will refer to the relevant sections in the aforementioned text for further information on some subjects. This is essentially a math book, but with a focus on various engineering applications. It relies on a previous knowledge of the other math books in this list.

Signals and Systems[9]

The Signals and Systems[10] book will provide a basis in the field of **systems theory**, of which control systems is a subset. Readers who have not read the Signals and Systems[11] book will be at a severe disadvantage when reading this book.

2.6 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

---

3   `https://en.wikibooks.org/wiki/Algebra`
4   `https://en.wikibooks.org/wiki/Calculus`
5   `https://en.wikibooks.org/wiki/Linear%20Algebra`
6   `https://en.wikibooks.org/wiki/Ordinary%20Differential%20Equations`
7   `https://en.wikibooks.org/wiki/Engineering%20Analysis`
8   `https://en.wikibooks.org/wiki/Engineering%20Analysis`
9   `https://en.wikibooks.org/wiki/Signals%20and%20Systems`
10   `https://en.wikibooks.org/wiki/Signals%20and%20Systems`
11   `https://en.wikibooks.org/wiki/Signals%20and%20Systems`

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 interchangeably (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 contributors.

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.

2.6.1 Versions

This wikibook has been expanded to include multiple versions[12] of its text, differentiated by the material covered, and the order in which the material is presented. Each different version is composed of the chapters of this book, included in a different order. This book covers a wide range of information, so if you don't need all the information that this book has to offer, perhaps one of the other versions would be right for you and your educational needs.

Each separate version has a table of contents outlining the different chapters that are included in that version. Also, each separate version comes complete with a printable version, and some even come with PDF versions as well.

Take a look at the **All Versions Listing Page**[13] to find the version of the book that is right for you and your needs.

2.7 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.

---

12   https://en.wikibooks.org/wiki/Control%20Systems%2FAll%20Versions
13   https://en.wikibooks.org/wiki/Control%20Systems%2FAll%20Versions

Let's review some differential equation basics. Consider the topic of interest from a bank. The amount of interest accrued on a given principal 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 principal), 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:

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

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 principal.

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 than the ODE counterparts. The Modern method instead breaks differential equations into systems of low-order equations, and expresses this system in terms of matrices. 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**[14].

---

14   https://en.wikibooks.org/wiki/Calculus

2.8 History

The field of control systems started essentially in the ancient world. Early civilizations, notably the Greeks and the Arabs were heavily 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 renaissance in Europe. Leonhard Euler (for whom **Euler's Formula** is named) discovered a powerful integral transform, but Pierre-Simon Laplace used the transform (later called the **Laplace Transform**) to solve complex problems in probability theory.

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

Figure 1	**Figure 2**
Pierre-Simon Laplace	Joseph Fourier
1749-1827	1768-1840

The "golden age" of control engineering occurred 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. **Rudolf Kalman** is famous for his work in modern control theory, and an adaptive controller called the **Kalman Filter** was named in his honor. 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.

2.9 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 its 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 definitely cover the first two branches, and will hopefully be expanded to cover some of the later branches, if time allows.

2.10 MATLAB

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

---

[a]   https://en.wikibooks.org/wiki/Control%20Systems%2FMATLAB

MATLAB ® is a programming tool that is commonly used in the field of control engineering. We will discuss MATLAB in specific sections of this book devoted to that purpose. MATLAB will not appear in discussions outside these specific sections, although MATLAB may be used in some example problems. An overview of the use of MATLAB in control engineering can be found in the **appendix** at: Control Systems/MATLAB[15].

For more information on MATLAB in general, see: MATLAB Programming[16].

For more information about properly referencing MATLAB, see:
Resources[a]

---

[a]   https://en.wikibooks.org/wiki/Control%20Systems%2FResources

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.

In the future, this book may be expanded to include information on **Simulink** ®, as well as MATLAB.

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 experience and prior knowledge of contributors.

---

15   https://en.wikibooks.org/wiki/Control%20Systems%2FMATLAB
16   https://en.wikibooks.org/wiki/MATLAB%20Programming

2.11 About Formatting

This book will use some simple conventions throughout.

2.11.1 Mathematical Conventions

Mathematical equations will be labeled 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:

[Inverse Laplace Transform]

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

Equations that are named in this manner will also be copied into the List of Equations Glossary[17] in the end of the book, for an easy reference.

Italics will be used for English variables, functions, and equations that appear in the main text. For example *e*, *j*, *f(t)* and *X(s)* are all italicized. Wikibooks contains a LaTeX mathematics formatting engine, although an attempt will be made not to employ formatted mathematical equations inline with other text because of the difference in size and font. Greek letters, and other non-English characters will not be italicized in the text unless they appear in the midst of multiple variables which are italicized (as a convenience to the editor).

Scalar time-domain functions and variables will be denoted with lower-case letters, along with a *t* in parenthesis, such as: *x(t)*, *y(t)*, and *h(t)*. Discrete-time functions will be written in a similar manner, except with an *[n]* instead of a *(t)*.

Fourier, Laplace, Z, and Star transformed functions will be denoted with capital letters followed by the appropriate variable in parenthesis. For example: *F(s)*, *X(jω)*, *Y(z)*, and *F*(s)*.

Matrices will be denoted with capital letters. Matrices which are functions of time will be denoted with a capital letter followed by a *t* in parenthesis. For example: *A(t)* is a matrix, *a(t)* is a scalar function of time.

Transforms of time-variant matrices will be displayed in uppercase bold letters, such as **H***(s)*.

Math equations rendered using LaTeX will appear on separate lines, and will be indented from the rest of the text.

---

17   https://en.wikibooks.org/wiki/Control%20Systems%2FList%20of%20Equations

2.11.2 Text Conventions

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.

⚠ **Warning**

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

3 System Identification

3.1 Systems

Systems, in one sense, are devices that take input and produce an output. A system can be thought to **operate** on the input to produce the output. The output is related to the input by a certain relationship known as the **system response**. The system response usually can be modeled with a mathematical relationship between the system input and the system output.

3.2 System Properties

Physical systems can be divided up into a number of different categories, 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 analysis. Some system classifications are very complex and have still not been investigated with any degree of success. By properly identifying the properties of a system, certain analysis and design tools can be selected for use with the system.

The early sections of 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. This chapter discusses some properties of systems.

Later chapters in this book will look at time variant systems and nonlinear systems. Both time variant and nonlinear systems are very complex areas of current research, and both can be difficult to analyze properly. Unfortunately, most physical real-world systems are time-variant, nonlinear, or both.

An introduction to system identification and least squares techniques can be found here[1]. An introduction to parameter identification techniques can be found here[2].

3.3 Initial Time

The **initial time** of a system is the time before which there is no input. Typically, the initial time of a system is defined to be zero, which will simplify the analysis significantly. Some

---

[1]   http://wikis.controltheorypro.com/index.php?title=Introduction_to_System_
       Identification
[2]   http://wikis.controltheorypro.com/index.php?title=Introduction_to_Parameter_
       Identification

techniques, such as the **Laplace Transform** require that the initial time of the system be zero. The initial time of a system is typically denoted by t_0.

The value of any variable at the initial time t_0 will be denoted with a 0 subscript. For instance, the value of variable x at time t_0 is given by:

$$x(t_0) = x_0$$

Likewise, any time t with a positive subscript are points in time *after* t_0, in ascending order:

$$t_0 \leq t_1 \leq t_2 \leq \cdots \leq t_n$$

So t_1 occurs after t_0, and t_2 occurs after both points. In a similar fashion above, a variable with a positive subscript (unless specifying an index into a vector) also occurs at that point in time:

$$x(t_1) = x_1$$

$$x(t_2) = x_2$$

This is valid for all points in time t.

3.4 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, use the following test:

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

$$y_1 = f(x_1)$$

$$y_2 = f(x_2)$$

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

$$x_3 = x_1 + x_2$$

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

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

Systems that satisfy this property are called **additive**. Additive systems are useful because a sum of simple inputs can be used to analyze the system response to a more complex input.

3.4.1 Example: Sinusoids

Given the following equation:

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

Create a sum of inputs as:

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

and construct the expected sum of outputs:

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

Now, substituting these values into our equation, test for equality:

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

The equality is not satisfied, and therefore the sine operation is not additive.

3.5 Homogeneity

A system satisfies the condition of **homogeneity** 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 **homogeneous**, perform the following test:

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

$$y = f(x)$$

Now, 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, assign x to be equal to x_1:

$$x_1 = x$$

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

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

Systems that are homogeneous are useful in many applications, especially applications with gain or amplification.

3.5.1 Example: Straight-Line

Given the equation for a straight line:

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

$y_1 = f(Cx_1) = 2(Cx_1) + 3 = C2x_1 + 3$

$x_1 = x$

Comparing the two results, it is easy to see they are not equal:

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

Therefore, the equation is not homogeneous.

Exercise:

Prove that additivity implies homogeneity, but that homogeneity does not imply additivity.

3.6 Linearity

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

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:

$$f(Ax_1 + Bx_2) = f(Ax_1) + f(Bx_2) = Af(x_1) + Bf(x_2) = Ay_1 + By_2$$

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

3.6.1 Example: Linear Differential Equations

Is the following equation linear:

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

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

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

Now, create the expected composite output:

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

Substituting the two into our original equation:

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

Factor out the derivative operator, as such:

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

Finally, 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 differential equations typically are linear equations.

3.7 Memory

A system is said to have **memory** if the output from the system is dependent on past inputs (or future inputs!) to the system. A system is called **memoryless** if the output is only dependent 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 **static** systems.

3.8 Causality

Causality is a property that is very similar to memory. A system is called **causal** if it is only dependent on past and/or current inputs. A system is called **anti-causal** if the output of the system is dependent only on future inputs. A system is called **non-causal** if the output depends on past and/or current and future inputs.

A system design that is not causal cannot be physically implemented (to operate in a real-time). If the system can't be built, the design is generally worthless. However, there are applications of non-causal systems, e.g. when a system does not need to operate in a real-time and already has signals stored in its memory (sound and image compression).

3.9 Time-Invariance

A system is called **time-invariant** if the system relationship between the input and output signals is not dependent 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. This facet of time-invariant systems will be discussed later.

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

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

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

Apply a second input x$_1$ to the system, and produce a second output:

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

Now, assign x$_1$ to be equal to the 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:

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

3.10 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 they will be considered almost exclusively in the beginning chapters of this book.

Systems which are not LTI are more common in practice, but are much more difficult to analyze.

3.11 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 that the system can be in.
2. There are a finite number of state variables.

The concept of "states" and "state variables" are relatively advanced, and they will be discussed in more detail in the discussion about **modern controls**.

Systems which are not lumped are called **distributed**. A simple example of a distributed system is a system with delay, that is, $A(s)y(t) = B(s)u(t-\tau)$, which has an infinite number of state variables (Here we use s to denote the Laplace variable). However, although distributed systems are quite common, they are very difficult to analyze in practice, and there are few tools available to work with such systems. Fortunately, in most cases, a delay can be sufficiently modeled with the Pade approximation. This book will not discuss distributed systems much.

3.12 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, i.e., there is no stored energy in the system.

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

In terms of differential equations, a relaxed system is said to have "zero initial state". Systems without an initial state are easier to work with, but systems that are not relaxed can frequently be modified to approximate relaxed systems.

3.13 Stability

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

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 5 volts is applied to the input terminals of a given circuit, it would be best 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**.

There are a number of other types of stability, most of which are based off the concept of BIBO stability. Because stability is such an important and complicated topic, an entire section of this text is devoted to its study.

3.14 Inputs and Outputs

Systems can also be categorized by the number of inputs and the number of outputs the system has. Consider a television as a system, for instance. The system has two inputs:

the power wire and the signal cable. It has one output: the video display. A system with one input and one output is called **single-input, single output**, or SISO. a system with multiple inputs and multiple outputs is called **multi-input, multi-output**, or MIMO.

These systems will be discussed in more detail later.

Exercise:

Based on the definitions of SISO and MIMO, above, determine what the acronyms SIMO and MISO mean.

4 Digital and Analog

4.1 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.

4.1.1 Continuous Time

This operation can be performed using this MATLAB command: `isct`

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. Here is an example of an analog waveform:
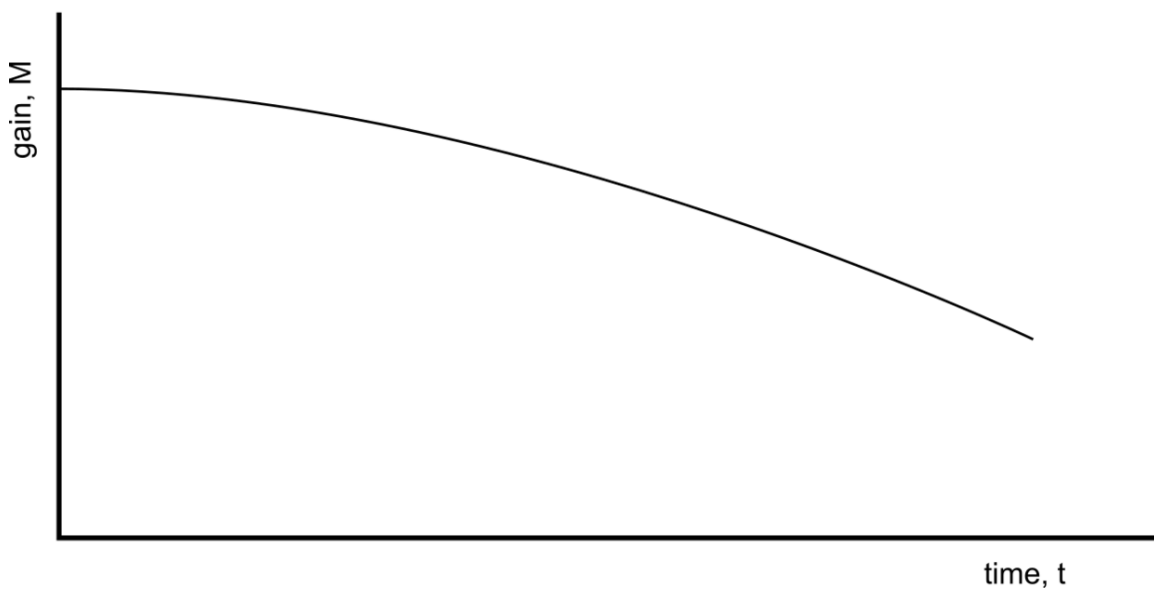
Figure 3

4.1.2 Discrete Time

This operation can be performed using this MATLAB command: `isdt`

A signal is called **discrete-time** if it is only defined for particular points in time. A discrete-time system takes discrete-time input signals, and produces discrete-time output signals. The following image shows the difference between an analog waveform and the sampled discrete time equivalent:
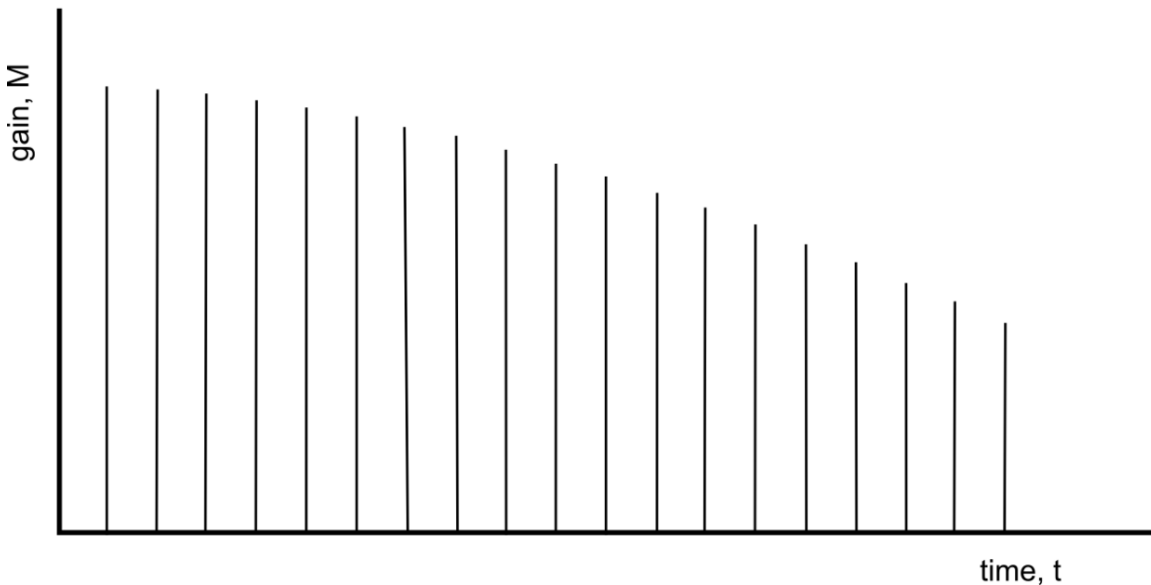
Figure 4

4.1.3 Quantized

A signal is called **Quantized** if it can only be certain values, and cannot be other values. This concept is best illustrated with examples:

1. Students with a strong background in physics will recognize this concept as being the root word in "Quantum Mechanics". In quantum mechanics, it is known that energy comes only in discrete packets. An electron bound to an atom, for example, may occupy one of several discrete energy levels, but not intermediate levels.
2. Another common example is population statistics. For instance, a common statistic is that a household in a particular country may have an average of "3.5 children", or some other fractional number. Actual households may have 3 children, or they may have 4 children, but no household has 3.5 children.
3. People with a computer science background will recognize that integer variables are quantized because they can only hold certain integer values, not fractions or decimal points.

The last example concerning computers is the most relevant, because quantized systems are frequently computer-based. Systems that are implemented with computer software and hardware will typically be quantized.

Here is an example waveform of a quantized signal. Notice how the magnitude of the wave can only take certain values, and that creates a step-like appearance. This image is discrete in magnitude, but is continuous in time:
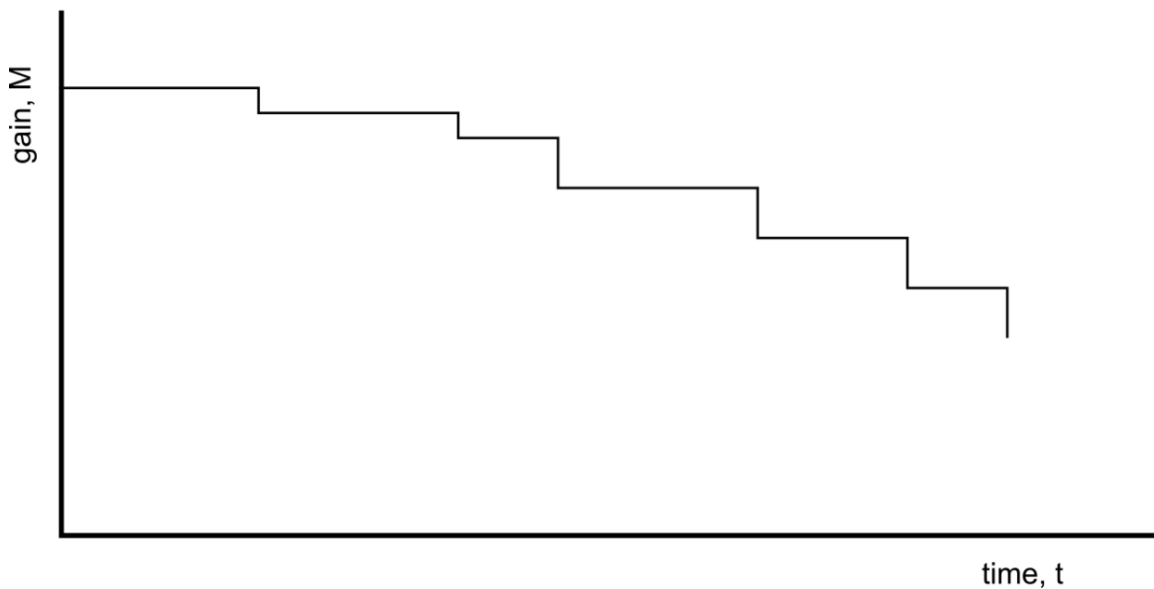
Figure 5

4.2 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 its range.

An analog system is a system that represents data using a direct conversion from one form to another. In other words, an analog system is a system that is continuous in both time and magnitude.

4.2.1 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 voltage that is 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.

4.2.2 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 dependent on functions of time.

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

4.3 Digital

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

Digital

A signal or system is considered digital if it is both 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 than 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. Here is an example of an analog waveform and the digital equivalent. Notice that the digital waveform is discrete in both time and magnitude:
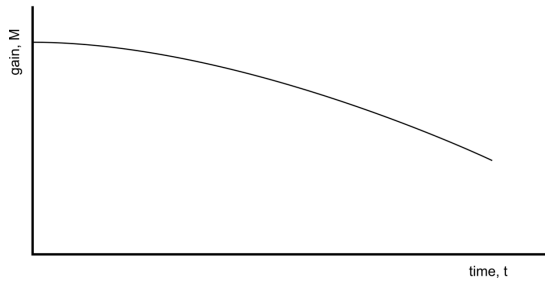
Figure 6

Analog Waveform
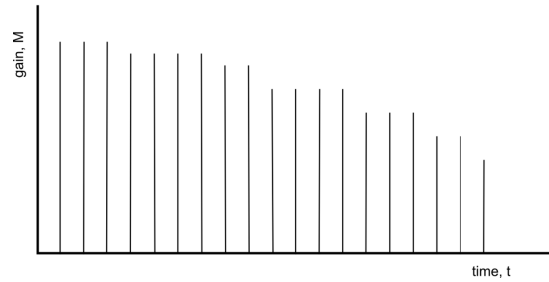
Figure 7

Digital Waveform

4.3.1 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 than that precision.

4.4 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**.

4.4.1 Example: Automobile 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**.

4.5 Continuous and Discrete

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

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

Discrete systems can come in three flavors:

1. Discrete time (sampled)
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 a **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.

4.6 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 than is

available in this book, see the Analog and Digital Conversion[1] wikibook. Here is an example of a reconstructed waveform. Notice that the reconstructed waveform here is quantized because it is constructed from a digital signal:

Figure 8

---

1   https://en.wikibooks.org/wiki/Analog%20and%20Digital%20Conversion
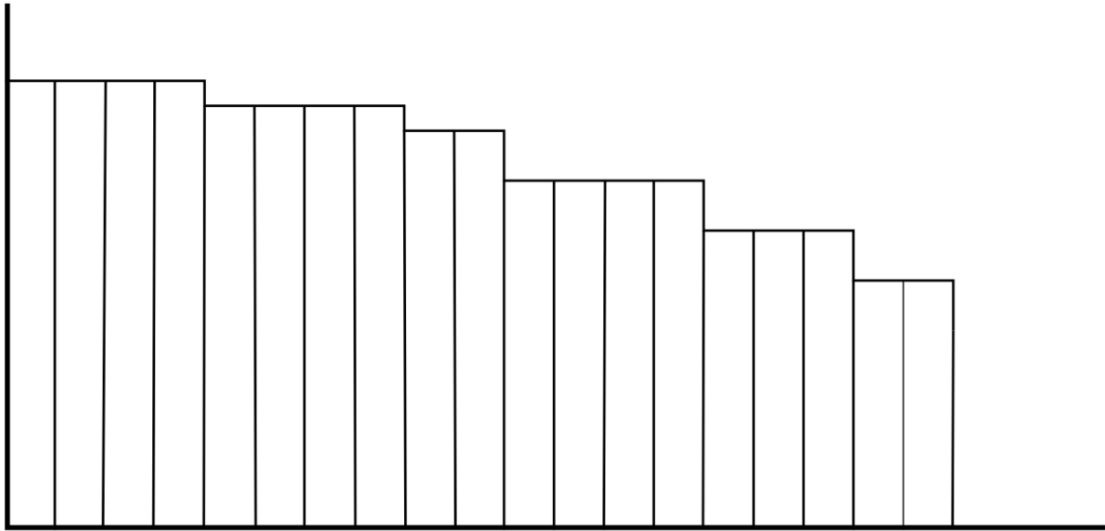
5 System Metrics

5.1 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.

5.2 Standard Inputs

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

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.

Unit Step

A unit step function is defined piecewise as such:

[Unit Step Function]

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

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.
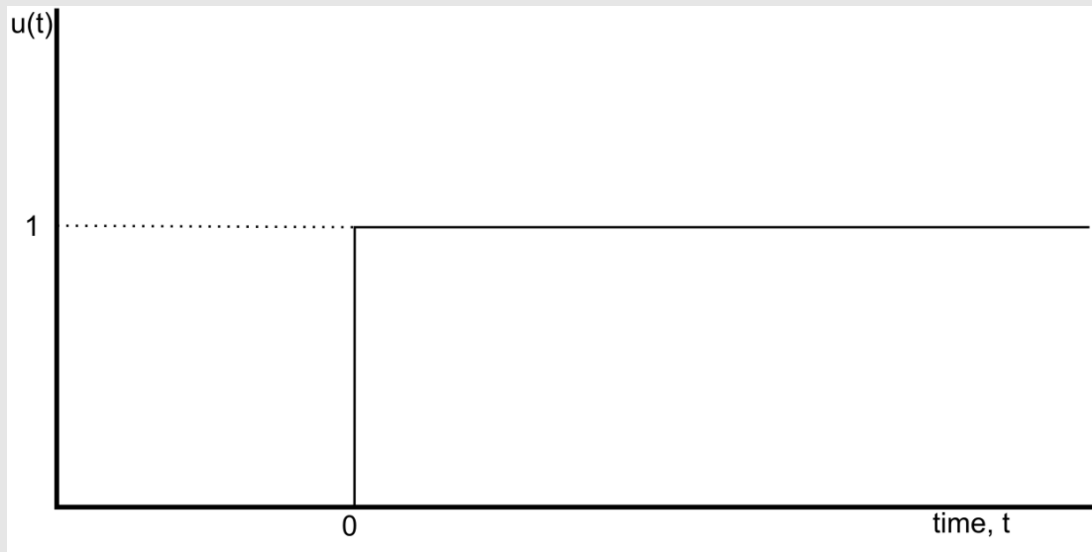
Figure 9

Ramp

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

[Unit Ramp Function]

$$r(t) = tu(t)$$

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

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

This definition will come in handy when we learn about the **Laplace Transform**.
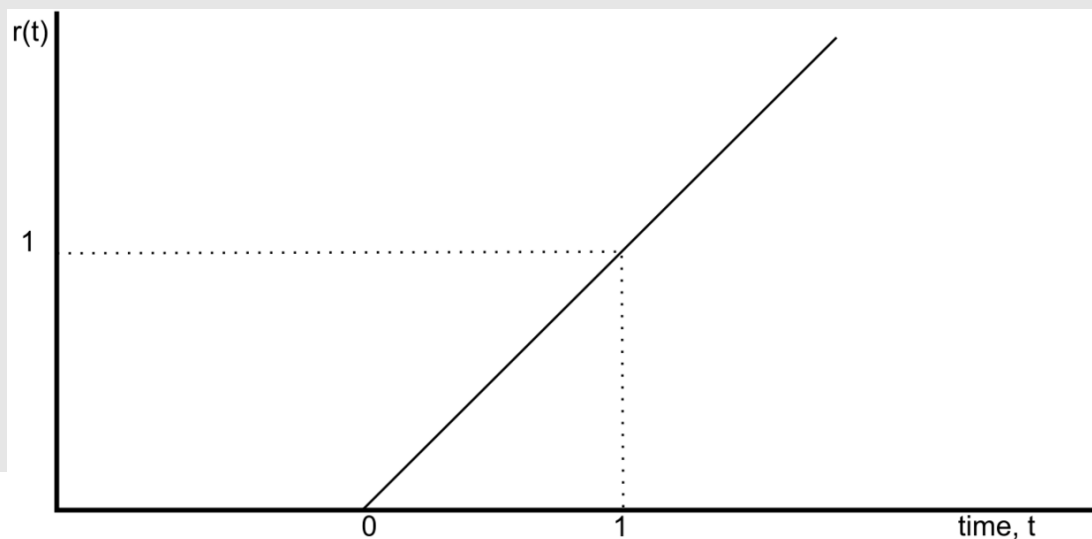
34 **Figure 10**

Parabolic

A unit parabolic input is similar to a ramp input:

[Unit Parabolic Function]

$$p(t) = \frac{1}{2}t^2 u(t)$$

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

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

Again, this result will become important when we learn about the Laplace Transform.
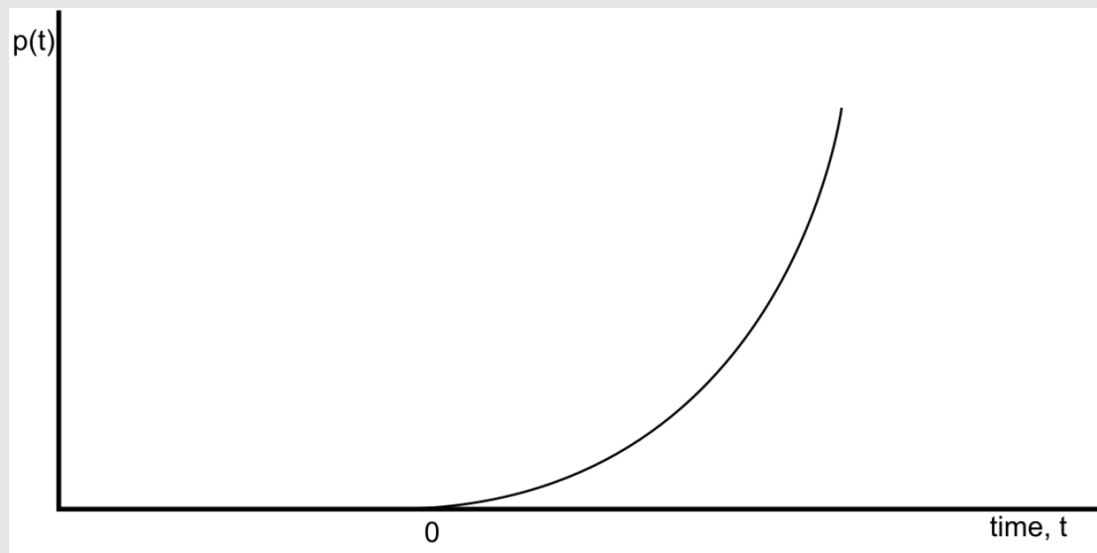
Figure 11

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

5.3 Steady State

Note:
To be more precise, we should have taken the limit as t approaches infinity. However, as a shorthand notation, we will typically say "t equals infinity", and assume the reader understands the shortcut that is being used.

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. Most system responses are **asymptotic**, that is that the response approaches a particular value. Systems that are asymptotic are typically obvious from viewing the graph of that response.

5.3.1 Step Response

The step response of a system is most frequently used to analyze systems, and there is a large amount of terminology involved with step responses. When exposed to the step input, the system will initially have an undesirable output period known as the **transient response**. The transient response occurs because a system is approaching its final output value. The steady-state response of the system is the response after the transient response has ended.

The amount of time it takes for the system output to reach the desired value (before the transient response has ended, typically) is known as the **rise time**. The amount of time it takes for the transient response to end and the steady-state response to begin is known as the **settling time**.

It is common for a systems engineer to try and improve the step response of a system. In general, it is desired for the transient response to be reduced, the rise and settling times to be shorter, and the steady-state to approach a particular desired "reference" output.

Figure 12

An arbitrary step function with

$$x(t) = M u(t)$$

Figure 13

A step response graph of input $x(t)$ to a made-up system
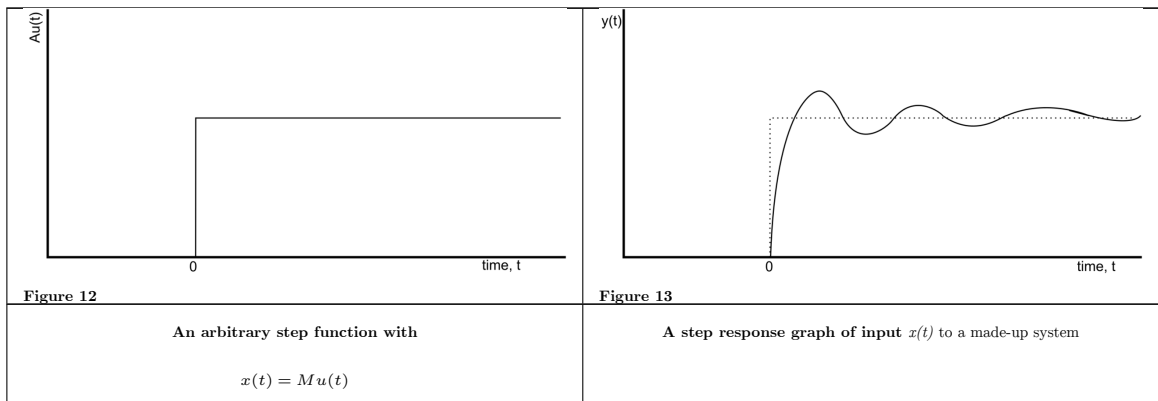
5.4 Target Value

The target output value is the value that our system attempts to obtain for a given input. This is not the same as the steady-state value, which is the actual value that the system 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.

5.5 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 the time it takes to rise between the initial position and 80% of 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. Rise time is typically denoted t_r, or t_{rise}.

Rise time is not the amount of time it takes to achieve steady-state, only the amount of time it takes to reach the desired target value for the first time.

5.6 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. Percent overshoot is typically denoted with the term PO.

Example: Refrigerator

Consider an ordinary household refrigerator. The refrigerator 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 than 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 than it needs to initially, we can say that it "overshoots" the target value by a certain specified amount.

Example: Refrigerator

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 its 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.

5.7 Steady-State Error

Usually, the letter e or E will be used to denote error values.

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.

5.8 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. The acceptable range for settling time is typically determined on a per-problem basis, although common values are 20%, 10%, or 5% of the target value. The settling time will be denoted as t_{s}.

5.9 System Order

The **order** of the system is defined by the number of independent energy storage elements in the system, and intuitively by the highest order of the linear differential equation that describes the system. In a transfer function representation, the order is the highest exponent in the transfer function. In a **proper system**, the system order is defined as the degree of the denominator polynomial. In a state-space equation, the system order is the number of state-variables used in the system. The order of a system will frequently be denoted with an n or N, although these variables are also used for other purposes. This book will make clear distinction on the use of these variables.

5.9.1 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 than (but never equal to) the degree of the numerator polynomial. A **biproper system** is a system where the degree of the denominator polynomial equals 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 a lot of time designing and analyzing imaginary systems.

5.9.2 Example: System Order

1=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 than the numerator, this system is strictly 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.

5.10 System Type

Let's say that we have a process transfer function (or combination of functions, such as a controller feeding in to a process), all in the forward branch of a unity feedback loop. Say that the overall forward branch transfer function is in the following generalized form (known as **pole-zero form**):

[Pole-Zero Form]

$$G(s) = \frac{K \prod_i (s - s_i)}{s^M \prod_j (s - s_j)}$$

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

we call the parameter M 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. System type will generally be denoted with a letter like N, M, or m. Because these variables are typically reused for other purposes, this book will make clear distinction when they are employed.

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 stimulated by a unit step input. We define the position error constant as follows:

[Position Error Constant]

$$K_p = \lim_{s \to 0} G(s)$$

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:

[Velocity Error Constant]

$$K_v = \lim_{s \to 0} sG(s)$$

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:

[Acceleration Error Constant]

$$K_a = \lim_{s \to 0} s^2 G(s)$$

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

Type, M	**Unit System Input**		
	$Au(t)$	$Ar(t)$	$Ap(t)$
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	$e_{ss} = 0$	$e_{ss} = 0$	$e_{ss} = 0$

5.10.1 Z-Domain Type

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

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

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

[Z-Domain Error Constants]

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

5.11 Visually

Here is an image of the various system metrics, acting on a system in response to a step input:
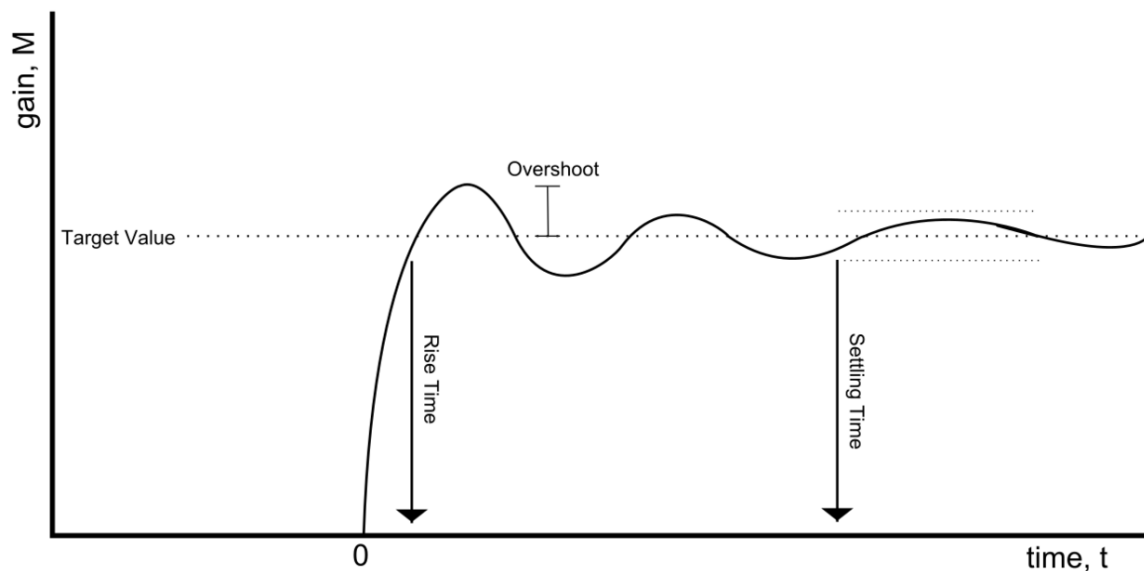
Figure 14

The target value is the value of the input step response. The rise time is the time at which the waveform first reaches the target value. The overshoot is the amount by which the waveform exceeds the target value. The settling time is the time it takes for the system to settle into a particular bounded region. This bounded region is denoted with two short dotted lines above and below the target value.

6 System Modeling

6.1 The Control Process

It is the job of a control engineer to analyze existing systems, and to design new systems to meet specific needs. Sometimes new systems need to be designed, but more frequently a controller unit needs to be designed to improve the performance of existing systems. 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.

6.2 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.
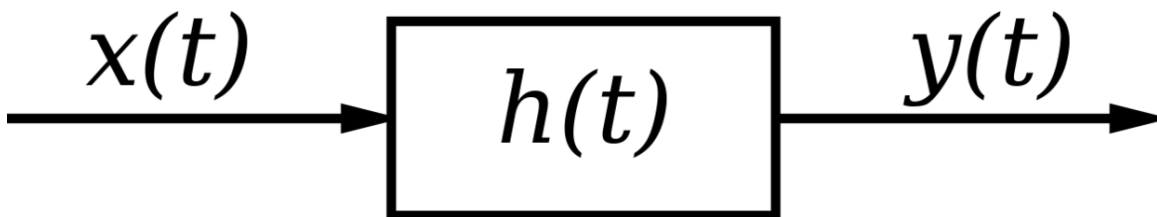
Figure 15

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:

[General System Description]

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

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

If a system is causal (i.e. an input at $t=r$ affects system behaviour only for $t \geq r$) and there is no input of the system before $t=0$, we can change the limits of the integration:

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

6.2.1 Time-Invariant Systems

If furthermore a system is time-invariant, 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.

6.3 Internal Description

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

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

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

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

6.4 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.

6.5 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.

6.6 Analysis

Once a system is modeled using one of the representations listed above, the system needs to be analyzed. We can determine the system metrics and then we can compare those metrics to our specification. If our system meets the specifications we are finished with the design process. However 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.

6.6.1 Frequency Domain

For proposals, early stage designs, and quick turn around analyses a frequency domain model is often superior to a time domain model. Frequency domain models take disturbance PSDs (Power Spectral Densities) directly, use transfer functions directly, and produce output or residual PSDs directly. The answer is a steady-state response. Oftentimes the controller is shooting for 0 so the steady-state response is also the residual error that will be the analysis output or metric for report.

Table 1: Frequency Domain Model Inputs and Outputs

Input	Model	Output

PSD	Transfer Function	PSD

Brief Overview of the Math

Frequency domain modeling is a matter of determining the impulse response of a system to a random process.
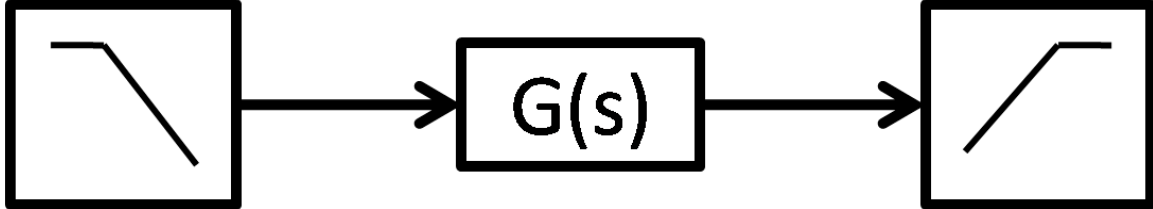
Figure 16   Figure 1: Frequency Domain System

$$S_{YY}(\omega) = G^*(\omega) G(\omega) S_{XX} = |G(\omega)| S_{XX}{}^1$$

where

$S_{XX}(\omega)$ is the one-sided input PSD in $\frac{magnitude^2}{Hz}$

$G(\omega)$ is the frequency response function of the system and

$S_{YY}(\omega)$ is the one-sided output PSD or auto power spectral density function.

The frequency response function, $G(\omega)$, is related to the impulse response function (transfer function) by

$$g(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\omega t} H(\omega)\, d\omega$$

Note some texts will state that this is only valid for random processes which are stationary. Other texts suggest stationary and ergodic while still others state weakly stationary processes. Some texts do not distinguish between strictly stationary and weakly stationary. From practice, the rule of thumb is if the PSD of the input process is the same from hour to hour and day to day then the input PSD can be used and the above equation is valid.

Notes

See a full explanation with example at ControlTheoryPro.com[2]

---

1   Sun, Jian-Qiao (2006). *Stochastic Dynamics and Control, Volume 4*. Amsterdam: Elsevier Science. 0444522301
.

2   http://wikis.controltheorypro.com/index.php?title=Frequency_Domain_Modeling

6.7 Modeling Examples

Modeling in Control Systems is oftentimes a matter of judgement. This judgement is developed by creating models and learning from other people's models. ControlTheoryPro.com is a site with a lot of examples. Here are links to a few of them

- Hovering Helicopter Example[3]
- Reaction Torque Cancellation Example[4]
- List of all examples at ControlTheoryPro.com[5]

6.8 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.

---

3   http://wikis.controltheorypro.com/index.php?title=Helicopter_Hover_Example
4   http://wikis.controltheorypro.com/index.php?title=Reaction_Cancellation_Example
5   http://wikis.controltheorypro.com/index.php?title=Category:Examples

7 Transforms

7.1 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[1] has lots of information on these subjects.

7.1.1 Transform Basics

A **transform** is a mathematical tool that converts an equation from one variable (or one set of variables) into a new variable (or a new set of variables). To do this, the transform must remove all instances of the first variable, the "Domain Variable", and add a new "Range Variable". Integrals are excellent choices for transforms, because the limits of the definite integral will be substituted into the domain variable, and all instances of that variable will be removed from the equation. An integral transform that converts from a domain variable a to a range variable b will typically be formatted as such:

$$\mathcal{T}[f(a)] = F(b) = \int_C f(a)g(a,b)da$$

Where the function $f(a)$ is the function being transformed, and $g(a,b)$ is known as the **kernel** of the transform. Typically, the only difference between the various integral transforms is the kernel.

7.2 Laplace Transform

w:Laplace transform[2]

This operation can be performed using this MATLAB command: `laplace`

---

1   https://en.wikipedia.org/wiki/
2   https://en.wikipedia.org/wiki/Laplace%20transform

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 interchangeably 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.
4. The system or signal in question is causal.

The transform is defined as such:

[Laplace Transform]

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

Laplace transform results have been tabulated extensively. More information on the Laplace transform, including a transform table can be found in **the Appendix**[3].

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:

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

7.2.1 Inverse Laplace Transform

This operation can be performed using this MATLAB command: `ilaplace`

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

[Inverse Laplace Transform]

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

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

---

3   https://en.wikibooks.org/wiki/Control%20Systems%2FTransforms%20Appendix

7.2.2 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:

$$y_1(t) = a_1 x_1(t)$$

$$y_2(t) = a_2 x_2(t)$$

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.

7.2.3 Example: RL Circuit

For more information about electric circuits, see:
Circuit Theory[a]

---
[a]   https://en.wikibooks.org/wiki/Circuit%20Theory

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:

$$V(t) = L\frac{dI(t)}{dt}$$

Where L is a special quantity called the "Inductance" that is a property of inductors.
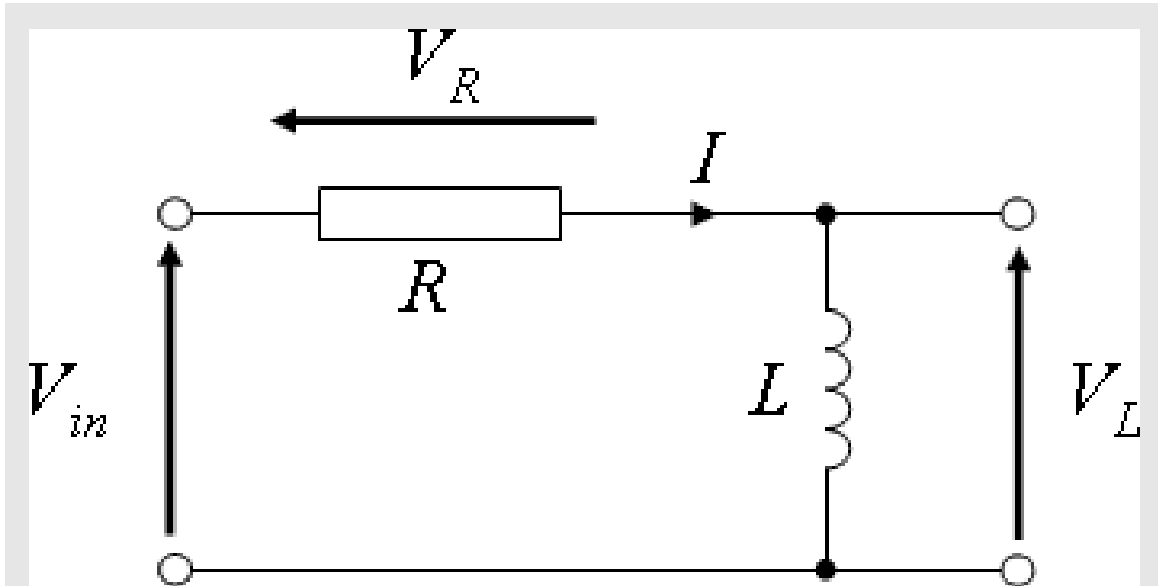
Figure 17 Circuit diagram for the RL circuit example problem. V_L is the voltage over the inductor, and is the quantity we are trying to find.

Let's say that we have a 1st order RL series electric circuit. The resistor has resistance R, the inductor has inductance L, and the voltage source has input voltage V_{in}. The system output of our circuit is the voltage over the inductor, V_{out}. In the time domain, we have the following first-order differential equations to describe the circuit:

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

$$V_{in}(t) = RI(t) + L\frac{dI(t)}{dt}$$

However, since the circuit is essentially acting 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:

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

7.2.4 Partial Fraction Expansion

For more information about Partial Fraction Expansion, see:
Calculus[a]

---

[a]   https://en.wikibooks.org/wiki/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**[4].

7.2.5 Example: Second-Order System

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 $s \rightarrow$ *-2*, the A term disappears, and we are left with $B \rightarrow$ *3*. When we set $s \rightarrow$ *-1*, we can solve for $A \rightarrow$ *-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:

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

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.

---

4   https://en.wikibooks.org/wiki/Calculus

7.2.6 Example: Fourth-Order System

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:

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

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

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

Canceling terms wouldn't be enough here, we will open the brackets (separated onto multiple lines):

$As^3 + 30As^2 + 300As + 1000A + Bs+$

$Cs^2 + 10Cs + Ds^3 + 20Ds^2 + 100Ds$

$= 79s^2 + 916s + 1000$

Let's compare coefficients:

$A + D = 0$

$30A + C + 20D = 79$

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

$1000A = 1000$

And solving gives us:

$A = 1$

$B = 26$

$C = 69$

$D = -1$

We know from the Laplace Transform table that the following relation holds:

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

We can plug in our values for A, B, C, and D into our expansion, and try to convert it into the form above.

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

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

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

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

7.2.7 Example: Complex Roots

Given the following transfer function:

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

When the solution of the denominator is a complex number, we use a complex representation $A + iB$, like $3+i4$ as opposed 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 $40 + j9$ and $40 - j9$.

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

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:

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

7.2.8 Example: Sixth-Order System

Given the following transfer function:

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

We multiply through by the denominators to make the equation rational:

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

$= 90s^2 - 1110$

And then we combine terms:

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

$= 90s^2 - 1110$

Comparing coefficients:

A + B + C = 0

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

73A + 37B - 3D = 0

-111A = -1110

Now, we can solve for A, B, C and D:

A = 10

B = -10

C = 0

D = 120

And now for the "fitting":

The roots of s^2 - 12s + 37 are *6 + j* and *6 - j*

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

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.

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

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

7.2.9 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. The real part of the poles of the function must be <0.

[Final Value Theorem (Laplace)]

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

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.

7.2.10 Example: Final Value Theorem

Find the final value of the following polynomial:

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

We can apply the **Final Value Theorem**:

$\lim_{s \to 0} s \frac{1+s}{1+2s+s^2}$

We obtain the value:

$\lim_{s \to 0} s \frac{1+s}{1+2s+s^2} = 0 \cdot \frac{1+0}{1+2\cdot 0+0^2} = 0 \cdot 1 = 0$

7.2.11 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.

[Initial Value Theorem (Laplace)]

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

7.2.12 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:

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

7.3 Fourier Transform

w:Fourier Transform[5]

The **Fourier Transform** is very similar to the Laplace transform. The fourier transform uses the assumption that any finite time-domain signal 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 its frequency-domain representation, as a function of the radial frequency, ω, The Fourier Transform is defined as such:

[Fourier Transform]

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

This operation can be performed using this MATLAB command: `fourier`

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**[6].

7.3.1 Inverse Fourier Transform

This operation can be performed using this MATLAB command: `ifourier`

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

[Inverse Fourier Transform]

$$f(t) = \mathcal{F}^{-1}\{F(j\omega)\} = \frac{1}{2\pi}\int_{-\infty}^\infty F(j\omega)e^{j\omega t}d\omega$$

---

5   https://en.wikipedia.org/wiki/Fourier%20Transform
6   https://en.wikibooks.org/wiki/Control%20Systems%2FTransforms%20Appendix

This transform is nearly identical to the Fourier Transform.

7.4 Complex Plane
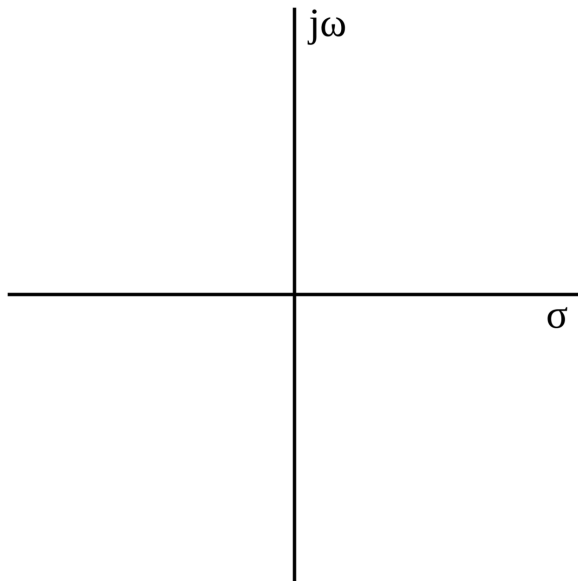
Figure 18

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 independent values, it is frequently of some value to graph the variable s on its own special "S-plane". The S-plane graphs the variable σ on the horizontal axis, and the value of $j\omega$ on the vertical axis. This axis arrangement is shown at right.

7.5 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:

[Euler's Formula]

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

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

7.6 MATLAB

The MATLAB symbolic toolbox contains functions to compute the Laplace and Fourier transforms automatically. The function **laplace**, and the function **fourier** can be used to calculate the Laplace and Fourier transforms of the input functions, respectively. For instance, the code:

```
t = sym('t');
fx = 30*t^2 + 20*t;
laplace(fx)
```

produces the output:

```
ans =

60/s^3+20/s^2
```

We will discuss these functions more in The Appendix[7].

7.7 Further reading

- Digital Signal Processing/Continuous-Time Fourier Transform[8]
- Signals and Systems/Aperiodic Signals[9]
- Circuit Theory/Laplace Transform[10]

---

7   https://en.wikibooks.org/wiki/Control%20Systems%2FMATLAB
8   https://en.wikibooks.org/wiki/Digital%20Signal%20Processing%2FContinuous-Time%20Fourier%20Transform
9   https://en.wikibooks.org/wiki/Signals%20and%20Systems%2FAperiodic%20Signals
10   https://en.wikibooks.org/wiki/Circuit%20Theory%2FLaplace%20Transform

8 Transfer Functions

8.1 Transfer Functions

This operation can be performed using this MATLAB command: `tf`

A **Transfer Function** is the ratio of the output of a system to the input of a system, in the Laplace domain considering its initial conditions and equilibrium point to be zero. This assumption is relaxed for systems observing transience. If we have an input function of $X(s)$, and an output function $Y(s)$, we define the transfer function $H(s)$ to be:

[Transfer Function]

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

Readers who have read the Circuit Theory[1] book will recognize the transfer function as being the impedance, admittance, impedance ratio of a voltage divider or the admittance ratio of a current divider.
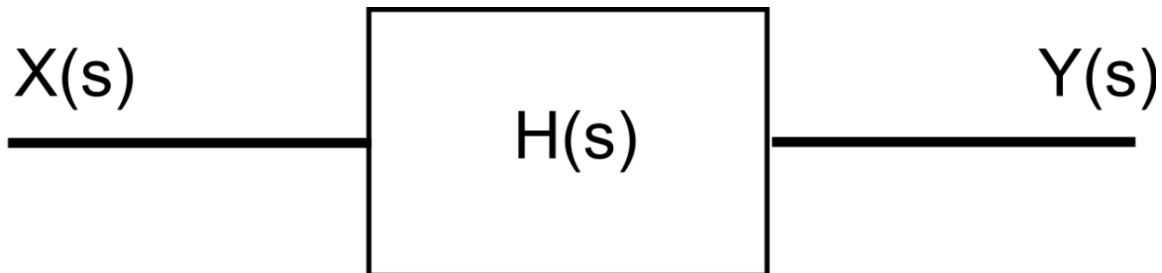
Figure 19

8.2 Impulse Response

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.

---

1   https://en.wikibooks.org/wiki/Circuit%20Theory

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**, $h(t)$.

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

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

8.2.1 Impulse Function

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

[Impulse Function]

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

The impulse function is also known as the **delta function** because it's denoted with the Greek lower-case letter δ. The delta function is typically graphed as an arrow towards infinity, as shown below:
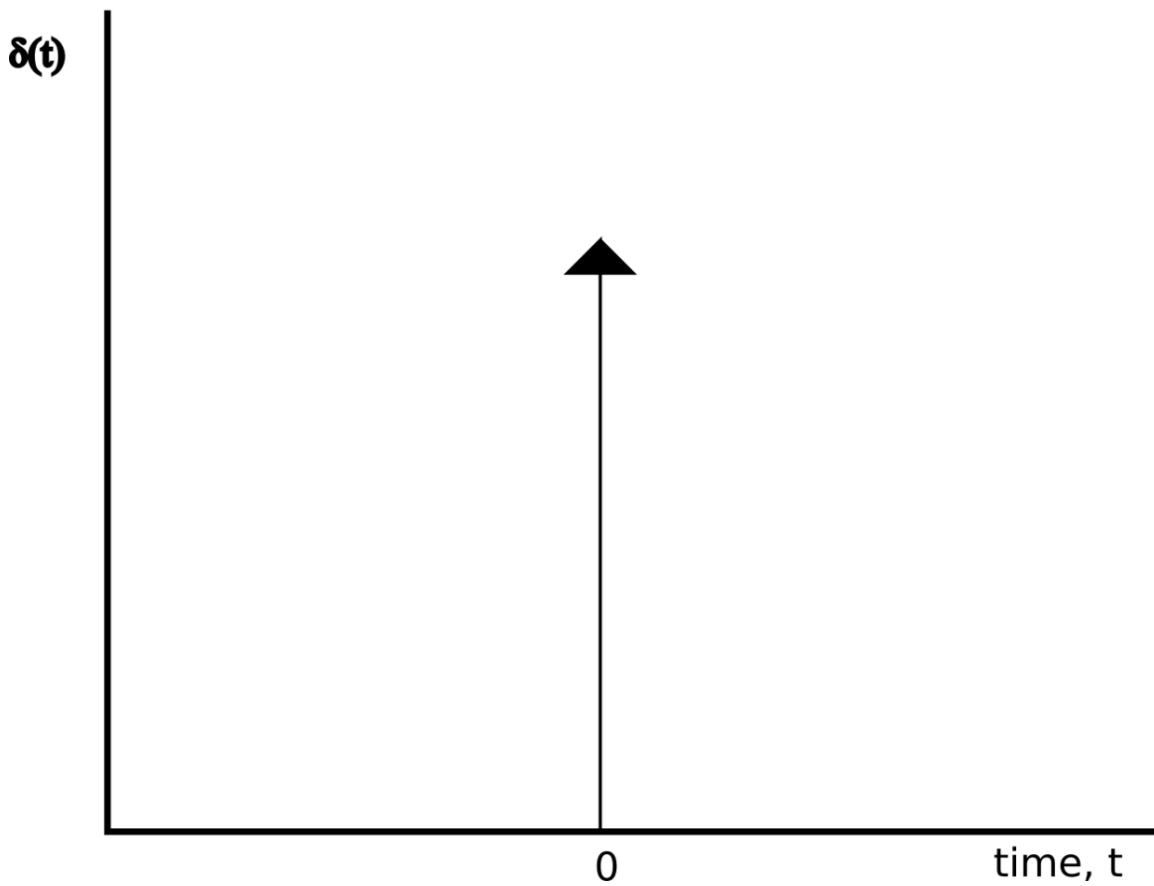
Figure 20

It is drawn as an arrow because it is difficult to show a single point at infinity in any other graphing method. Notice how the arrow only exists at location 0, and does not exist for any other time *t*. The delta function works with regular time shifts just like any other function. For instance, we can graph the function *δ(t - N)* by shifting the function *δ(t)* to the right, as such:
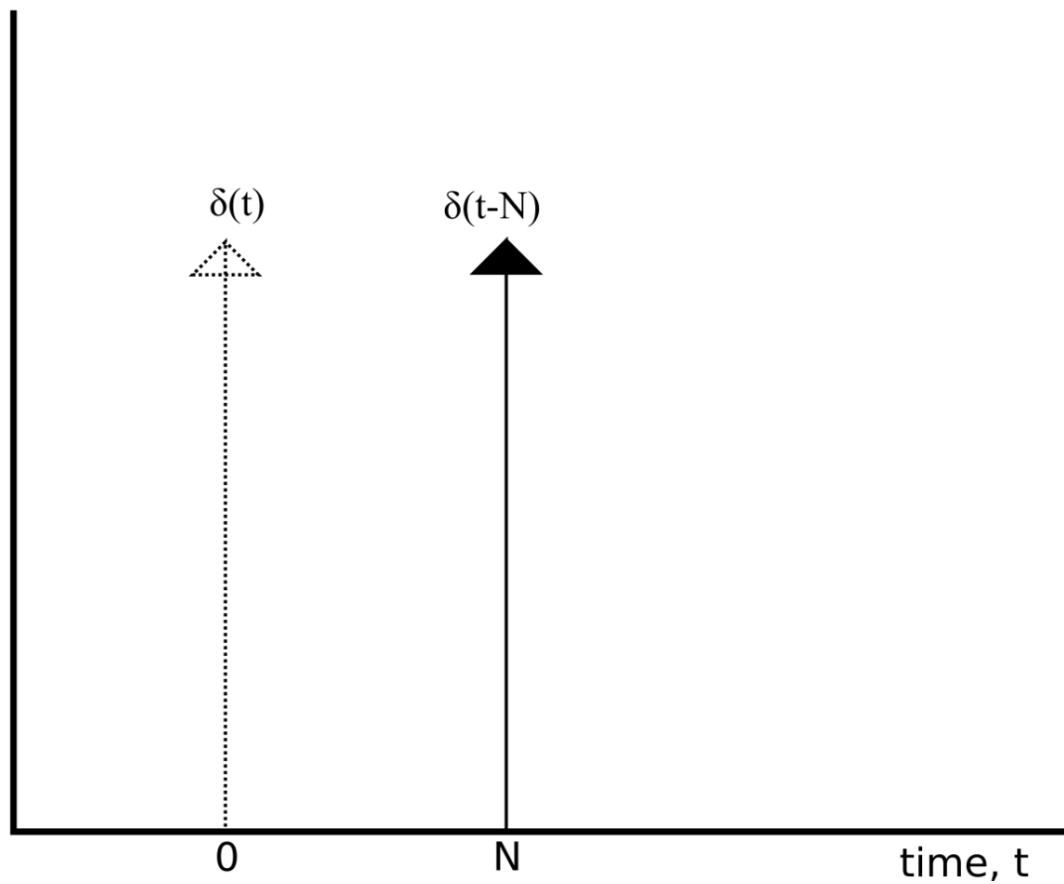
Figure 21

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 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**[2].

8.2.2 Step Response

This operation can be performed using this MATLAB command: `step`[3]

Similar to the impulse response, the **step response** of a system is the output of the system when a unit step function is used as the input. The step response is a common analysis tool used to determine certain metrics about a system. Typically, when a new system is designed, the step response of the system is the first characteristic of the system to be analyzed.

8.3 Convolution

This operation can be performed using this MATLAB command: `conv`

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:

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

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

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:

[Convolution]

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

---

2   https://en.wikibooks.org/wiki/Control%20Systems%2FTransforms%20Appendix%23Table%20of%20Laplace%20Transforms

(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**.

8.3.1 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:

[Convolution Description]

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

8.4 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:

[Convolution Theorem]

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

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

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

8.5 Using the Transfer Function

The Transfer Function fully describes 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, 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 its 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:

[Transfer Function Description]

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

8.5.1 Example: Impulse Response

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

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

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

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

$Y(s) = (1)H(s)$

$Y(s) = H(s)$

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

8.5.2 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:

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

$Y(s) = \frac{1}{s}H(s)$

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

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

8.5.3 Example: MATLAB Step Response

Use MATLAB to find the step response of the following transfer function:

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

We can separate out our numerator and denominator polynomials as such:

num = [79 916 1000]; den = [1 30 300 1000 0]; sys = tf(num, den);

% if you are using the System Identification Toolbox instead of the Control System Tooolbox: sys = idtf(num, den);

Now, we can get our step response from the **step** function, and plot it for time from 1 to 10 seconds:

T = 1:0.001:10; step(sys, T);

8.6 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 its input, represented in the Fourier Domain.
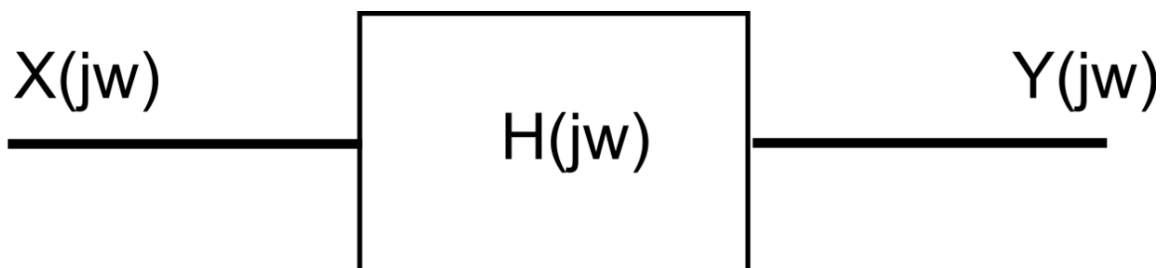
Figure 22

Because the frequency response and the transfer function are so closely related, typically only one is ever calculated, and the other is gained by simple variable substitution. However, despite the close relationship between the two representations, they are both useful individually, and are each used for different purposes.

9 Sampled Data Systems

9.1 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:

$$f^*(t) = f(0)\big(\mathrm{u}(t-0) - \mathrm{u}(t-T)\big) + f(T)\big(\mathrm{u}(t-T) - \mathrm{u}(t-2T)\big) + \cdots + f(nT)\big(\mathrm{u}(t-nT) - \mathrm{u}(t-(n+1)T)\big) +$$

Note that the value of f^* at time $t = 1.5\ T$ is the same as at time $t = T$. 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.

9.2 Geometric Series

w:Geometric progression[1]

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:

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

---

1   https://en.wikipedia.org/wiki/Geometric%20progression

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:

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

We can generalize the sum to this series as follows:

[Geometric Series]

$$\sum_{k=m}^{n} ar^k = \frac{a\left(r^m - r^{n+1}\right)}{1 - r}$$

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:

[Geometric convergence condition]

$$r \leq 1$$

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

9.3 The Star Transform

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

[Star Transform]

$$F^*(s) = \mathcal{L}^*[f(t)] = \sum_{k=0}^{\infty} f(kT)e^{-skT}$$

w:Star transform[2]

The Star Transform depends on the sampling time T and is different for a single signal depending on the frequency 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 convergence**. We will cover this topic more when we discuss the Z-transform.

9.3.1 Star ↔ Laplace

For more information about residues, see:
Complex Analysis/Residue Theory[a]

a   https://en.wikibooks.org/wiki/Complex%20Analysis%2FResidue%20Theory

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:

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

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 than is absolutely necessary in this book. Suffice it to say, however, that the Laplace transform and the Star Transform *are related* mathematically.

2   https://en.wikipedia.org/wiki/Star%20transform

9.3.2 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)*.

9.3.3 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 restrictions on the star transform that allow it to converge are known as the **region of convergence** (ROC) of the transform. Typically a transform must be accompanied by the explicit mention of the ROC.

9.4 The Z-Transform

w:Z-transform[3]

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] ... ]
```

---

3   https://en.wikipedia.org/wiki/Z-transform

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

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

[Z Transform]

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

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

---

[a]   https://en.wikibooks.org/wiki/Control%20Systems%2FTransforms%20Appendix

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

9.4.1 Z Transfer Functions

Like the Laplace Transform, in the Z-domain we can use the input-output relationship of the system to define a **transfer function**.
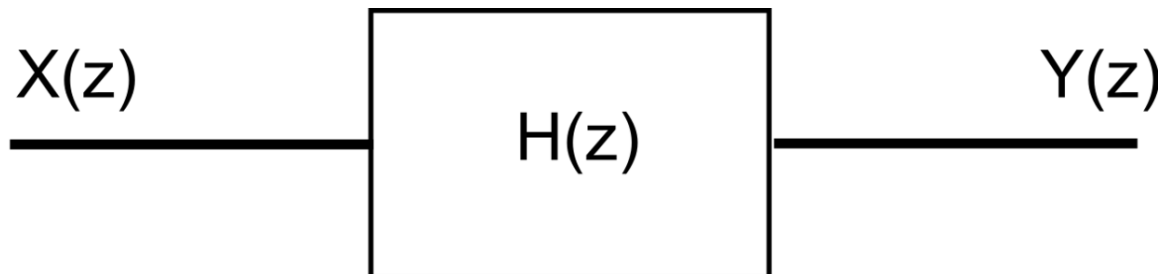
Figure 23

The transfer function in the Z domain operates exactly the same as the transfer function in the S Domain:

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

$$\mathcal{Z}\{h[n]\} = H(z)$$

Similarly, the value $h[n]$ which represents the response of the digital system is known as the **impulse response** of the system. It is important to note, however, that the definition of an "impulse" is different in the analog and digital domains.

9.4.2 Inverse Z Transform

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

[Inverse Z Transform]

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

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 $X(z)$.

There is more information about complex integrals in the book Engineering Analysis[a].

---

[a]   https://en.wikibooks.org/wiki/Engineering%20Analysis

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. There are a number of Z-transform pairs available in table form in **The Appendix**[4].

9.4.3 Final Value Theorem

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

[Final Value Theorem (Z)]

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

This equation can be used to find the steady-state response of a system, and also to calculate the steady-state error of the system.

9.5 Star $\leftrightarrow$ 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-inserted 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.

9.5.1 Z plane

Note:
 The lower-case z is the name of the variable, and the upper-case Z is the name of the Transform and the plane.

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

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

Since z can be broken down into two independent 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-transform 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.

9.5.2 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:

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

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

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

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

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

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

$a = 1$

$r = (e^2 z)^{-1}$

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

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

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

$|(e^2z)^{-1}| = \left|\frac{1}{e^2z}\right| \leq 1$

$|e^2z| \geq 1$

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

$|z| \geq \frac{1}{e^2}$

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:

$$x = A + jB$$

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

9.5.3 Laplace $\leftrightarrow$ 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[5].

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

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:

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

---

5   https://en.wikibooks.org/wiki/Control%20Systems%2FZ%20Transform%20Mappings

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 pieces. 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.

9.5.4 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:

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

9.6 Z $\leftrightarrow$ Fourier

By substituting variables, we can relate the Star transform to the Fourier Transform as well:

$$e^{sT} = e^{j\omega}$$

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

If we assume that $T = 1$, we can relate the two equations together by setting the real part of s to zero. Notice that the relationship between the Laplace and Fourier transforms is mirrored here, where the Fourier transform is the Laplace transform with no real-part to the transform variable.

There are a number of discrete-time variants to the Fourier transform as well, which are not discussed in this book. For more information about these variants, see Digital Signal Processing[6].

9.7 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.

---

6   https://en.wikibooks.org/wiki/Digital%20Signal%20Processing
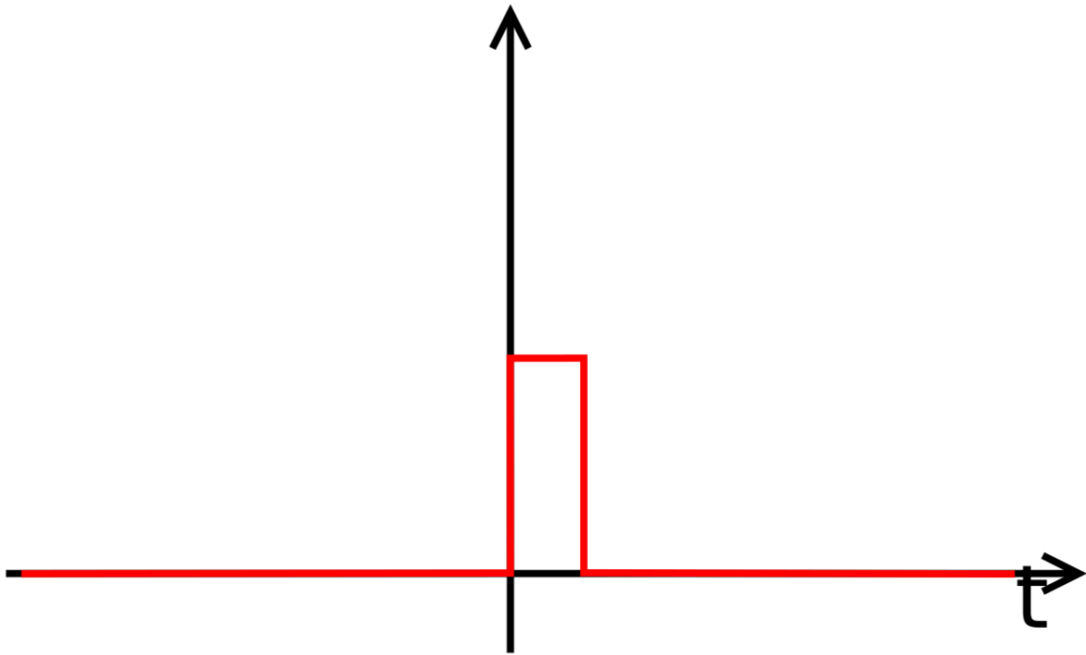
9.7.1 Zero order Hold

Figure 24   Zero-Order Hold impulse response

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]

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

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.
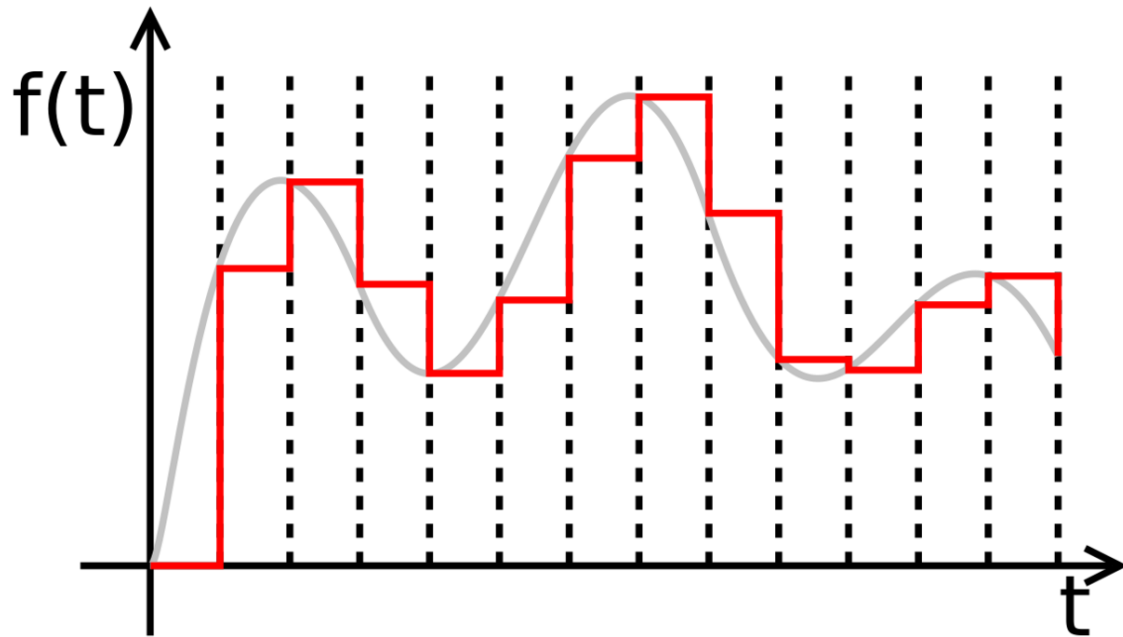
Figure 25 A continuous input signal (gray) and the sampled signal with a zero-order hold (red)

9.7.2 First Order Hold
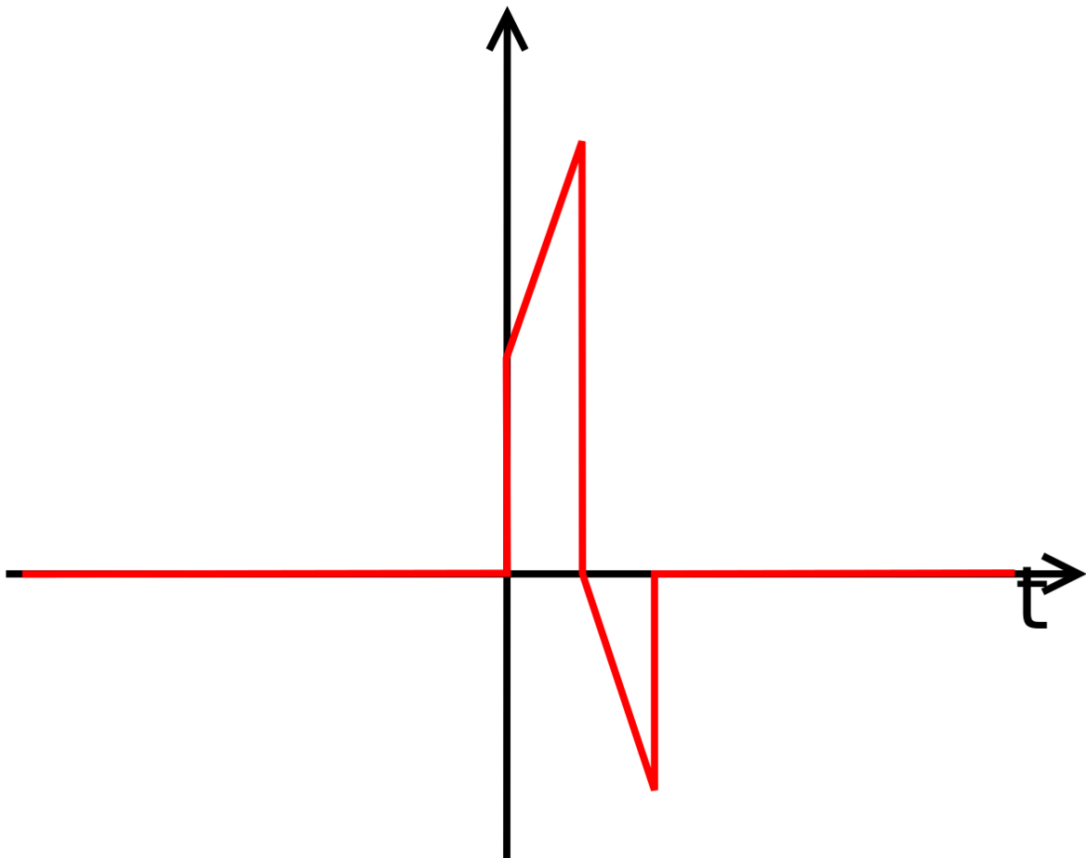
Figure 26 Impulse response of a first-order hold.

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.

[First Order Hold]

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

Keep in mind, however, that the next value of the signal will probably not be the same as the expected value of the next data point, and therefore the first-order hold may have a number of discontinuities.
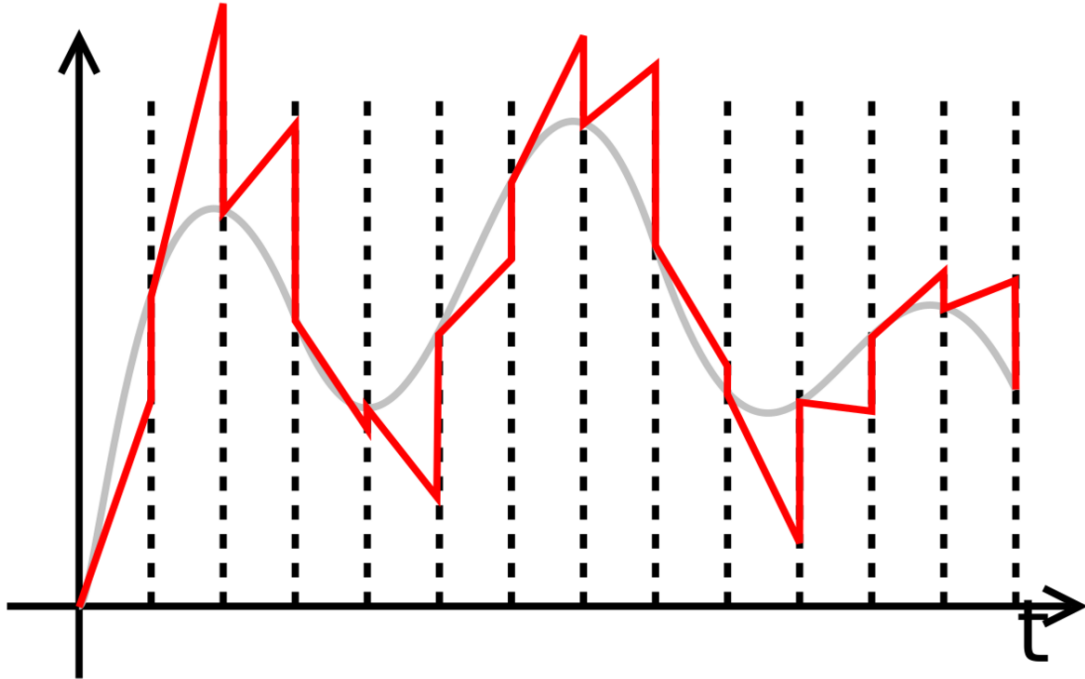
Figure 27   An input signal (grey) and the first-order hold circuit output (red)

9.7.3 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.

[**Fractional Order Hold**]

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

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.

9.7.4 Other Reconstruction Circuits
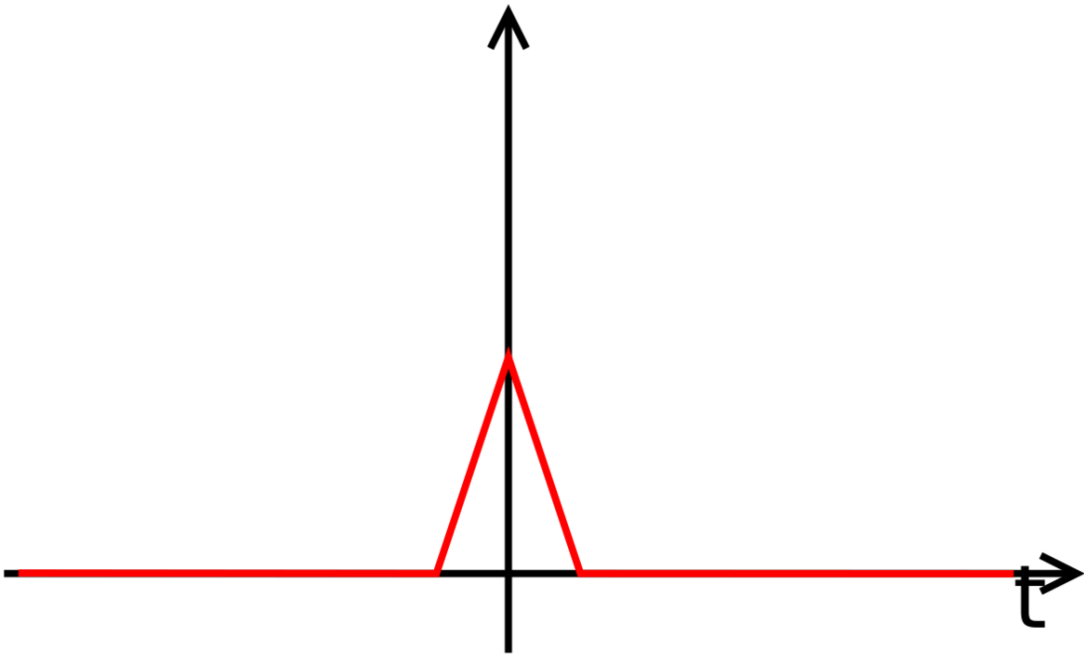
Figure 28 Impulse response to a linear-approximation circuit.

Another type of circuit that can be used is a **linear approximation** circuit.
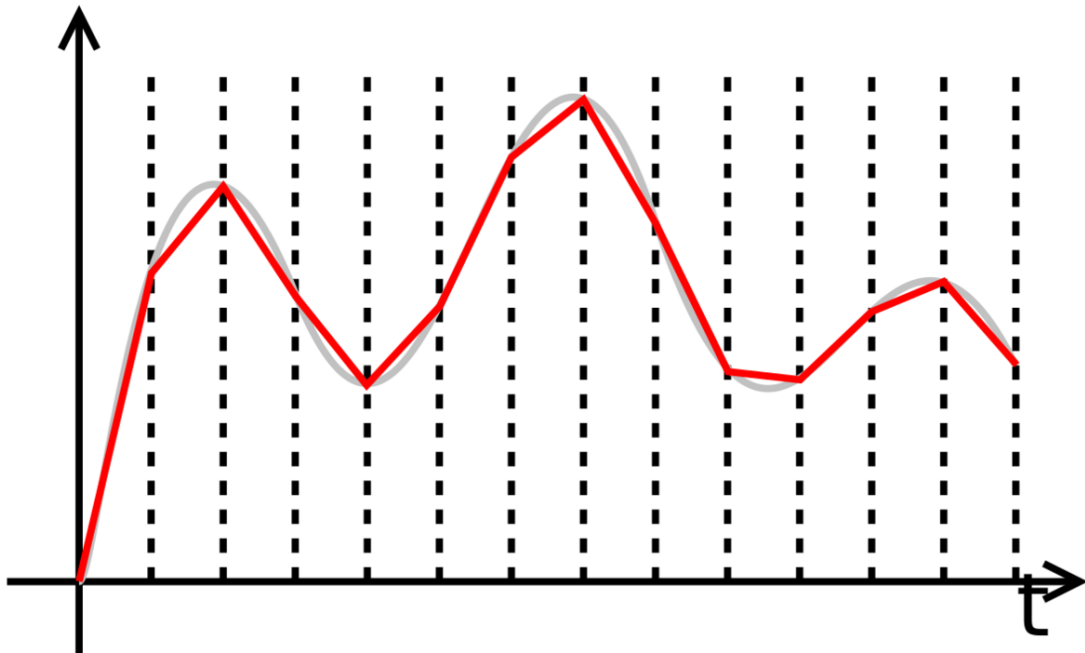
Figure 29   An input signal (grey) and the output signal through a linear approximation circuit

9.8 Further reading

- Hamming, Richard. "Numerical Methods for Scientists and Engineers" 0486652416
- Digital Signal Processing/Z Transform[7]
- Complex Analysis/Residue Theory[8]
- Analog and Digital Conversion[9]

7   https://en.wikibooks.org/wiki/Digital%20Signal%20Processing%2FZ%20Transform
8   https://en.wikibooks.org/wiki/Complex%20Analysis%2FResidue%20Theory
9   https://en.wikibooks.org/wiki/Analog%20and%20Digital%20Conversion

10 System Delays

10.1 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. Once we represent a delay in the Laplace domain, it is an easy matter, through change of variables, to express delays in other domains.

10.1.1 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.

$$Z^{-1}$$
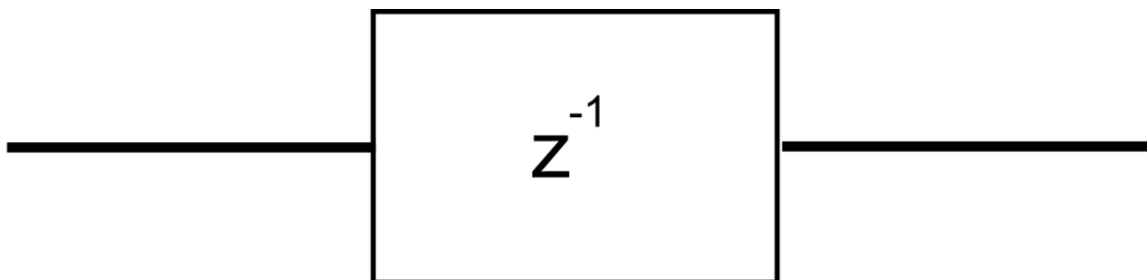
Figure 30

10.2 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.

10.2.1 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:

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

10.3 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.

10.4 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 φ (phi). Phase margin is defined as such for a second-order system:

[Delay Margin]

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

Oftentimes, the phase margin is approximated by the following relationship:

[Delay Margin (approx)]

$$\phi_m \approx 100\zeta$$

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

10.5 Transform-Domain Delays

The ordinary Z-Transform does not account for a system which experiences 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 known as the **Advanced Z-Transform**.

10.5.1 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:

[Delayed Star Transform]

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

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

10.5.2 Delayed Z-Transform

Since we know that the Star Transform 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:

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

And finally:

[Delayed Z Transform]

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

10.6 Modified Z-Transform

w:Advanced Z-transform[1]

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:

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

And it is defined explicitly:

[Modified Z Transform]

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

---

1   https://en.wikipedia.org/wiki/Advanced%20Z-transform

11 Poles and Zeros

11.1 Poles and Zeros

Poles and **Zeros** of a transfer function are the frequencies for which the value of the denominator and numerator of transfer function becomes zero respectively. 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 the most simple sense, can be designed simply by assigning specific values to the poles and zeros of the system.

Physically realizable control systems must have a number of poles greater than the number of zeros. Systems that satisfy this relationship are called **Proper**. We will elaborate on this below.

11.2 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**[1] on the imaginary exponent, we get:

---

1   https://en.wikipedia.org/wiki/Euler%27s%20identity

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

If a complex pole is present it is always accomponied by another pole that is its complex conjugate. The imaginary parts of their time domain representations thus cancel and we are left with 2 of the same real parts. Assuming that the complex conjugate pole of the first term is present, we can take 2 times the real part of this equation and we are left with our final result:

$$2Ae^{\sigma_l t}\cos(\omega_l t)u(t)$$

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

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

11.3 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*.

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

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 than poles, we can state that the polynomial order of *D(s)* must be greater than or equal to the polynomial order of *N(s)*.

11.3.1 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 $s \rightarrow$ *-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 real values.

11.4 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. Addition of poles to the transfer function has the effect of pulling the root locus to the right, making the system less stable. Addition of zeros to the transfer function has the effect of pulling the root locus to the left, making the system more stable.

11.5 Second-Order Systems

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

[Second-order transfer function]

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

Where K is the **system gain**, ζ is called the **damping ratio** of the function, and ω is called the **natural frequency** of the system. ζ and ω, if exactly known for a second order

system, the time responses can be easily plotted and stability can easily be checked. More information on second order systems can be found here[2].

11.5.1 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 slower settling time. Damping is the inherent ability of the system to oppose the oscillatory nature of the system's transient response. Larger values of damping coefficient or damping factor produces transient responses with lesser oscillatory nature.

11.5.2 Natural Frequency

The natural frequency is occasionally written with a subscript:

$$\omega \to \omega_n$$

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. Also, $\omega = \omega_n$ when $\zeta = 0$.

11.6 Higher-Order Systems

---

[2]   http://wikis.controltheorypro.com/index.php?title=Second_Order_Systems

12 State-Space Equations

12.1 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 that:

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

The Laplace transform is transforming the fact that we are dealing with second-order differential equations. The Laplace transform moves a system out of the time-domain into the complex frequency domain, to study and manipulate our systems as algebraic polynomials instead of linear ODEs. Given the complexity of differential equations, why would we ever want to work in the time domain?

It turns out that to decompose our higher-order differential equations into multiple first-order equations, one 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. This demonstrates why the "modern" state-space approach to controls has become popular.

12.2 State-Space

w:State space (controls)[1]

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 system of equations known collectively as **state-space equations**. The state-space is the vector space that consists of all the possible internal states of the system.

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

1. **The system must be "lumped"**

"Lumped" in this context, means that we can find a *finite*-dimensional state-space vector which fully characterises all such internal states of the system.

This text mostly considers linear state space systems, where the state and output equations satisfy the superposition principle and the state space is linear. However, the state-space approach is equally valid for nonlinear systems although some specific methods are not applicable to nonlinear systems.

State

Central to the state-space notation is the idea of a **state**. A state of a system is the current value of internal elements of the system, that change separately (but not completely unrelated) to the output of the system. In essence, the state of a system is an explicit account of the values of the internal system components. Here are some examples:

> Consider an electric circuit with both an input and an output terminal. This circuit may contain any number of inductors and capacitors. The state variables may represent the magnetic and electric fields of the inductors and capacitors, respectively.

> Consider a spring-mass-dashpot system. The state variables may represent the compression of the spring, or the acceleration at the dashpot.

> Consider a chemical reaction where certain reagents are poured into a mixing container, and the output is the amount of the chemical product produced over time. The state variables may represent the amounts of un-reacted chemicals in the container, or other properties such as the quantity of thermal energy in the container (that can serve to facilitate the reaction).

---

1   `https://en.wikipedia.org/wiki/State%20space%20%28controls%29`

12.3 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 independent of one another, and only dependent 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 u, the output variables with y, and the state variables with x. In essence, we have the following relationship:

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

Where $f(x, u)$ 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.

12.4 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. We will discuss MIMO systems in the MIMO Systems Chapter[2].

---

2   Chapter 17 on page 141

12.5 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 $x'(t)$ (note the prime) for the first derivative of the state vector of the system, as dependent 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:

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

$$y(t) = h[t, x(t), u(t)]$$

> **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.

The first equation shows that the system state change is dependent 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 dependent 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 input vectors, then we can say the systems are linear systems, and we can rewrite them in matrix form:

[State Equation]

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

[Output Equation]

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

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

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

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

The **State Equation** shows the relationship between the system's current state and its input, and the future state of the system. The **Output Equation** shows the relationship between the system state and its input, and the output. These equations show that in a

given system, the current output is dependent on the current input and the current state. The future state is also dependent 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 matrices, however, that make certain computations easier. Converting between these forms will require knowledge of linear algebra.

> ;State-Space Basis Theorem:Any system that can be described by a finite number of nth order differential equations or n^{th} order difference equations, or any system that can be approximated 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.

12.5.1 Matrices: A B C D

Our system has the form:

$$\mathbf{x}'(t) = \mathbf{g}[t_0, t, \mathbf{x}(t), x(0), \mathbf{u}(t)]$$

$$\mathbf{y}(t) = \mathbf{h}[t, \mathbf{x}(t), \mathbf{u}(t)]$$

We've bolded several quantities to try and reinforce the fact that they can be vectors, not just scalar quantities. If these systems are time-invariant, we can simplify them by removing the time variables:

$$\mathbf{x}'(t) = \mathbf{g}[\mathbf{x}(t), x(0), \mathbf{u}(t)]$$

$$\mathbf{y}(t) = \mathbf{h}[\mathbf{x}(t), \mathbf{u}(t)]$$

Now, if we take the partial derivatives of these functions with respect to the input and the state vector at time t_0, we get our system matrices:

$$A = \mathbf{g}_x[x(0), x(0), u(0)]$$

$$B = \mathbf{g}_u[x(0), x(0), u(0)]$$

$$C = \mathbf{h}_x[x(0), u(0)]$$

$$D = \mathbf{h}_u[x(0), u(0)]$$

In our time-invariant state space equations, we write these matrices and their relationships as:

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

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

We have four 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 dependent 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 dependent 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 **feed-forward 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 feed-forward element, and therefore for most of the systems we have already considered, the D matrix is the zero matrix.

12.5.2 Matrix Dimensions

Because we are adding and multiplying multiple matrices and vectors together, we need to be absolutely certain that the matrices have compatible 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
$x : p \times 1$ $x' : p \times 1$ $u : q \times 1$ $y : r \times 1$	$A : p \times p$ $B : p \times q$ $C : r \times p$ $D : r \times q$

> **Matrix Dimensions:**
> 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 compatible dimensions or they cannot be combined using matrix operations.

For the rest of the book, we will be using the small template on the right as a reminder about the matrix dimensions, so that we can keep a constant notation throughout the book.

12.5.3 Notational Shorthand

The state equations and the output equations of systems can be expressed in terms of matrices A, B, C, and D. Because the form of these equations is always the same, we can use an ordered quadruplet to denote a system. We can use the shorthand *(A, B, C, D)* to denote a complete state-space representation. Also, because the state equation is very important for our later analyis, we can write an ordered pair *(A, B)* to refer to the state equation:

$$(A, B) \rightarrow x' = Ax + Bu$$

$$(A, B, C, D) \rightarrow \begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$$

12.6 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 text will not make such a distinction. Instead we will opt to use the generic coefficient matrices A, B, C and D for both continuous and discrete systems. Occasionally this book may employ the subscript C to denote a continuous-time version of the matrix, and the subscript D to denote the discrete-time version of the same matrix. 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.

12.6.1 From Differential Equations

Let's say that we have a general 3rd order differential equation in terms of input $u(t)$ and output $y(t)$:

$\frac{d^3 y(t)}{dt^3} + a_2 \frac{d^2 y(t)}{dt^2} + a_1 \frac{dy(t)}{dt} + a_0 y(t) = u(t)$

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

$x_1 = y(t)$

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

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

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

$x_1' = x_2$

$x_2' = x_3$

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

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.

12.6.2 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. We call the process of converting a system description from the Laplace domain to the state-space domain **realization**. We will discuss realization in more detail in a later chapter. 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:

$$
A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -b_0 & -b_1 & -b_2 & \cdots & -b_{n-1} \end{bmatrix}
$$

$$
B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}
$$

$$
C = \begin{bmatrix} a_0 & a_1 & \cdots & a_{m-1} \end{bmatrix}
$$

$$
D = 0
$$

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

Notice that to perform this method, the denominator and numerator polynomials must be *monic*, the coefficients of the highest-order term must be 1. If the coefficient of the highest order term is not 1, you must divide your equation by that coefficient to make it 1.

12.7 State-Space Representation

As an important note, remember that the state variables x are user-defined and therefore are arbitrary. 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 than others. Once these 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

$$
\frac{d}{dt} \left[\frac{d^2 y(t)}{dt^2} + a_2 \frac{dy(t)}{dt} + a_1 y(t) \right] + a_0 y(t) = u(t)
$$

We now define the state variables

$$
x_1 = y(t)
$$

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

$$x_3 = \frac{d^2 y(t)}{dt^2} + a_2 \frac{dy(t)}{dt} + a_1 y(t)$$

with first-order derivatives

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

$x_2' = \frac{d^2 y(t)}{dt^2} = -a_1 x_1 - a_2 x_2 + x_3$ (suspected error here. Fails to account that :$\frac{d}{dt}$ []. encapsulates :$\frac{d^2 y(t)}{dt^2} + a_2 \frac{dy(t)}{dt} + a_1 y(t)$ five lines earlier.)

$$x_3' = -a_0 y(t) + u(t)$$

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.

12.7.1 Example: Dummy Variables

The altitude control of a particular manned aircraft can be given by:

$\theta''(t) = \alpha + \delta$

Where α is the direction the aircraft is traveling in, θ is the direction the aircraft is facing (the attitude), and δ is the angle of the ailerons (the control input from the pilot). This equation is not in a proper format, so we need to produce some dummy-variables:

$\theta_1 = \theta$

$\theta_1' = \theta_2$

$\theta_2' = \alpha + \delta$

This in turn will provide us with our state equation:

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \delta \end{bmatrix}$$

As we can see from this equation, even though we have a valid state-equation, the variables θ_1 and θ_2 don't necessarily correspond to any measurable physical event, but are instead dummy variables constructed by the user to help define the system. Note,

however, that the variables α and δ do correspond to physical values, and cannot be changed.

12.8 Discretization

If we have a system *(A, B, C, D)* that is defined in continuous time, we can **discretize** the system so that an equivalent process can be performed using a digital computer. We can use the definition of the derivative, as such:

$$x'(t) = \lim_{T \to 0} \frac{x(t+T) - x(t)}{T}$$

And substituting this into the state equation with some approximation (and ignoring the limit for now) gives us:

$$\lim_{T \to 0} \frac{x(t+T) - x(t)}{T} = Ax(t) + Bu(t)$$

$$x(t+T) = x(t) + Ax(t)T + Bu(t)T$$

$$x(t+T) = (1+AT)x(t) + (BT)u(t)$$

We are able to remove that limit because in a discrete system, the time interval between samples is positive and non-negligible. By definition, a discrete system is only defined at certain time points, and not at all time points as the limit would have indicated. In a discrete system, we are interested only in the value of the system at discrete points. If those points are evenly spaced by every T seconds (the sampling time), then the samples of the system occur at $t = kT$, where k is an integer. Substituting kT for t into our equation above gives us:

$$x(kT+T) = (1+AT)x(kT) + TBu(kT)$$

Or, using the square-bracket shorthand that we've developed earlier, we can write:

$$x[k+1] = (1+AT)x[k] + TBu[k]$$

In this form, the state-space system can be implemented quite easily into a digital computer system using software, not complicated analog hardware. We will discuss this relationship and digital systems more specifically in a later chapter.

We will write out the discrete-time state-space equations as:

$$x[n+1] = A_d x[n] + B_d u[n]$$

$$y[n] = C_d x[n] + D_d u[n]$$

12.9 Note on Notations

The variable T is a common variable in control systems, especially when talking about the beginning and end points of a continuous-time system, or when discussing the sampling time of a digital system. However, another common use of the letter T is to signify the transpose operation on a matrix. To alleviate this ambiguity, we will denote the transpose of a matrix with a *prime*:

$$A^T \rightarrow A'$$

Where A' is the transpose of matrix A.

The prime notation is also frequently used to denote the time-derivative. Most of the matrices that we will be talking about are time-invariant; there is no ambiguity because we will never take the time derivative of a time-invariant matrix. However, for a time-variant matrix we will use the following notations to distinguish between the time-derivative and the transpose:

$A(t)'$ the transpose.

$A'(t)$ the time-derivative.

Note that certain variables which are time-variant are not written with the *(t)* postscript, such as the variables x, y, and u. For these variables, the default behavior of the prime is the time-derivative, such as in the state equation. If the transpose needs to be taken of one of these vectors, the *(t)'* postfix will be added explicitly to correspond to our notation above.

For instances where we need to use the Hermitian transpose, we will use the notation:

$$A^H$$

This notation is common in other literature, and raises no obvious ambiguities here.

12.10 MATLAB Representation

This operation can be performed using this MATLAB command: `ss`

State-space systems can be represented in MATLAB using the 4 system matrices, A, B, C, and D. We can create a system data structure using the **ss** function:

```
sys = ss(A, B, C, D);
```

Systems created in this way can be manipulated in the same way that the transfer function descriptions (described earlier) can be manipulated. To convert a transfer function to a state-space representation, we can use the **tf2ss** function:

```
[A, B, C, D] = tf2ss(num, den);
```

And to perform the opposite operation, we can use the **ss2tf** function:

```
[num, den] = ss2tf(A, B, C, D);
```

he:משתנימצב/תורתהבקרה[3]

---

[3]   https://he.wikibooks.org/wiki/%D7%AA%D7%95%D7%A8%D7%AA%20%D7%94%D7%91%D7%A7%D7%A8%D7%
94%2F%D7%9E%D7%A9%D7%AA%D7%A0%D7%99%20%D7%9E%D7%A6%D7%91

13 Solutions for Linear Systems

13.1 State Equation Solutions

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

---

a   https://en.wikibooks.org/wiki/Ordinary%20Differential%20Equations

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 Ordinary Differential Equations[1] 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).

13.2 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:

---

1   https://en.wikibooks.org/wiki/Ordinary%20Differential%20Equations

$$\frac{dx(t)}{x(t)} = A dt$$

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:

$$x(t) = e^{A(t-t_0)} x(t_0)$$

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.

13.3 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 $Ax(t)$ 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:

13.3.1 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:

$f(t) = e^{-At} \qquad f'(t) = -Ae^{-At}$

$g(t) = x(t) \qquad g'(t) = x'(t)$

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:

[General State Equation Solution]

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

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

[General Output Equation Solution]

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

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.

13.4 State-Transition Matrix

More information about **matrix exponentials** can be found in:
Engineering Analysis[a]

---

[a]   https://en.wikibooks.org/wiki/Engineering%20Analysis

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.

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

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

More information about **diagonal matrices** and **Jordan-form matrices** can be found in:
Engineering Analysis[a]

---

[a]   https://en.wikibooks.org/wiki/Engineering%20Analysis

Also, we can attempt to diagonalize the matrix A into a **diagonal matrix** or a **Jordan Canonical matrix**. The exponential of a diagonal matrix is simply the diagonal elements individually raised to that exponential. The exponential of a Jordan canonical 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[2].

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

13.4.1 Diagonal Matrices

If a matrix is diagonal, the state transition matrix can be calculated by raising each diagonal entry of the matrix raised as a power of e.

13.4.2 Jordan Canonical Form

If the A matrix is in the Jordan Canonical form, then the matrix exponential can be generated quickly using the following formula:

$$e^{Jt} = e^{\lambda t} \begin{bmatrix} 1 & t & \frac{1}{2!}t^2 & \cdots & \frac{1}{n!}t^n \\ 0 & 1 & t & \cdots & \frac{1}{(n-1)!}t^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Where λ is the eigenvalue (the value on the diagonal) of the jordan-canonical matrix.

---

2   https://en.wikibooks.org/wiki/Engineering%20Analysis

13.4.3 Inverse Laplace Method

We can calculate the state-transition matrix (or any matrix exponential function) by taking the following inverse Laplace transform:

$$e^{At} = \mathcal{L}^{-1}[(sI - A)^{-1}]$$

If A is a high-order matrix, this inverse can be difficult to solve.

If the A matrix is in the Jordan Canonical form, then the matrix exponential can be generated quickly using the following formula:

$$e^{Jt} = e^{\lambda t} \begin{bmatrix} 1 & t & \frac{1}{2!}t^2 & \cdots & \frac{1}{n!}t^n \\ 0 & 1 & t & \cdots & \frac{1}{(n-1)!}t^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Where λ is the eigenvalue (the value on the diagonal) of the jordan-canonical matrix.

13.4.4 Spectral Decomposition

If we know all the eigenvalues of A, we can create our transition matrix T, and our inverse transition matrix T^{-1} These matrices will be the matrices of the right and left eigenvectors, respectively. If we have both the left and the right eigenvectors, we can calculate the state-transition matrix as:

[Spectral Decomposition]

$$e^{At} = \sum_{i=1}^{n} e^{\lambda_i t} v_i w_i'$$

Note that w_i' is the transpose of the ith left-eigenvector, not the derivative of it. We will discuss the concepts of "eigenvalues", "eigenvectors", and the technique of spectral decomposition in more detail in a later chapter.

13.4.5 Cayley-Hamilton Theorem

For more information on the **Cayley-Hamilton Theorem**, see:
Engineering Analysis[a]

---

[a]   https://en.wikibooks.org/wiki/Engineering%20Analysis

The **Cayley-Hamilton Theorem** can also be used to find a solution for a matrix exponential. For any eigenvalue of the system matrix A, λ, we can show that the two equations are equivalent:

$$e^{\lambda t} = a_0 + a_1 \lambda t + a_2 \lambda^2 t^2 + \cdots + a_{n-1} \lambda^{n-1} t^{n-1}$$

Once we solve for the coefficients of the equation, a, we can then plug those coefficients into the following equation:

$$e^{At} = a_0 I + a_1 A t + a_2 A^2 t^2 + \cdots + a_{n-1} A^{n-1} t^{n-1}$$

13.4.6 Example: Off-Diagonal Matrix

Given the following matrix A, find the state-transition matrix:

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

We can find the eigenvalues of this matrix as $\lambda = i$, $-i$. If we plug these values into our eigenvector equation, we get:

$$\begin{vmatrix} i & -1 \\ 1 & i \end{vmatrix} v_1 = 0$$

$$\begin{vmatrix} -i & -1 \\ 1 & -i \end{vmatrix} v_2 = 0$$

And we can solve for our eigenvectors:

$$v_1 = \begin{bmatrix} 1 \\ i \end{bmatrix}$$

$$v_2 = \begin{bmatrix} 1 \\ -i \end{bmatrix}$$

With our eigenvectors, we can solve for our left-eigenvectors:

$$w_1 = \begin{bmatrix} 1 \\ -i \end{bmatrix}$$

$$w_2 = \begin{bmatrix} 1 \\ i \end{bmatrix}$$

Now, using spectral decomposition, we can construct the state-transition matrix:

$$e^{At} = e^{it} \begin{bmatrix} 1 \\ i \end{bmatrix} \begin{bmatrix} 1 & -i \end{bmatrix} + e^{-it} \begin{bmatrix} 1 \\ -i \end{bmatrix} \begin{bmatrix} 1 & i \end{bmatrix}$$

If we remember Euler's Identity, we can decompose the complex exponentials into sinusoids. Performing the vector multiplications, all the imaginary terms cancel out, and we are left with our result:

$$e^{At} = \begin{bmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{bmatrix}$$

The reader is encouraged to perform the multiplications, and attempt to derive this result.

13.4.7 Example: Sympy Calculation

With the freely available python library 'sympy' we can very easily calculate the state-transition matrix automatically:

```
>>> from sympy import *
>>> t = symbols('t', positive = true)
>>> A = Matrix([[0,1],[-1,0]])
>>> exp(A*t).expand(complex=True)
```

$$\begin{bmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{bmatrix}$$

You can also try it out yourself on this website:

sympy live[3]

13.4.8 Example: MATLAB Calculation

Using the **symbolic toolbox** in MATLAB, we can write MATLAB code to automatically generate the state-transition matrix for a given input matrix A. Here is an example of MATLAB code that can perform this task:

```
function [phi] = statetrans(A)
   t = sym('t');
   phi = expm(A * t);
end
```

Use this MATLAB function to find the state-transition matrix for the following matrices (warning, calculation may take some time):

1. $A_1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

2. $A_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

3. $A_3 = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$

   Matrix 1 is a diagonal matrix, Matrix 2 has complex eigenvalues, and Matrix 3 is Jordan canonical form. These three matrices should be representative of some of the common forms of system matrices. The following code snippets are the input commands into MATLAB to produce these matrices, and the output results:

Matrix A1

```
>> A1 = [2 0 ; 0 2]; >> statetrans(A1)

ans =

[ exp(2*t), 0] [ 0, exp(2*t)]
```

Matrix A2

```
>> A2 = [0 1 ; -1 0]; >> statetrans(A1)

ans =

[ cos(t), sin(t)] [ -sin(t), cos(t)]
```

Matrix A3

```
>> A1 = [2 1 ; 0 2]; >> statetrans(A1)

ans =

[ exp(2*t), t*exp(2*t)] [ 0, exp(2*t)]
```

13.4.9 Example: Multiple Methods in MATLAB

There are multiple methods in MATLAB to compute the state transtion matrix, from a scalar (time-invariant) matrix A. The following methods are all going to rely on the *Symbolic Toolbox* to perform the equation manipulations. At the end of each code snippet, the variable **eAt** contains the state-transition matrix of matrix A.

Direct Method

```
t = sym('t'); eAt = expm(A * t);
```

Laplace Transform Method

```
s = sym('s'); [n,n] = size(A); in = inv(s*eye(n) - A); eAt = ilaplace(in);
```

Spectral Decomposition

```
t = sym('t'); [n,n] = size(A); [V, e] = eig(A); W = inv(V); sum = [0 0;0 0]; for I = 1:n
sum = sum + expm(e(I,I)*t)*V(:,I)*W(I,:); end; eAt = sum;
```

All three of these methods should produce the same answers. The student is encouraged to verify this.

he:פתרונןמשוואתהמצבעבורהמערכתקבועהבזמן/תורתהבקרה[4]

---

14 Time-Variant System Solutions

14.1 General Time Variant Solution

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

[Time-Variant General Solution]

$$x(t) = \phi(t, t_0)x(t_0) + \int_{t_0}^{t} \phi(t, \tau)B(\tau)u(\tau)d\tau$$

Matrix Dimensions:
A: $p \times p$
B: $p \times q$
C: $r \times p$
D: $r \times q$

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-variant 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.

14.1.1 State Transition Matrix

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

The state transition matrix ϕ is not completely unknown, it must always satisfy 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:

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.	$\frac{d\phi(t_0, t_0)}{dt} = A(t)$

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 dependent 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.

14.2 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

14.2.1 Fundamental Matrix

Here, x is an $n \times 1$ vector, and A is an $n \times n$ matrix.

Given the equation:

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

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.

Readers who have a background in Linear Algebra[a] 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.

---

[a]   https://en.wikibooks.org/wiki/Linear%20Algebra

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

$$\mathcal{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$$

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.

14.2.2 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 transition matrix, and we will discuss other methods below.

14.2.3 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 first task is to find the inverse of the fundamental matrix. Because the fundamental matrix is a 2×2 matrix, the inverse can be given easily through a common formula:

$$\mathcal{X}^{-1}(t) = \frac{\begin{bmatrix} e^{-t} & -\frac{1}{2}e^{t} \\ 0 & e^{-t} \end{bmatrix}}{e^{-2t}} = \begin{bmatrix} e^{t} & -\frac{1}{2}e^{3t} \\ 0 & e^{t} \end{bmatrix}$$

The state-transition matrix is given by:

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

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

14.2.4 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:

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

The state transition matrix will commute as described above if any of the following conditions are true:

1. A is a constant matrix (time-invariant)
2. A is a diagonal matrix
3. If $A = \bar{A}f(t)$, where $\bar{A}$ is a constant matrix, and f(t) is a scalar-valued function (not a matrix).

If none of the above conditions are true, then you must use **method 3**.

Method 3

If A(t) can be decomposed as the following sum:

$$A(t) = \sum_{i=1}^{n} M_i f_i(t)$$

Where M_i is a constant matrix such that $M_i M_j = M_j M_i$, and f_i is a scalar-valued function. If A(t) can be decomposed in this way, then the state-transition matrix can be given as:

$$\phi(t, \tau) = \prod_{i=1}^{n} e^{M_i \int_{\tau}^{t} f_i(\theta) d\theta}$$

It will be left as an exercise 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 $e^{A(t-\tau)}$.

14.2.5 Example: Using Method 3

Use method 3, above, to compute the state-transition matrix for the system if the system matrix A is given by:

$$A = \begin{bmatrix} t & 1 \\ -1 & t \end{bmatrix}$$

We can decompose this matrix as follows:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} t + \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Where $f_1(t) = t$, and $f_2(t) = 1$. Using the formula described above gives us:

$$\phi(t, \tau) = e^{M_1 \int_{\tau}^{t} \theta d\theta} e^{M_2 \int_{\tau}^{t} d\theta}$$

Solving the two integrations gives us:

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

The first term is a diagonal matrix, and the solution to that matrix function is all the individual elements of the matrix raised as an exponent of e. The second term can be decomposed as:

$$e^{\begin{bmatrix} 0 & t-\tau \\ -t+\tau & 0 \end{bmatrix}} = e^{\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}(t-\tau)} = \begin{bmatrix} \cos(t-\tau) & \sin(t-\tau) \\ -\sin(t-\tau) & \cos(t-\tau) \end{bmatrix}$$

The final solution is given as:

$$\phi(t,\tau) = \begin{bmatrix} e^{\frac{1}{2}(t^2-\tau^2)} & 0 \\ 0 & e^{\frac{1}{2}(t^2-\tau^2)} \end{bmatrix} \begin{bmatrix} \cos(t-\tau) & \sin(t-\tau) \\ -\sin(t-\tau) & \cos(t-\tau) \end{bmatrix} =$$

$$\begin{bmatrix} e^{\frac{1}{2}(t^2-\tau^2)}\cos(t-\tau) & e^{\frac{1}{2}(t^2-\tau^2)}\sin(t-\tau) \\ -e^{\frac{1}{2}(t^2-\tau^2)}\sin(t-\tau) & e^{\frac{1}{2}(t^2-\tau^2)}\cos(t-\tau) \end{bmatrix}$$

14.3 Time-Variant, Non-zero Input

If the input to the system is not zero, it turns out that all the analysis that we performed above still holds. We can still construct the fundamental matrix, and we can still represent the system solution in terms of the state transition matrix ϕ.

We can show that the general solution to the state-space equations is actually the solution:

$$x(t) = \phi(t,t_0)x(t_0) + \int_{t_0}^{t} \phi(t,\tau)B(\tau)u(\tau)d\tau$$

15 Digital State-Space

15.1 Digital Systems

Digital systems, expressed previously as difference equations or Z-Transform transfer functions can also be used with the state-space representation. Also, all the same techniques for dealing with analog systems can be applied to digital systems, with only minor changes.

15.2 Digital Systems

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

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

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

15.2.1 Zero-Order Hold Derivation

If we have a continuous-time state equation:

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

We can derive the digital version of this equation that we discussed above. We take the Laplace transform of our equation:

$$X(s) = (sI - A)^{-1}Bu(s) + (sI - A)^{-1}x(0)$$

Now, taking the inverse Laplace transform gives us our time-domain system, keeping in mind that the inverse Laplace transform of the *(sI - A)* term is our state-transition matrix, Φ:

$$x(t) = \mathcal{L}^{-1}(X(s)) = \Phi(t - t_0)x(0) + \int_{t_0}^{t} \Phi(t - \tau)Bu(\tau)d\tau$$

Now, we apply a zero-order hold on our input, to make the system digital. Notice that we set our start time $t_0 = \text{kT}$, because we are only interested in the behavior of our system during a single sample period:

$$u(t) = u(kT), kT \leq t \leq (k+1)T$$

$$x(t) = \Phi(t,kT)x(kT) + \int_{kT}^{t} \Phi(t,\tau)Bd\tau u(kT)$$

We were able to remove $u(kT)$ from the integral because it did not rely on τ. We now define a new function, Γ, as follows:

$$\Gamma(t,t_0) = \int_{t_0}^{t} \Phi(t,\tau)Bd\tau$$

Inserting this new expression into our equation, and setting $t = (k + 1)T$ gives us:

$$x((k+1)T) = \Phi((k+1)T,kT)x(kT) + \Gamma((k+1)T,kT)u(kT)$$

Now $\Phi(T)$ and $\Gamma(T)$ are constant matrices, and we can give them new names. The d subscript denotes that they are digital versions of the coefficient matrices:

$$A_d = \Phi((k+1)T,kT)$$

$$B_d = \Gamma((k+1)T,kT)$$

We can use these values in our state equation, converting to our bracket notation instead:

$$x[k+1] = A_d x[k] + B_d u[k]$$

15.3 Relating Continuous and Discrete Systems

w:Discretization[1] 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. The process of converting an analog system for use with digital hardware is called **discretization**. We've given a basic introduction to discretization already, but we will discuss it in more detail here.

---

1   https://en.wikipedia.org/wiki/Discretization

15.3.1 Discrete Coefficient Matrices

Of primary importance in discretization is the computation of the associated coefficient matrices from the continuous-time counterparts. If we have the continuous system *(A, B, C, D)*, we can use the relationship $t = kT$ to transform the state-space solution into a sampled system:

$$x(kT) = e^{AkT}x(0) + \int_0^{kT} e^{A(kT-\tau)}Bu(\tau)d\tau$$

$$x[k] = e^{AkT}x[0] + \int_0^{kT} e^{A(kT-\tau)}Bu(\tau)d\tau$$

Now, if we want to analyze the *k+1* term, we can solve the equation again:

$$x[k+1] = e^{A(k+1)T}x[0] + \int_0^{(k+1)T} e^{A((k+1)T-\tau)}Bu(\tau)d\tau$$

Separating out the variables, and breaking the integral into two parts gives us:

$$x[k+1] = e^{AT}e^{AkT}x[0] + \int_0^{kT} e^{AT}e^{A(kT-\tau)}Bu(\tau)d\tau + \int_{kT}^{(k+1)T} e^{A(kT+T-\tau)}Bu(\tau)d\tau$$

If we substitute in a new variable $\beta = (k + 1)T + \tau$, and if we see the following relationship:

$$e^{AkT}x[0] = x[k]$$

We get our final result:

$$x[k+1] = e^{AT}x[k] + \left(\int_0^T e^{A\alpha}d\alpha\right)Bu[k]$$

Comparing this equation to our regular solution gives us a set of relationships for converting the continuous-time system into a discrete-time system. 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.

Matrix Dimensions:
A: $p \times p$
B: $p \times q$
C: $r \times p$
D: $r \times q$

$A_d = e^{A_c T}$

$B_d = \int_0^T e^{A_c \tau} d\tau B_c$
$C_d = C_c$
$D_d = D_c$

c2d

If the A_c matrix is nonsingular, then we can find its inverse and instead define B_d as:

$$B_d = A_c^{-1}(A_d - I)B_c$$

The differences in the discrete and continuous matrices are due to the fact that the underlying equations that describe our systems are different. Continuous-time systems are represented by linear differential equations, while the digital systems are described by difference equations. High order terms in a difference equation are delayed copies of the signals, while high order terms in the differential equations are derivatives of the analog signal.

If we have a complicated analog system, and we would like to implement that system in a digital computer, we can use the above transformations to make our matrices conform to the new paradigm.

15.3.2 Notation

Because the coefficient matrices for the discrete systems are computed differently from the continuous-time coefficient matrices, and because the matrices technically represent different things, it is not uncommon in the literature to denote these matrices with different variables. For instance, the following variables are used in place of A and B frequently:

$$\Omega = A_d$$

$$R = B_d$$

These substitutions would give us a system defined by the ordered quadruple *(Ω, R, C, D)* for representing our equations.

As a matter of notational convenience, we will use the letters A and B to represent these matrices throughout the rest of this book.

15.4 Converting Difference Equations

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

$$y[n+3]+a_2 y[n+2]+a_1 y[n+1]+a_0 y[n]=u[n]$$

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:

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

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

15.5 Solving for x[n]

We can find a general time-invariant solution for the discrete time difference equations. Let us start working up a pattern. We know the discrete state equation:

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

Starting from time $n = 0$, we can start to create a pattern:

$$x[1]=Ax[0]+Bu[0]$$

$$x[2]=Ax[1]+Bu[1]=A^2 x[0]+ABu[0]+Bu[1]$$

$$x[3]=Ax[2]+Bu[2]=A^3 x[0]+A^2 Bu[0]+ABu[1]+Bu[2]$$

With a little algebraic trickery, we can reduce this pattern to a single equation:

[General State Equation Solution]

$$x[n] = A^n x[n_0] + \sum_{m=0}^{n-1} A^{n-1-m} Bu[m]$$

Substituting this result into the output equation gives us:

[General Output Equation Solution]

$$y[n] = CA^n x[n_0] + \sum_{m=0}^{n-1} CA^{n-1-m} Bu[m] + Du[n]$$

15.6 Time Variant Solutions

If the system is time-variant, we have a general solution that is similar to the continuous-time case:

$$x[n] = \phi[n, n_0] x[n_0] + \sum_{m=n_0}^{n-1} \phi[n, m+1] B[m] u[m]$$

$$y[n] = C[n] \phi[n, n_0] x[n_0] + C[n] \sum_{m=n_0}^{n-1} \phi[n, m+1] B[m] u[m] + D[n] u[n]$$

Where φ, the **state transition matrix**, is defined in a similar manner to the state-transition matrix in the continuous case. However, some of the properties in the discrete time are different. For instance, the inverse of the state-transition matrix does not need to exist, and in many systems it does not exist.

15.6.1 State Transition Matrix

The discrete time state transition matrix is the unique solution of the equation:

$$\phi[k+1, k_0] = A[k] \phi[k, k_0]$$

Where the following restriction must hold:

$$\phi[k_0, k_0] = I$$

From this definition, an obvious way to calculate this state transition matrix presents itself:

$$\phi[k, k_0] = A[k-1] A[k-2] A[k-3] \cdots A[k_0]$$

Or,

$$\phi[k, k_0] = \prod_{m=1}^{k-k_0} A[k-m]$$

15.7 MATLAB Calculations

MATLAB is a computer program, and therefore calculates all systems using digital methods. The MATLAB function **lsim** is used to simulate a continuous system with a specified input. This function works by calling the **c2d**, which converts a system *(A, B, C, D)* into the equivalent discrete system. Once the system model is discretized, the function passes control to the **dlsim** function, which is used to simulate discrete-time systems with the specified input.

Because of this, simulation programs like MATLAB are subjected to round-off errors associated with the discretization process.

16 Eigenvalues and Eigenvectors

16.1 Eigenvalues and Eigenvectors

Eigenvalues and Eigenvectors cannot be calculated from time-variant matrices. If the system is time-variant, the methods described in this chapter will not produce valid results.

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" comes from German and means "own", while it is the Dutch word for "characteristic", and so this chapter could also be called "Characteristic values and characteristic vectors". The terms "Eigenvalues" and "Eigenvectors" are most commonly used. 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 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.

16.2 Characteristic Equation

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

[Matrix Characteristic Equation]

$$Av = \lambda v$$

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:

[Left-Eigenvector Equation]

$$wA = \lambda w$$

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

- Linear Algebra[1]
- Engineering Analysis[2]

16.2.1 Diagonalization

> **Note:**
> The transition matrix T should not be confused with the sampling time of a discrete system. If needed, we will use subscripts to differentiate between the two.

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 right-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:

---

1   https://en.wikibooks.org/wiki/Linear%20Algebra
2   https://en.wikibooks.org/wiki/Engineering%20Analysis

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

$$T^{-1} = \begin{bmatrix} w'_1 \\ w'_2 \\ w'_3 \\ \vdots \\ w'_n \end{bmatrix}$$

We will further discuss the concept of diagonalization later in this chapter.

16.3 Exponential Matrix Decomposition

For more information about spectral decomposition, see:
Spectral Decomposition[a]

---

[a]   https://en.wikibooks.org/wiki/Engineering%20Analysis%2FSpectral%20Decomposition

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

$$e^{At} = \sum_{i=1}^{n} e^{\lambda_i t} v_i w'_i$$

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:

$$e^{At} x(t_0) = \sum_{i=1}^{n} \alpha_i e^{\lambda_i t} v_i$$

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]

$$x(t) = \sum_{i=1}^{n} \alpha_i e^{\lambda_i t} v_i + \sum_{i=1}^{n} \int_0^t e^{\lambda_i(t-\tau)} v_i w'_i B u(\tau) d\tau$$

We will talk about this equation in the following sections.

16.3.1 State Relationship

As we can see from the above equation, the individual elements of the state vector $x(t)$ cannot take arbitrary values, but they are instead related by weighted sums of multiples of the systems right-eigenvectors.

16.3.2 Decoupling

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 to decouple the system.

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.

16.3.3 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:

[Condition Number]

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

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.

16.3.4 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 *positive*, 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 severely 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.

16.4 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 canonical form**, not the diagonal form we were using earlier.

16.4.1 Generalized Eigenvectors

Generalized eigenvectors can be generated using the following equation:

[Generalized Eigenvector Generating Equation]

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

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.

16.4.2 Example: One Repeated Set

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.

16.4.3 Example: Two Repeated Sets

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:

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

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

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:

$[a \; b \; d \; c]$

16.4.4 Jordan Canonical Form

For more information about **Jordan Canonical Form**, see:
Matrix Forms[a]

---

[a]   https://en.wikibooks.org/wiki/Engineering%20Analysis%2FMatrix%20Forms

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 sub-diagonal.

16.4.5 System Response

16.5 Equivalence Transformations

If we have a non-singular $n \times n$ 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:

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

$$\bar{y}(t) = \bar{C}\bar{x}(t) + \bar{D}u(t)$$

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.

16.5.1 Lyapunov Transformations

The transformation matrix P is called a **Lyapunov Transformation** if the following conditions hold:

- $P(t)$ is nonsingular.
- $P(t)$ and $P'(t)$ are continuous
- $P(t)$ and the inverse transformation matrix P^{-1}(t) are finite for all t.

If a system is time-variant, it can frequently be useful to use a Lyapunov transformation to convert the system to an equivalent system with a constant A matrix. This is not always possible in general, however it is possible if the $A(t)$ matrix is periodic.

16.5.2 System Diagonalization

If the A matrix is time-invariant, we can construct the matrix V from the eigenvectors of A. The V matrix can be used to transform the A matrix to a diagonal matrix. Our new system becomes:

$$Vx'(t) = VAV^{-1}Vx(t) + VBu(t)$$

$$y(t) = CV^{-1}Vx(t) + Du(t)$$

Since our system matrix is now diagonal (or Jordan canonical), the calculation of the state-transition matrix is simplified:

$$e^{VAV^{-1}} = \Lambda$$

Where Λ is a diagonal matrix.

16.5.3 MATLAB Transformations

The MATLAB function **ss2ss** can be used to apply an equivalence transformation to a system. If we have a set of matrices A, B, C and D, we can create equivalent matrices as such:

```
[Ap, Bp, Cp, Dp] = ss2ss(A, B, C, D, p);
```

Where p is the equivalence transformation matrix.

17 MIMO Systems

17.1 Multi-Input, Multi-Output

Systems with more than one input and/or more than 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.

17.2 State-Space Representation

See the Formatting Section[a] in the introduction if the notation in this page is confusing.

---
[a]   Chapter 2.11 on page 14

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 dependent on the input vector and the state vector.

$$X'(t) = AX(t) + BU(t)$$

$$Y(t) = CX(t) + DU(t)$$

17.2.1 Example: Two Inputs and Two Outputs

Let's say that we have two outputs, y_1 and y_2, and two inputs, u_1 and u_2. These are related in our system through the following system of differential equations:

$$y_1'' + a_1 y_1' + a_0(y_1 + y_2) = u_1(t)$$

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

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

$$x_1 = y_1$$

$$x_4 = y_2$$

$$x_1' = y_1' = x_2$$

$$x_2' = -a_1 x_2 - a_0(x_1 + x_4) + u_1(t)$$

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

And finally we can assemble our state space equations:

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

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

17.3 Transfer Function Matrix

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

$$\mathcal{L}[X'(t)] = \mathcal{L}[AX(t)] + \mathcal{L}[BU(t)]$$

$$\mathcal{L}[Y(t)] = \mathcal{L}[CX(t)] + \mathcal{L}[DU(t)]$$

Which gives us the result:

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

$$\mathbf{Y}(s) = C\mathbf{X}(s) + D\mathbf{U}(s)$$

Where X(0) is the initial conditions of the system state vector in the time domain. 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:

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

Then factor out an $\mathbf{X}(s)$:

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

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

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

Now, if we plug in this value for $\mathbf{X}$(s) into our output equation, above, we get a more complicated equation:

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

And we can distribute the matrix $\mathbf{C}$ to give us our answer:

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

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 $\mathbf{U}$(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**, $\mathbf{H}$(s):

[**Transfer Matrix**]

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

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

[**Transfer Matrix Description**]

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

If $\mathbf{Y}$*(s)* and $\mathbf{X}$*(s)* are *1 × 1* vectors (a SISO system), then we have our external description:

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

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

17.3.1 Dimensions

If our system has q inputs, and r outputs, our transfer function matrix will be an $r \times q$ matrix.

17.3.2 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 $n \times m$ transfer functions, where each entry is the transfer function relationship between each individual input, and each individual output.

Through this derivation of the transfer function matrix, we have shown the equivalency between 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 explanation, we will use the Laplace and State Space methods interchangeably, opting to use one or the other where appropriate.

17.3.3 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:

- $C[sI - A]^{-1}X(0)$ 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.

17.4 Discrete MIMO Systems

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

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

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

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

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

[Transfer Matrix]

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

[Transfer Matrix Description]

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

17.4.1 Example: 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, $Y[n]$, to the system input vector, $U[n]$:

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

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:

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

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

18 System Realization

18.1 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".

18.2 Realization Conditions

> **Note:**
> Discrete systems $G(z)$ are also realizable if these conditions are satisfied.

- 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 four system matrices, is called a **realization** of the system $G(s)$. If the system can be expressed as such an ordered quadruple, the system is realizable.
- A system G is realizable if and only if the transfer matrix $\mathbf{G}(s)$ is a proper rational matrix. In other words, every entry in the matrix $\mathbf{G}(s)$ (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.

18.3 Realizing the Transfer Matrix

We can decompose a transfer matrix $\mathbf{G}(s)$ 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 $\mathbf{G}(s)$:

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

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

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 $p \times q$ 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:

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

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

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

19 Gain

19.1 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:

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

In a state-space system, the gain term k can be inserted as follows:

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

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

The gain term can also be inserted into other places in the system, and in those cases the equations will be slightly different.
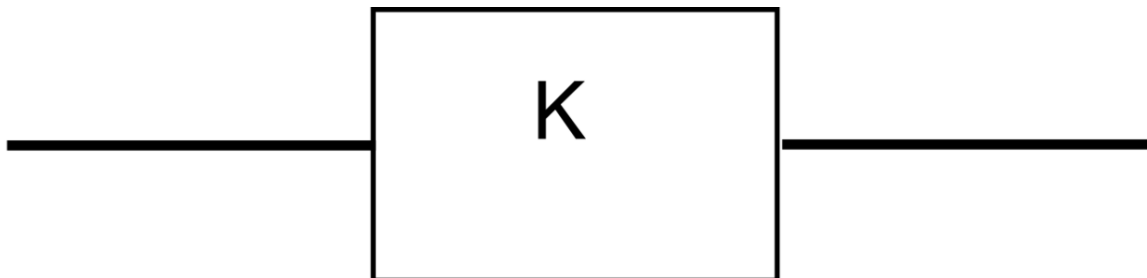
Figure 31

19.1.1 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.

19.2 Responses to Gain

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

19.3 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**. But it will decrease the steady state error.

19.3.1 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 value of the gain, and often the threshold where the system becomes unstable is important to find.

20 Block Diagrams

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 what it means:

20.1 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 product of the individual systems.
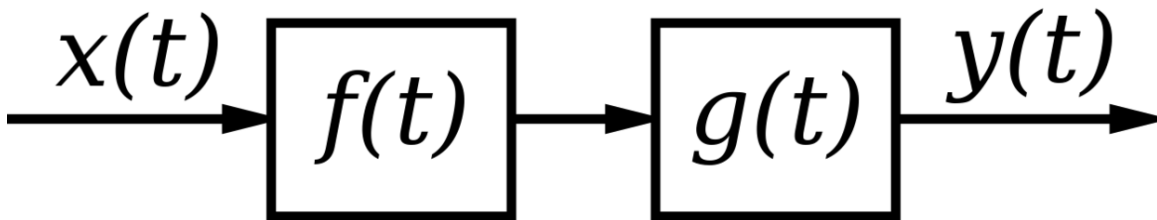
Figure 32

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

If we define the output of the first system as *h(t)*, we can define *h(t)* as:

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

Now, we can define the system output *y(t)* in terms of *h(t)* as:

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

We can expand *h(t)*:

$$y(t) = [x(t) * f(t)] * g(t)$$

But, since convolution is associative, we can re-write this as:

$$y(t) = x(t) * [f(t) * g(t)]$$

Our system can be simplified therefore as such:
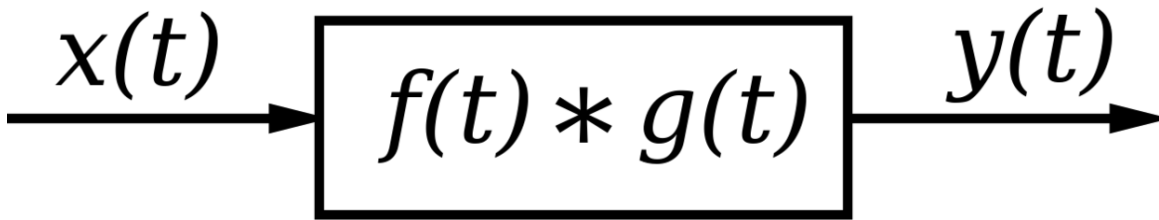
Figure 33

20.1.1 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.
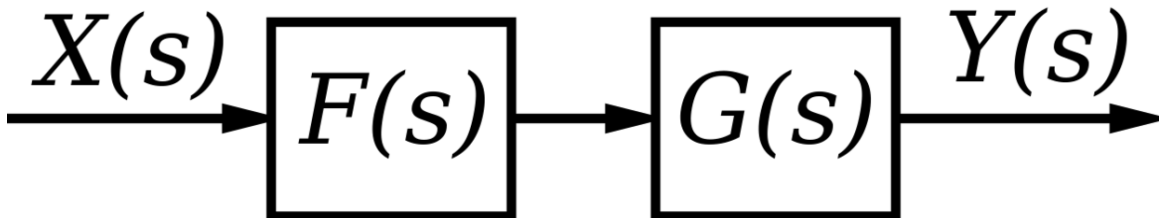
Figure 34

In the time-domain we know that:

$$y(t) = x(t) * [f(t) * g(t)]$$

But, in the frequency domain we know that convolution becomes multiplication, so we can re-write this as:

$$Y(s) = X(s)[F(s)G(s)]$$

We can represent our system in the frequency domain as:
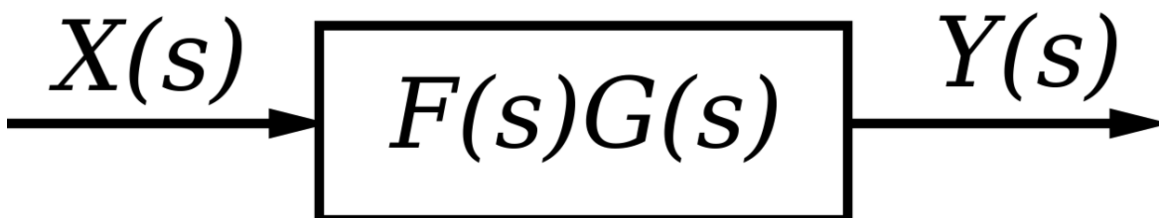
Figure 35

20.1.2 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:

$$x'_F = A_F x_F + B_F u$$

$$y_F = C_F x_F + D_F u$$

System 2:

$$x'_G = A_G x_G + B_G y_F$$

$$y_G = C_G x_G + D_G y_F$$

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

[Series state equation]

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

[Series output equation]

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

20.2 Systems in Parallel
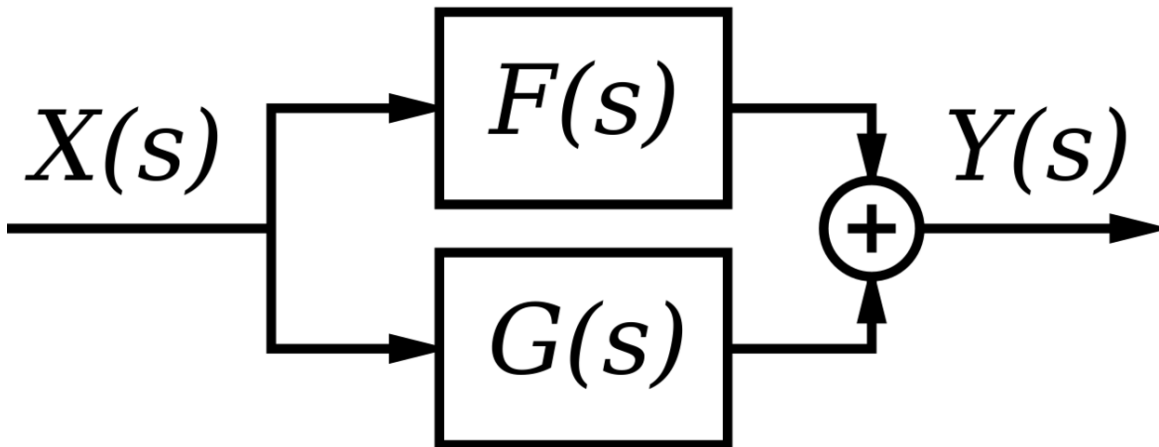
Figure 36

Blocks may not be placed in parallel without the use of an adder. Blocks connected by an adder as shown above have a total transfer function of:

$$Y(s) = X(s)[F(s) + G(s)]$$

Since the Laplace transform is linear, we can easily transfer this to the time domain by converting the multiplication to convolution:

$$y(t) = x(t) * [f(t) + g(t)]$$
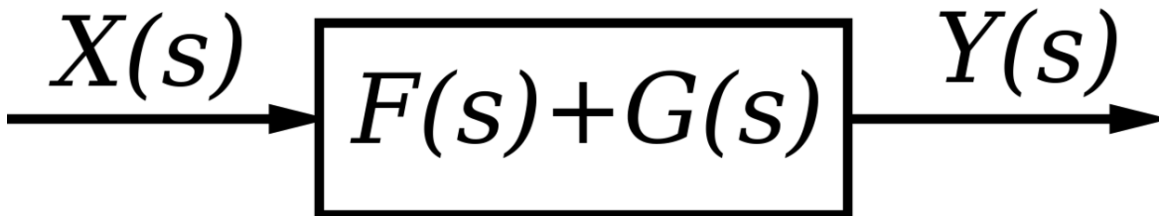
Figure 37

20.3 State Space Model

The state-space equations, with non-zero A, B, C, and D matrices conceptually model the following system:
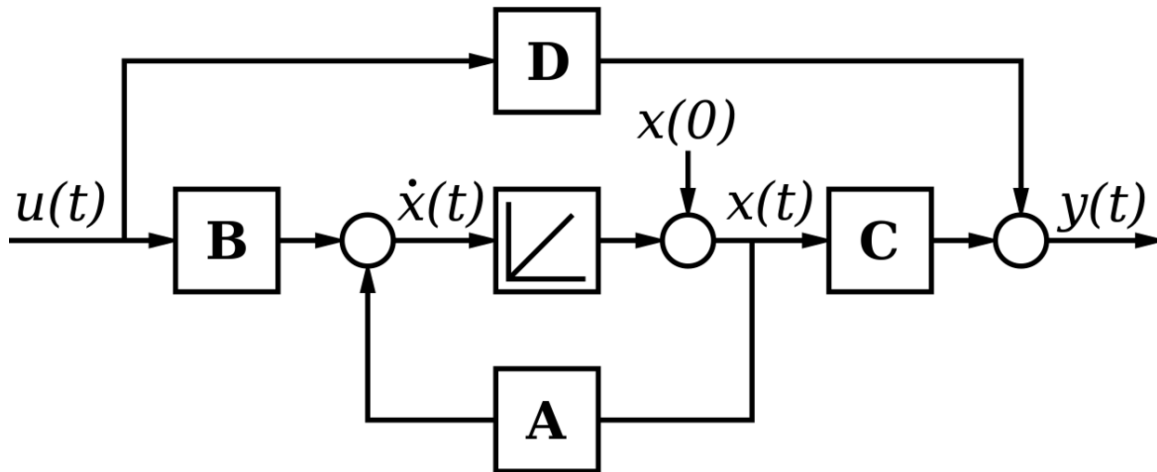
Figure 38

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

$\frac{1}{s}$ 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:
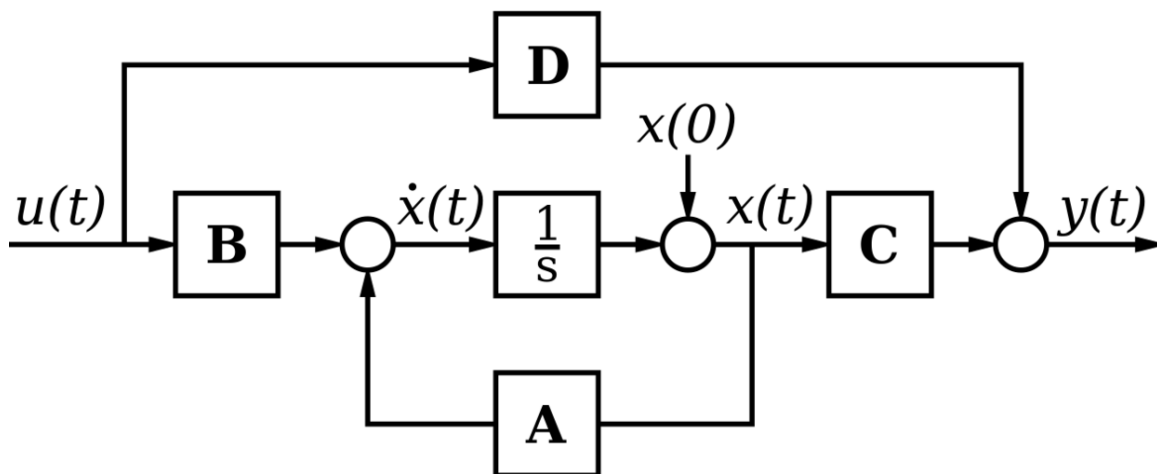
Figure 39

20.3.1 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:

$$\frac{Y(s)}{U(s)} = B(s)\left(\frac{1}{s - A(s)}\right)C(s) + D(s)$$
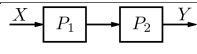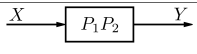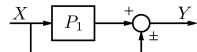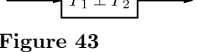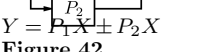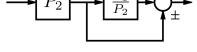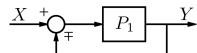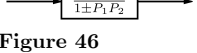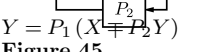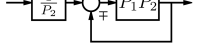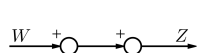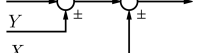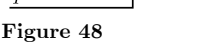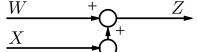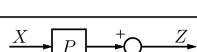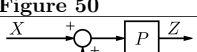
We will explain how we got this result, and how we deal with feedforward and feedback loop structures in the next chapter.
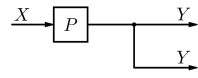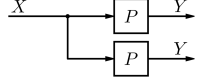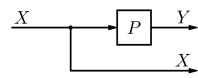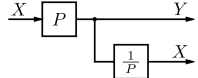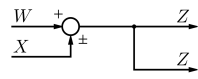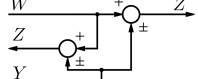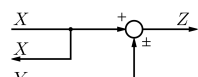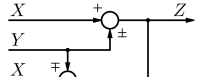
20.4 Adders and Multipliers

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

20.5 Simplifying Block Diagrams

Block diagrams can be systematically simplified. Note that this table is from Schaum's Outline: Feedback and Controls Systems by DiStefano et al

	Transformation	Equation	Block Diagram	Equivalent Block Diagram
1	Cascaded Blocks	$Y = (P_1 P_2) X$	Figure 40	Figure 41
2	Combining Blocks in Parallel	$Y = P_1 X \pm P_2 X$		Figure 43
3		Removing a Block from a Forward Loop	$Y = P_1 X \pm P_2 X$ Figure 42	Figure 44
4	Eliminating a Feedback Loop	$Y = P_1 (X \mp P_2 Y)$		Figure 46
5		Removing a Block from a Feedback Loop	$Y = P_1 (X \mp P_2 Y)$ Figure 45	Figure 47
6	Rearranging Summing Junctions	$Z = W \pm X \pm Y$	Figure 48	Figure 49 / Figure 50
7	Moving a Summing Junction in front of a Block	$Z = PX \pm Y$	Figure 51	Figure 52
8	Moving a Summing Junction beyond a Block	$Z = P(X \pm Y)$	Figure 53	Figure 54

	Transformation	Equation	Block Diagram	Equivalent Block Diagram
9	Moving a Takeoff Point in front of a Block	$Y = PX$	**Figure 55**	**Figure 56**
10	Moving a Takeoff Point beyond a Block	$Y = PX$	**Figure 57**	**Figure 58**
11	Moving a Takeoff Point in front of a Summing Junction	$Z = W \pm X$	**Figure 59**	**Figure 60**
12	Moving a Takeoff Point beyond a Summing Junction	$Z = X \pm Y$	**Figure 61**	**Figure 62**

20.6 External links

- SISO Block Diagram with transfer functions on ControlTheoryPro.com[1]

---

1   http://wikis.controltheorypro.com/index.php?title=Block_Diagram_Quick_Reference

21 Feedback Loops

21.1 Feedback

A **feedback loop** is a common and powerful tool when designing a control system. Feedback loops take the system output into consideration, which enables the system to adjust its performance to meet a desired output response.

When talking about control systems it is important to keep in mind that engineers typically are given existing systems such as actuators, sensors[1], motors, and other devices with set parameters, and are asked to adjust the performance of those systems. In many cases, it may not be possible to open the system (the "plant") and adjust it from the inside: modifications need to be made external to the system to force the system response to act as desired. This is performed by adding controllers, compensators, and feedback structures to the system.

21.2 Basic Feedback Structure
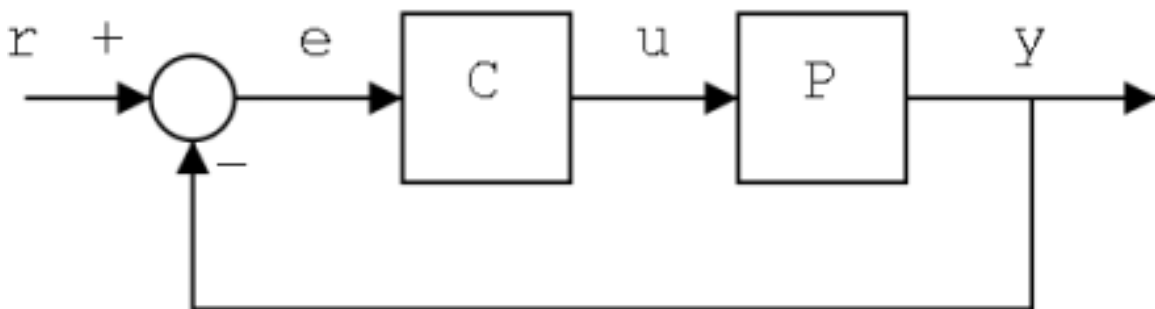
Figure 63 framed

w:Feedback[2] 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

---

1   https://en.wikibooks.org/wiki/Sensory%20Systems%2FVisual%20Signal%20Processing
2   https://en.wikipedia.org/wiki/Feedback

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 summer, before or after the plant, and in the feedback loop. In the picture above, our compensator is denoted with a C.

Some texts, or texts in other disciplines may refer to a "summer" as an **adder**.

Summer

A summer 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 summer, 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.

21.3 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.

Positive feedback has the property that signals tend to reinforce themselves, and grow larger. In a positive feedback system, noise from the system is added back to the input, and that in turn produces more noise. As an example of a positive feedback system, consider an audio amplification system with a speaker and a microphone. Placing the microphone near the speaker creates a positive feedback loop, and the result is a sound that grows louder and louder. Because the majority of noise in an electrical system is high-frequency, the sound output of the system becomes high-pitched.

21.3.1 Example: State-Space Equation

In the previous chapter, we showed you this picture:
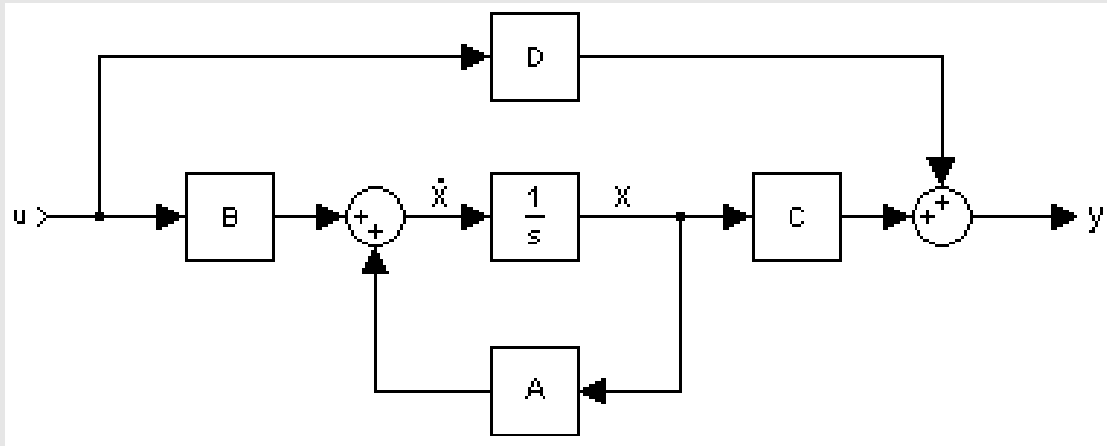
Figure 64

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 sumer 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.

21.4 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:

[Feedback Transfer Function]

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

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.

21.4.1 Example: Elevator

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:

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

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.

21.4.2 State-Space Feedback Loops

In the state-space representation, the plant is typically defined by the state-space equations:

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

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

The plant is considered to be pre-existing, and the matrices A, B, C, and D are considered to be internal to the plant (and therefore unchangeable). Also, in a typical system, the state variables are either fictional (in the sense of dummy-variables), or are not measurable. For these reasons, we need to add external components, such as a gain element, or a feedback element to the plant to enhance performance.

Consider the addition of a gain matrix K installed at the input of the plant, and a negative feedback element F that is multiplied by the system output y, and is added to the input signal of the plant. There are two cases:

1. The feedback element F is subtracted from the input before multiplication of the K gain matrix.
2. The feedback element F is subtracted from the input after multiplication of the K gain matrix.

In case 1, the feedback element F is added to the input before the multiplicative gain is applied to the input. If v is the input to the entire system, then we can define u as:

$$u(t) = Fv(t) - FKy(t)$$

In case 2, the feeback element F is subtracted from the input after the multiplicative gain is applied to the input. If v is the input to the entire system, then we can define u as:

$$u(t) = Kv(t) - Fy(t)$$

21.5 Open Loop vs Closed Loop
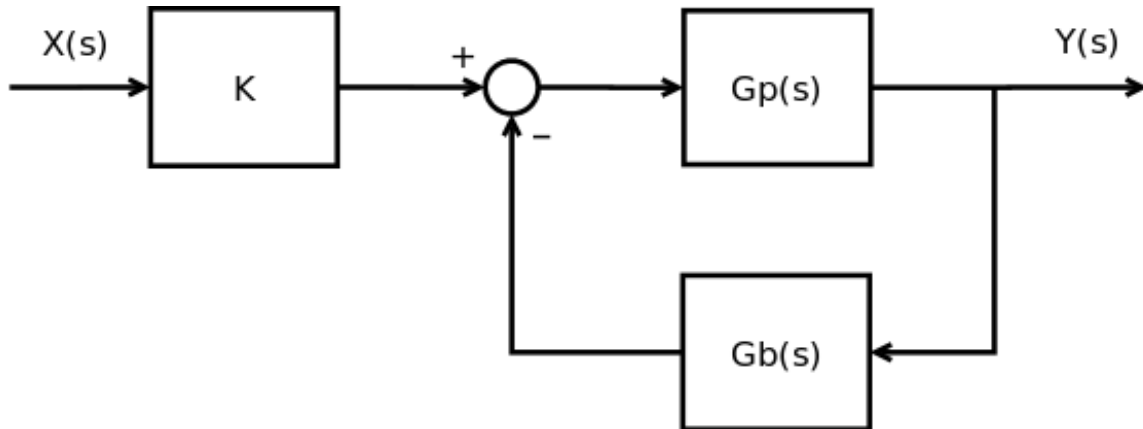
Figure 65

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:

[Closed-Loop Transfer Function]

$$H_{cl}(s) = \frac{KGp(s)}{1 + Gp(s)Gb(s)}$$

If we "open" the loop, and break the feedback node, we can define the **Open-Loop Transfer Function**, as:

[Open-Loop Transfer Function]

$$H_{ol}(s) = KGp(s)$$

We can redefine the closed-loop transfer function in terms of this open-loop transfer function:

$$H_{cl}(s) = \frac{H_{ol}(s)}{1 + Gp(s)Gb(s)}$$

These results are important, and they will be used without further explanation or derivation throughout the rest of the book.

21.6 Placement of a Controller

There are a number of different places where we could place an additional controller.
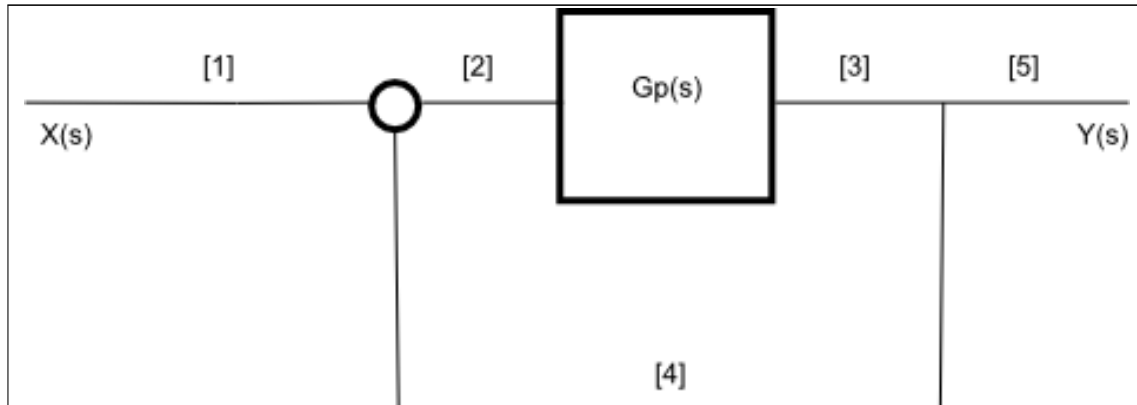
Figure 66

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.

21.7 Second-Order Systems

The general expression of the transfer function of a second order system is given as:

$$\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

where ζ and ω_n are damping ratio and natural frequency of the system respectively.

21.7.1 Damping Ratio

The damping ratio is defined by way of the sign ζ. The damping ratio gives us an idea about the nature of the transient response detailing the amount of overshoot & oscillation that the system will undergo. This is completely regardless of time scaling.

If :

- ζ = zero, the system is undamped;
- $\zeta < 1$, the system is underdamped;
- $\zeta = 1$, the system is critically damped;
- $\zeta > 1$, the system is overdamped.

ζ is used in conjunction with the natural frequency to determine system properties. To find the zeta value you must first find the natural response!

21.7.2 Natural Frequency

Natural Frequency, denoted by ω_n is defined as the frequency with which the system would oscillate if it were not damped and we define the damping ratio as $\zeta = \frac{\sigma}{\omega_n}$.

21.8 System Sensitivity

22 Signal Flow Diagrams

22.1 Signal-flow graphs

w:Signal-flow graph[1] **Signal-flow graphs** 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.

---

1   https://en.wikipedia.org/wiki/Signal-flow%20graph

$$\frac{\tilde{\theta}_1(s)}{\theta c_1(s)} = \frac{\omega_n^2}{s^2 + 2\varsigma\omega ns + \omega_n^2}$$
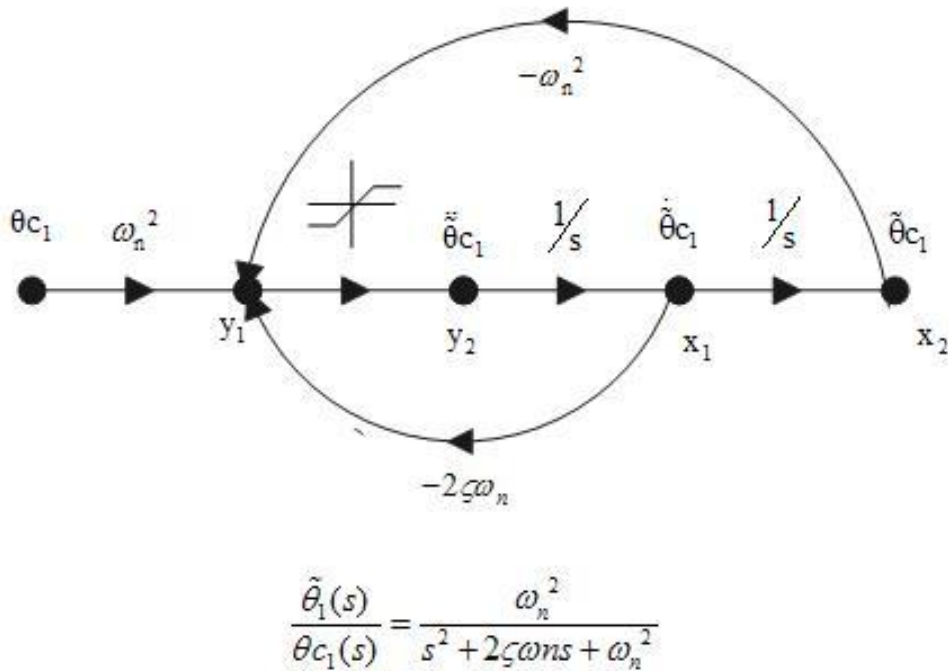
Figure 67   An example of a signal flow diagram.

22.1.1 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 than once. A single system can have multiple forward paths.

22.1.2 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.

22.1.3 Delta Values

The Delta value of a system, denoted with a Greek Δ is computed as follows:

$$\Delta = 1 - A + B - C + D - E + F \ldots\ldots + \infty$$

Where:

- A is the sum of all individual loop gains
- B is the sum of the products of all the pairs of non-touching loops
- C is the sum of the products of all the sets of 3 non-touching loops
- D is the sum of the products of all the sets of 4 non-touching loops
- et cetera.

If the given system has no pairs of loops that do not touch, for instance, B and all additional letters after B will be zero.

22.1.4 Mason's Rule

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. w:Mason's rule[2]

If we have computed our delta values (above), we can then use **Mason's Gain Rule** to find the complete gain of the system:

[Mason's Rule]

$$M = \frac{y_{out}}{y_{in}} = \sum_{k=1}^{N} \frac{M_k \Delta_k}{\Delta}$$

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.

22.2 Examples

22.2.1 Solving a signal-flow graph by systematic reduction : Two interlocking loops

This example shows how a system of five equations in five unknowns is solved using systematic reduction rules. The independent variable is x_{in}. The dependent variables are x_1, x_2, x_3, x_4, x_{out}. The coefficients are labeled a, b, c, d, e. Here is the starting flowgraph:

---

2   https://en.wikipedia.org/wiki/Mason%27s%20rule
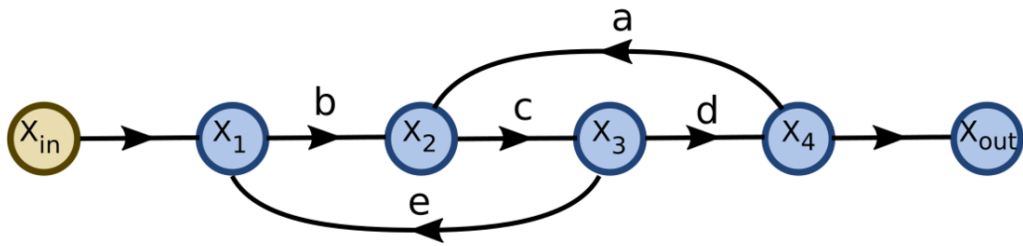
Figure 68   upright=2

$$x_1 = x_{\text{in}} + ex_3$$
$$x_2 = bx_1 + ax_4$$
$$x_3 = cx_2$$
$$x_4 = dx_3$$
$$x_{\text{out}} = x_4$$

The steps for solving x_{out} follow.

Removing edge c from x2 to x3
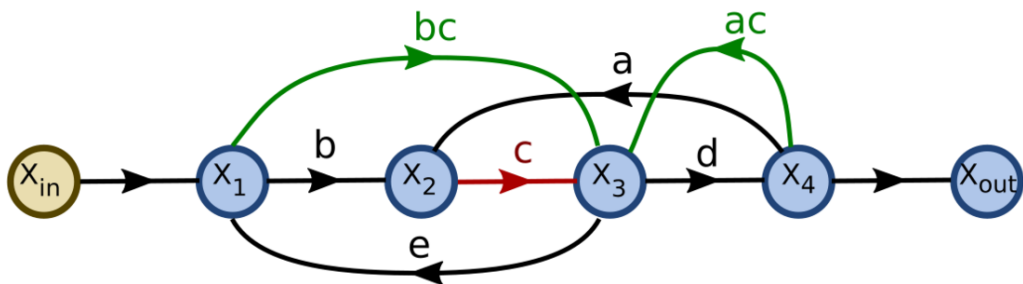
Figure 69   upright=2

$$x_1 = x_{\text{in}} + ex_3$$
$$x_2 = bx_1 + ax_4$$
$$x_3 = cx_2$$
$$x_3 = c(bx_1 + ax_4)$$
$$x_3 = bcx_1 + cax_4$$
$$x_4 = dx_3$$
$$x_{\text{out}} = x_4$$
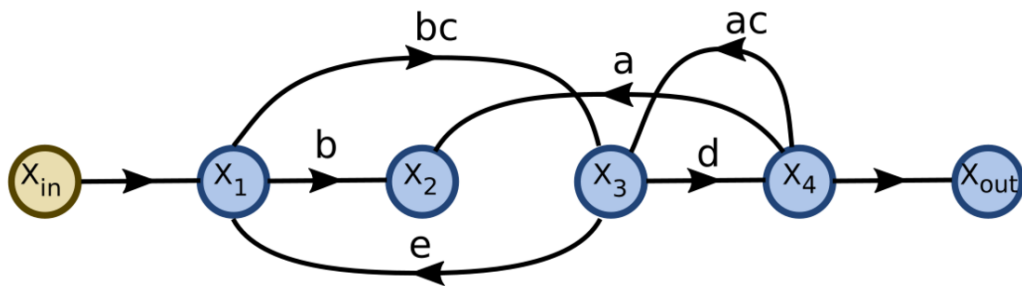
Figure 70 upright=2

Removing node x2 and its inflows

x_2 has no outflows, and is not a node of interest.
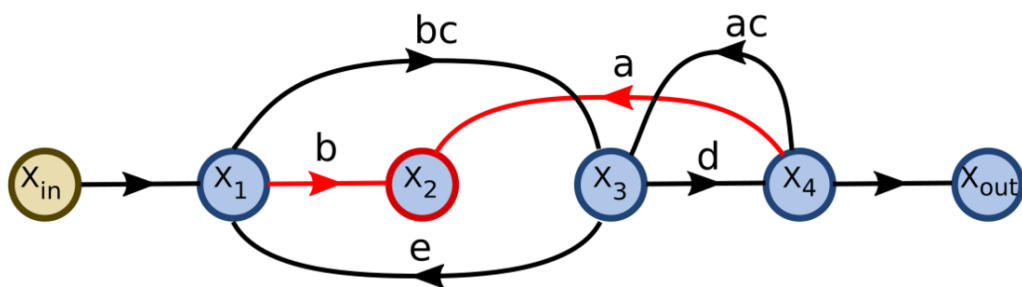
Figure 71 upright=2
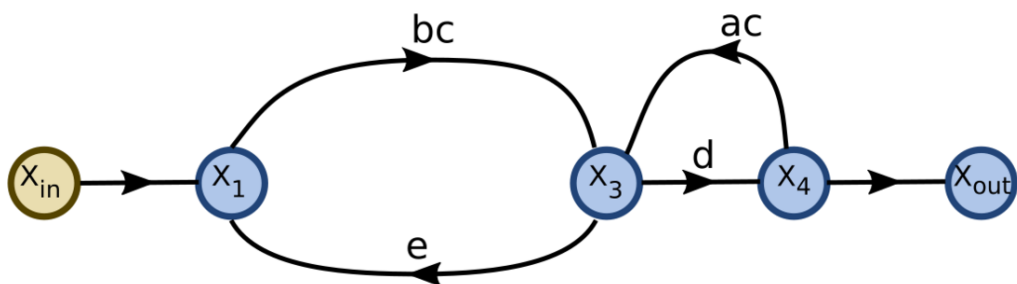
Figure 72 upright=2

Removing edge e from x3 to x1
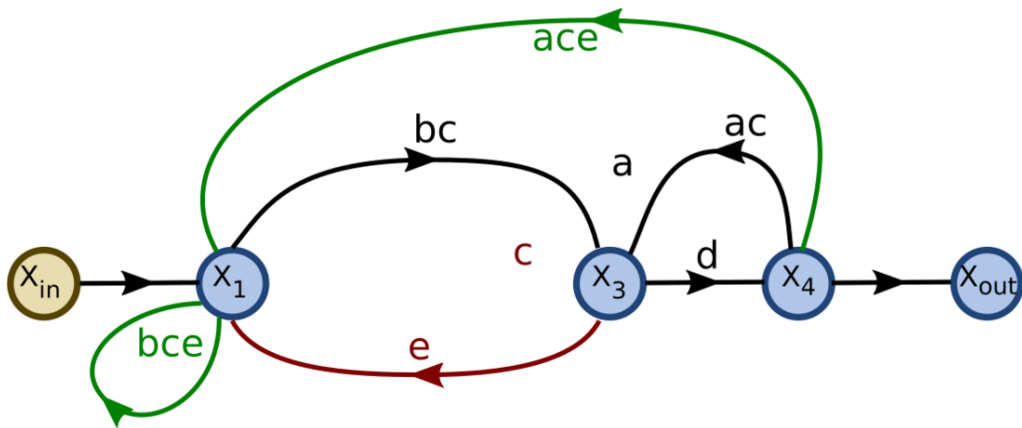
Figure 73 upright=2

$$x_1 = x_{\text{in}} + ex_3$$
$$x_1 = x_{\text{in}} + e(bcx_1 + cax_4)$$
$$x_1 = x_{\text{in}} + bcex_1 + acex_4$$
$$x_2 = bx_1 + ax_4$$
$$x_3 = bcx_1 + cax_4$$
$$x_4 = dx_3$$
$$x_{\text{out}} = x_4$$
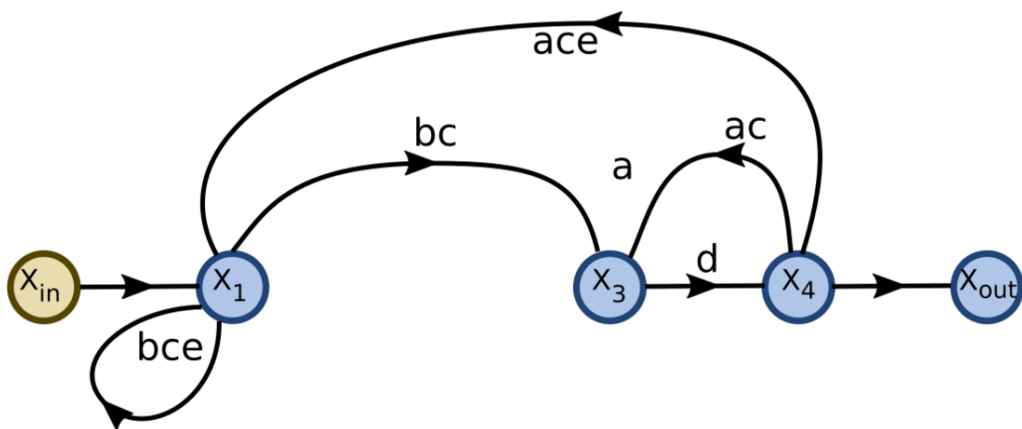
Figure 74 upright=2

Remove edge d from x3 to x4
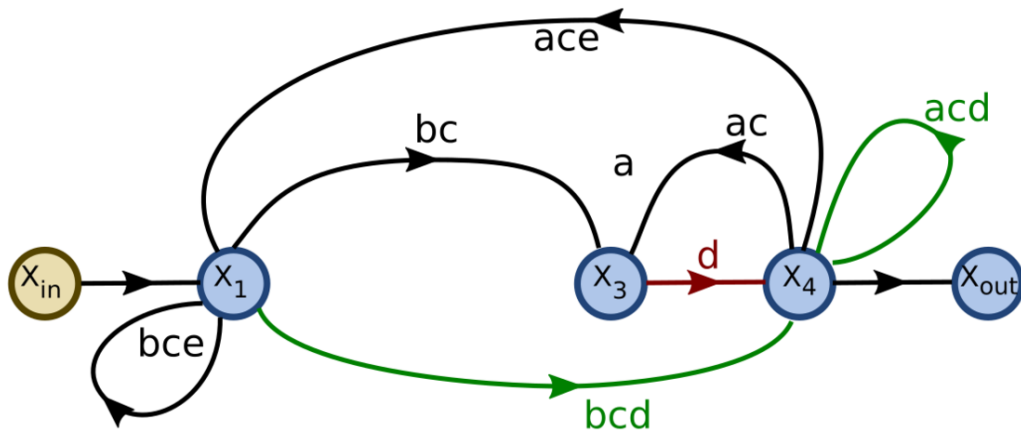
Figure 75   upright=2

$$x_1 = x_{\text{in}} + acex_4 + bcex_1$$
$$x_3 = bcx_1 + acx_4$$
$$x_4 = dx_3$$
$$x_4 = d(bcx_1 + acx_4)$$
$$x_4 = bcdx_1 + acdx_4$$
$$x_{\text{out}} = x_4$$

Node x_3 has no outflows and is not a node of interest. It is deleted along with its inflows.

$$x_1 = x_{\text{in}} + acex_4 + bcex_1$$
$$x_4 = bcdx_1 + acdx_4$$
$$x_{\text{out}} = x_4$$
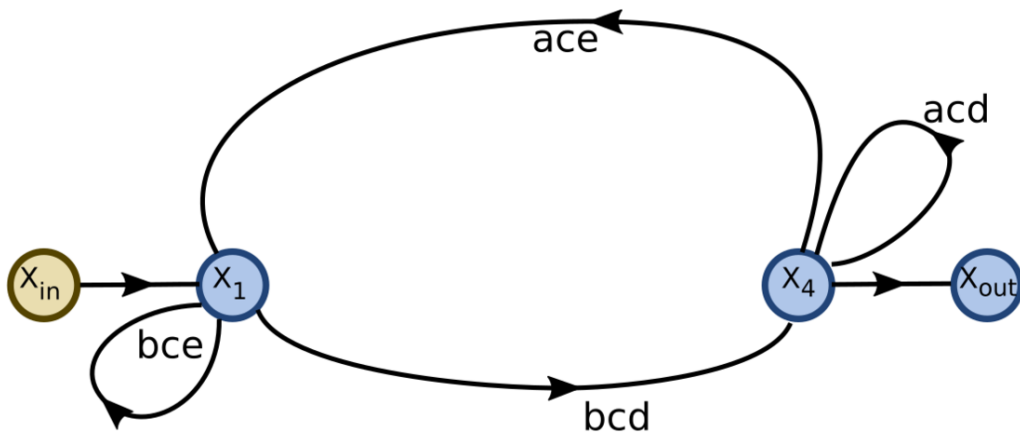
Figure 76   upright=2

Removing self-loop at x1
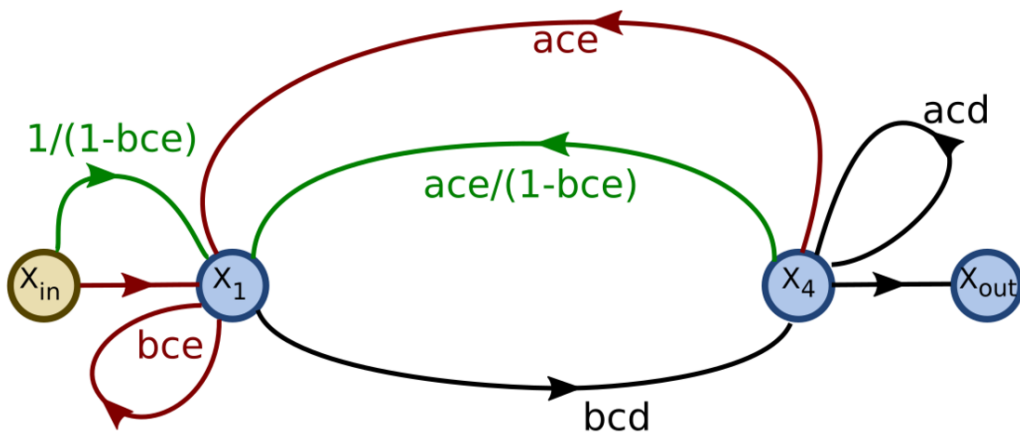
Figure 77   upright=2

$$x_1 = x_{\text{in}} + acex_4 + bcex_1$$
$$x_1(1 - bce) = x_{\text{in}} + acex_4$$
$$x_1 = \frac{1}{1 - bce}x_{\text{in}} + \frac{ace}{1 - bce}x_4$$
$$x_4 = bcdx_1 + acdx_4$$
$$x_{\text{out}} = x_4$$
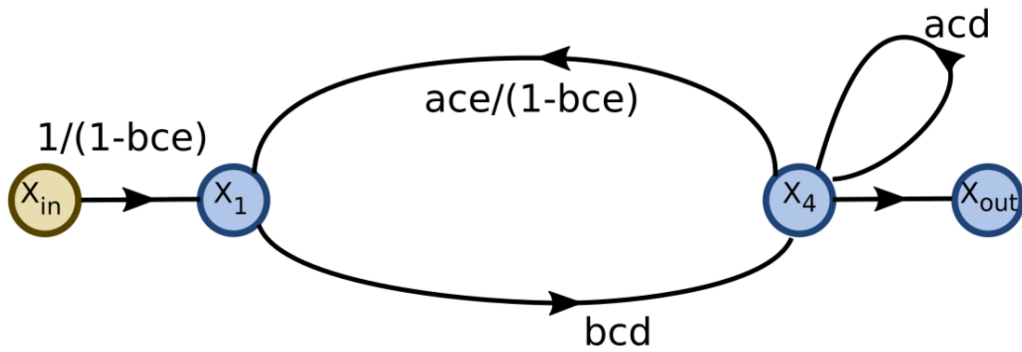
Figure 78   upright=2

Removing self-loop at x4
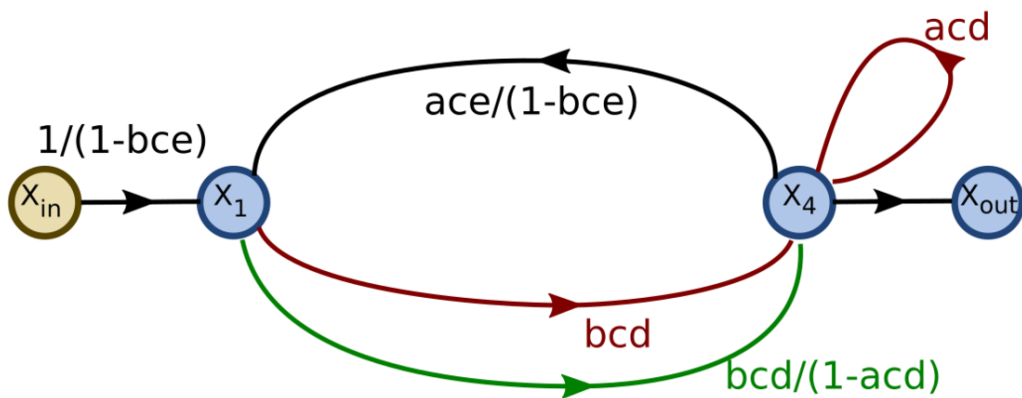
Figure 79   upright=2

$$x_1 = \frac{1}{1-bce}x_{\text{in}} + \frac{ace}{1-bce}x_4$$

$$x_4 = bcdx_1 + acdx_4$$

$$x_4(1-acd) = bcdx_1$$

$$x_4 = \frac{bcd}{1-acd}x_1$$
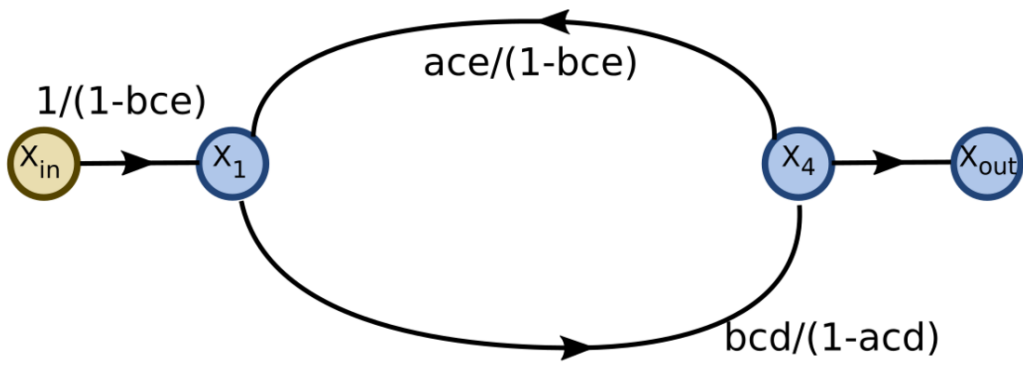
$$x_{\text{out}} = x_4$$

Figure 80 upright=2

Remove edge from x4 to x1
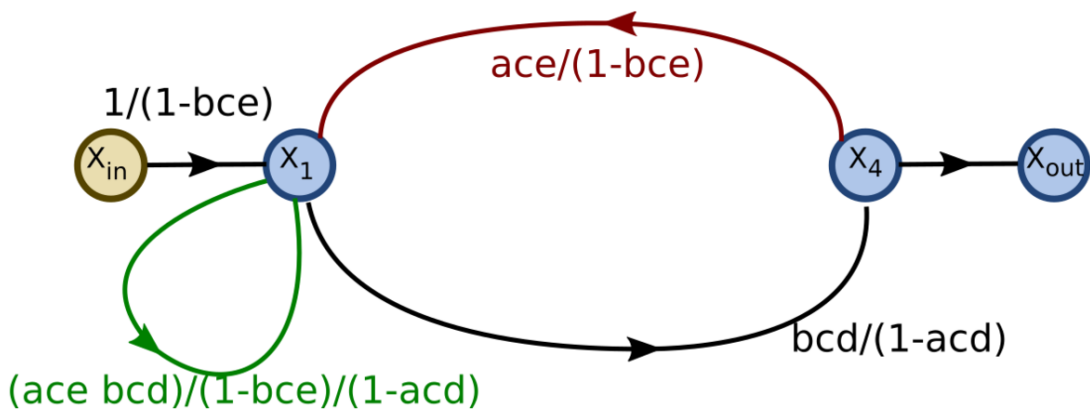
Figure 81 upright=2

$$x_1 = \frac{1}{1-bce}x_{\text{in}} + \frac{ace}{1-bce}x_4$$

$$x_1 = \frac{1}{1-bce}x_{\text{in}} + \frac{ace}{1-bce} \times \frac{bcd}{1-acd}x_1$$

$$x_4 = \frac{bcd}{1-acd}x_1$$

$$x_{\text{out}} = x_4$$

Remove outflow from x4 to x$_{\text{out}}$

$$x_1 = \frac{1}{1-bce}x_{\text{in}} + \frac{ace}{1-bce} \times \frac{bcd}{1-acd}x_1$$
$$x_{\text{out}} = \frac{bcd}{1-acd}x_1$$

x_4's outflow is then eliminated: x_{out} is connected directly to x_1 using the product of the gains from the two edges replaced.

x_4 is not a variable of interest; thus, its node and its inflows are eliminated.
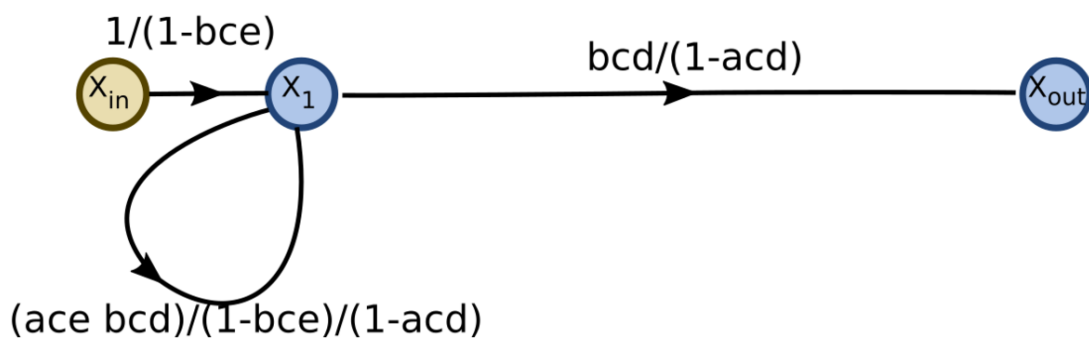
Figure 82 upright=2

Eliminating self-loop at x1

$$x_1 = \frac{1}{1-bce}x_{\text{in}} + \frac{ace}{1-bce} \times \frac{bcd}{1-acd}x_1$$
$$x_1(1 - \frac{ace}{1-bce} \times \frac{bcd}{1-acd}) = \frac{1}{1-bce}x_{\text{in}}$$
$$x_1 = \frac{1}{(1-bce) \times (1 - \frac{ace}{1-bce} \times \frac{bcd}{1-acd})}x_{\text{in}}$$
$$x_{\text{out}} = \frac{bcd}{1-acd}x_1$$
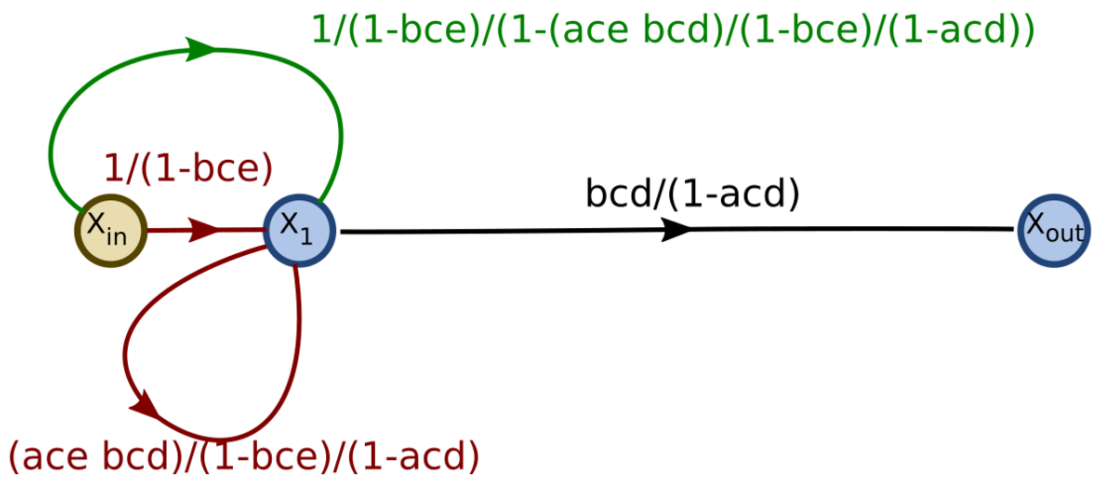
Figure 83   upright=2
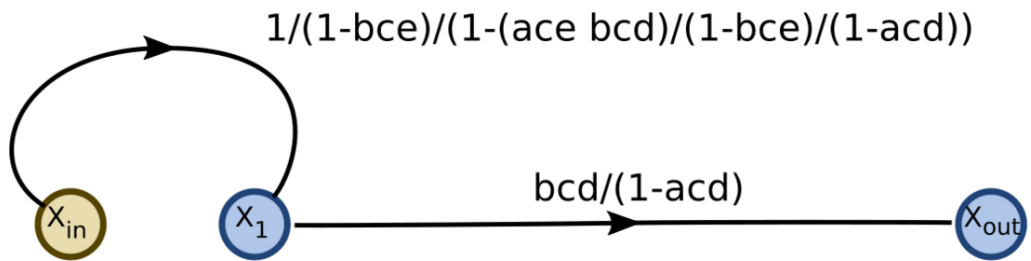
Figure 84   upright=2

Eliminating outflow from x1, then eliminating x1 and its inflows
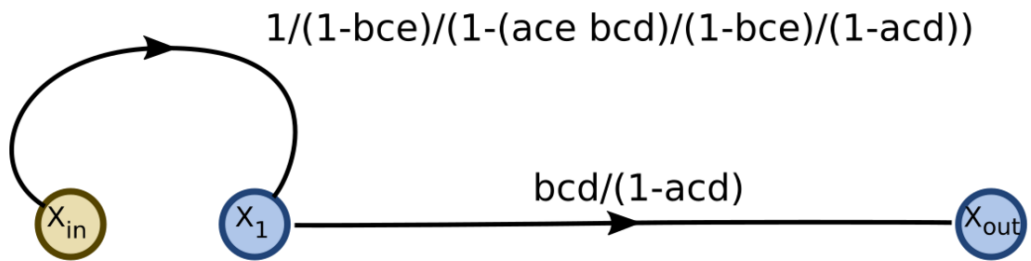
Figure 85   upright=2

$$x_1 = \frac{1}{(1 - bce) \times (1 - \frac{ace}{1-bce} \times \frac{bcd}{1-acd})} x_{\text{in}}$$

$$x_{\text{out}} = \frac{bcd}{1 - acd} x_1$$

$$x_{\text{out}} = \frac{bcd}{1 - acd} \times \frac{1}{(1 - bce) \times (1 - \frac{ace}{1-bce} \times \frac{bcd}{1-acd})} x_{\text{in}}$$

x_1 is not a variable of interest; x_1 and its inflows are eliminated

$$x_{\text{out}} = \frac{bcd}{1 - acd} \times \frac{1}{(1 - bce) \times (1 - \frac{ace}{1-bce} \times \frac{bcd}{1-acd})} x_{\text{in}}$$

bcd/(1-acd)/(1-bce)/(1-(ace bcd)/(1-bce)/(1-acd))
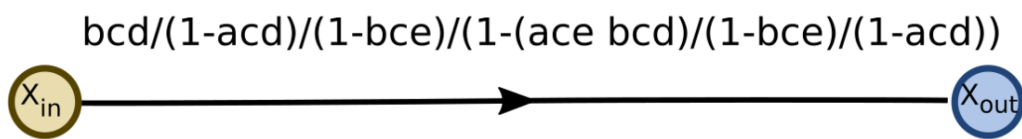
Figure 86   upright=2

Simplifying the gain expression

$$x_{\text{out}} = \frac{-bcd}{bce + acd - 1} x_{\text{in}}$$

- bcd / (bce + acd - 1)
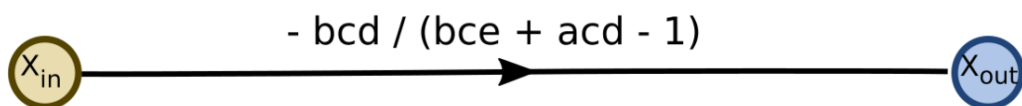
Figure 87   upright=2

22.2.2 Solving a signal-flow graph by systematic reduction: Three equations in three unknowns

This example shows how a system of three equations in three unknowns is solved using systematic reduction rules. The independent variables are y_1, y_2, y_3. The dependent variables are x_1, x_2, x_3. The coefficients are labeled c_{jk}. The steps for solving x_1 follow:
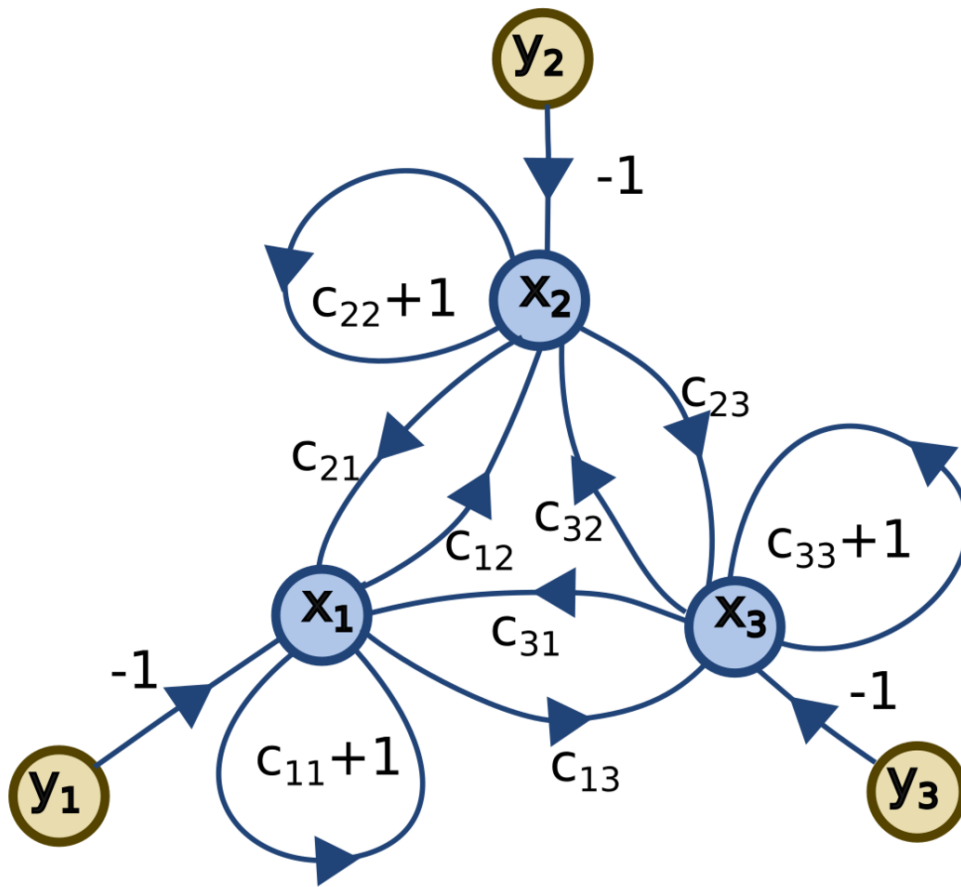
Figure 88 upright=2
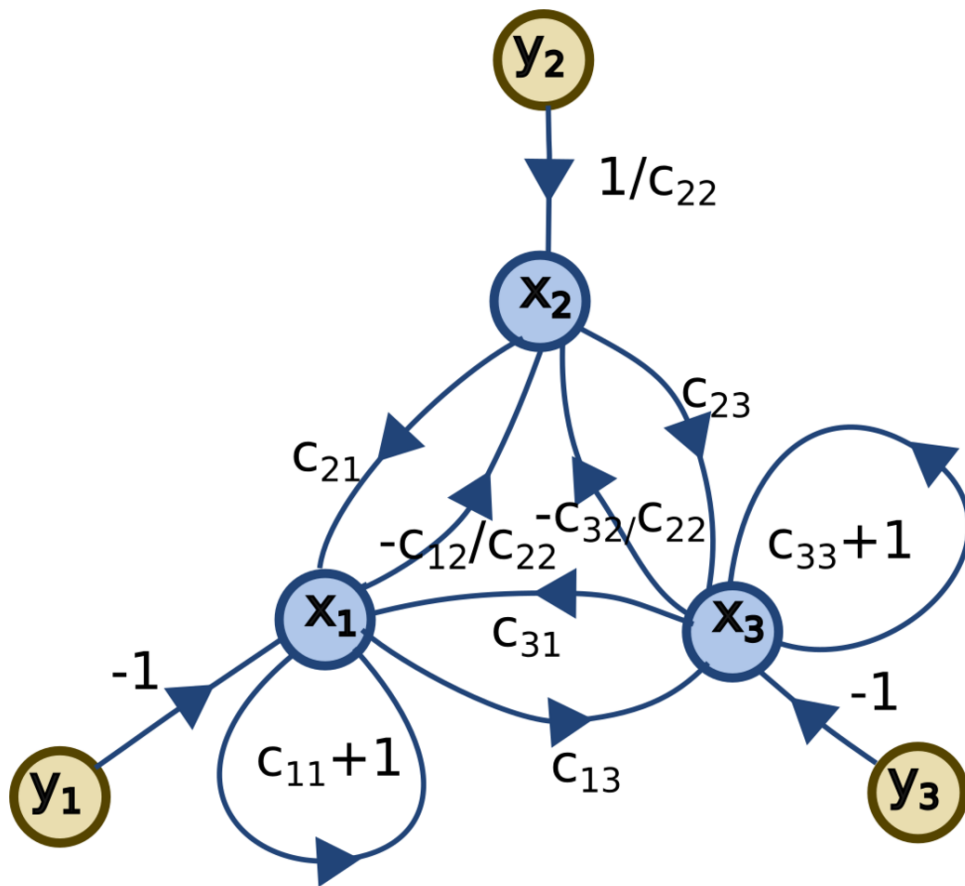
Figure 89 upright=2
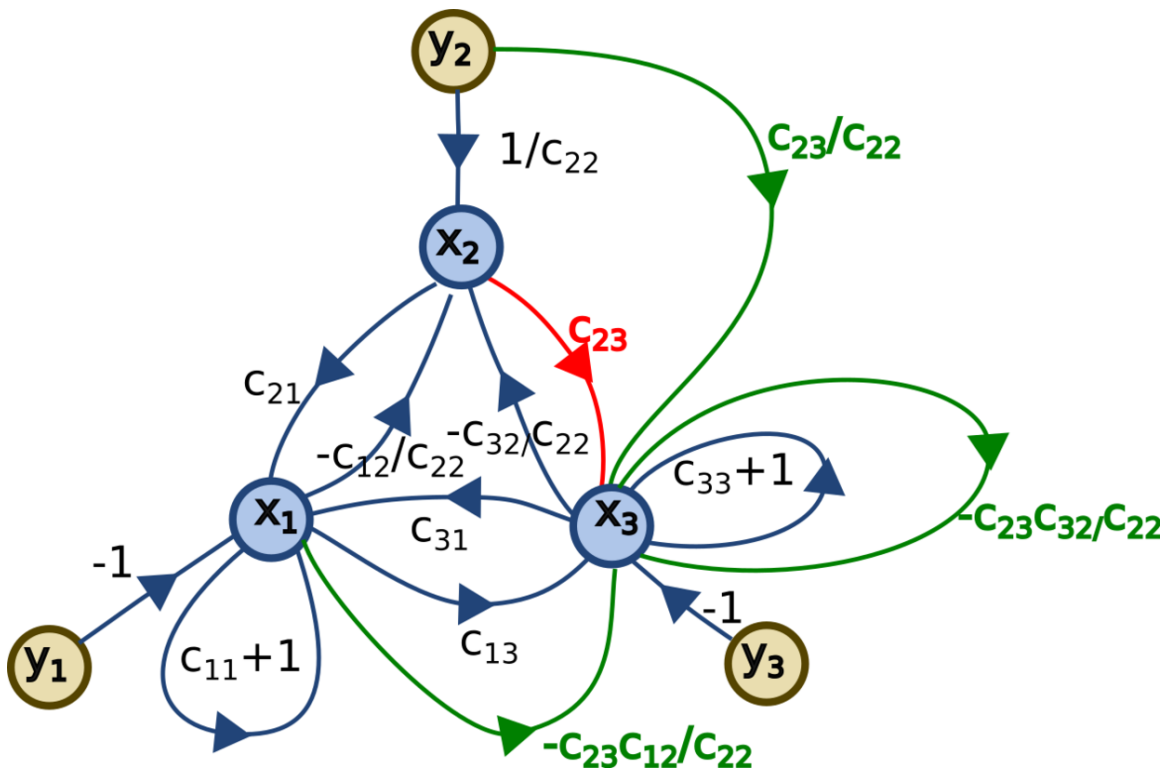
Figure 90   upright=2
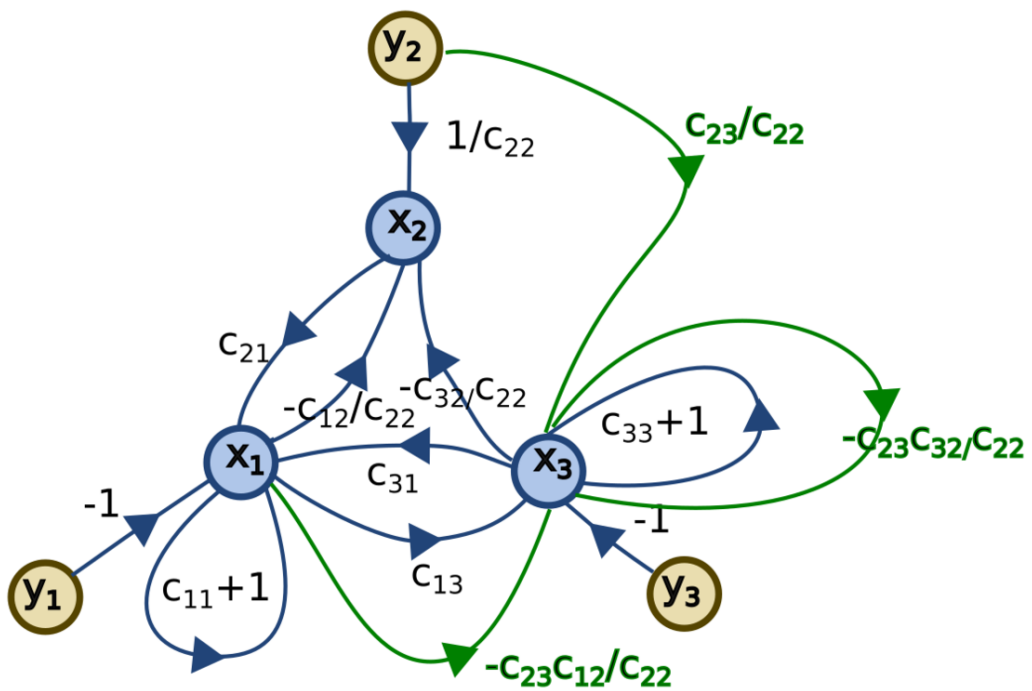
Figure 91   upright=2
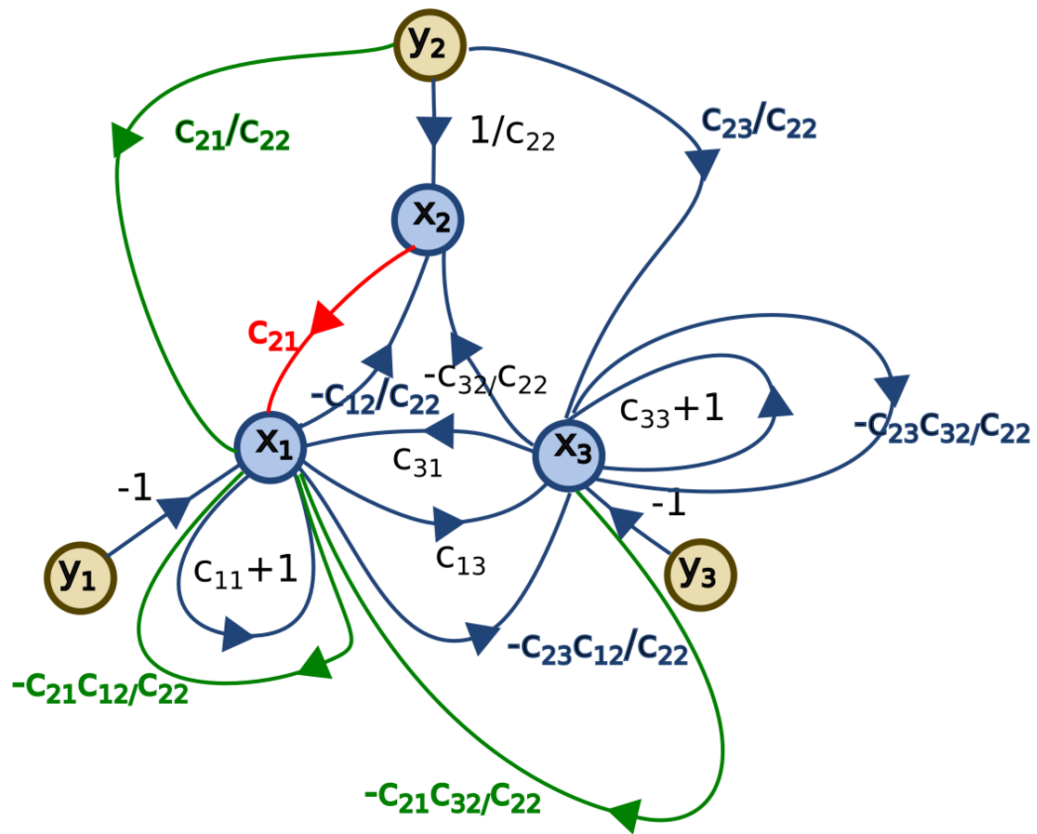
Figure 92   upright=2
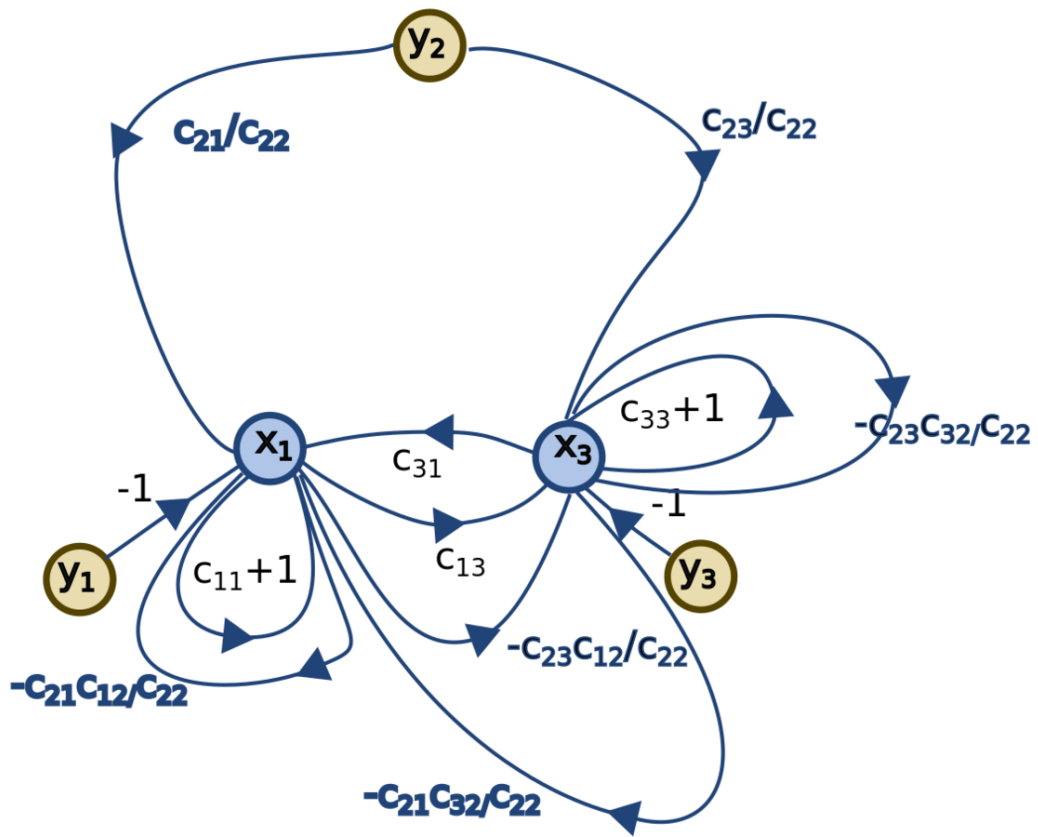
Figure 93 upright=2
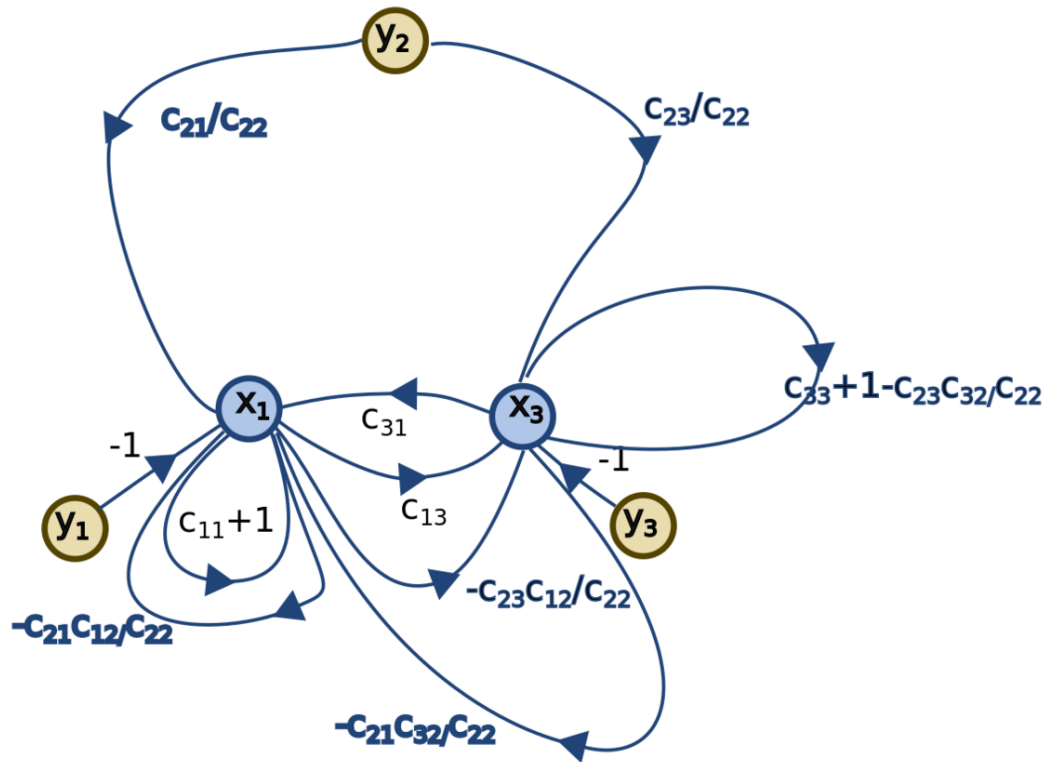
Figure 94   upright=2
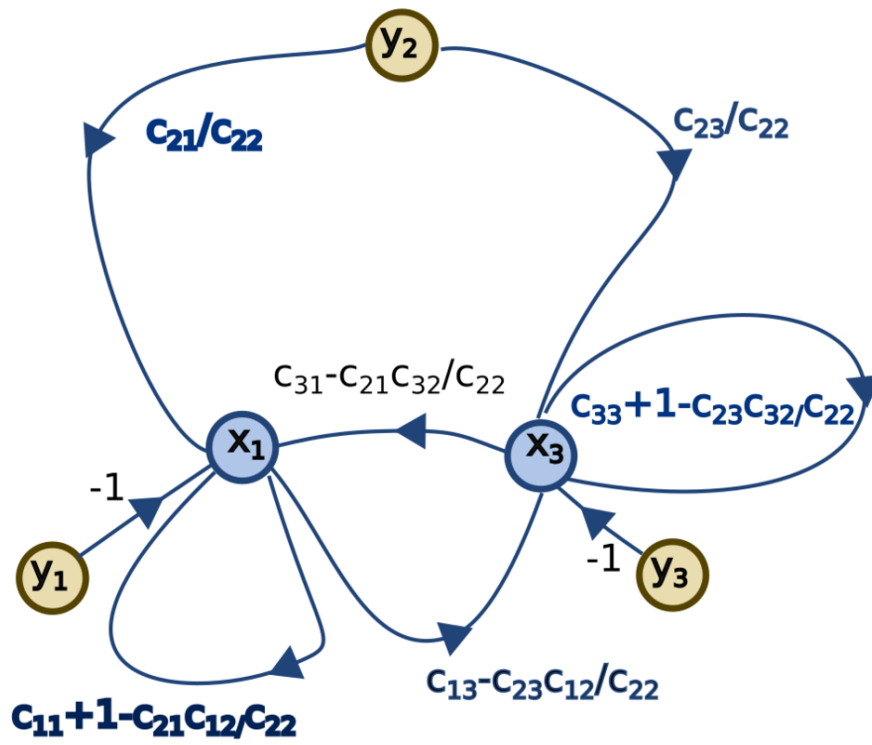
Figure 95 upright=2
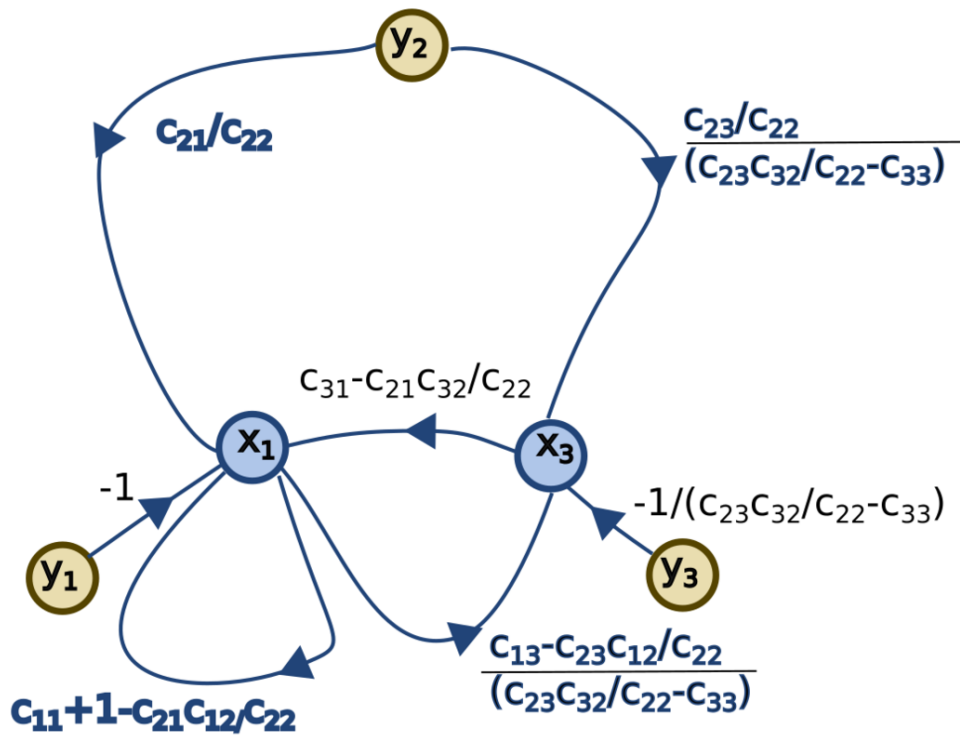
Figure 96   upright=2
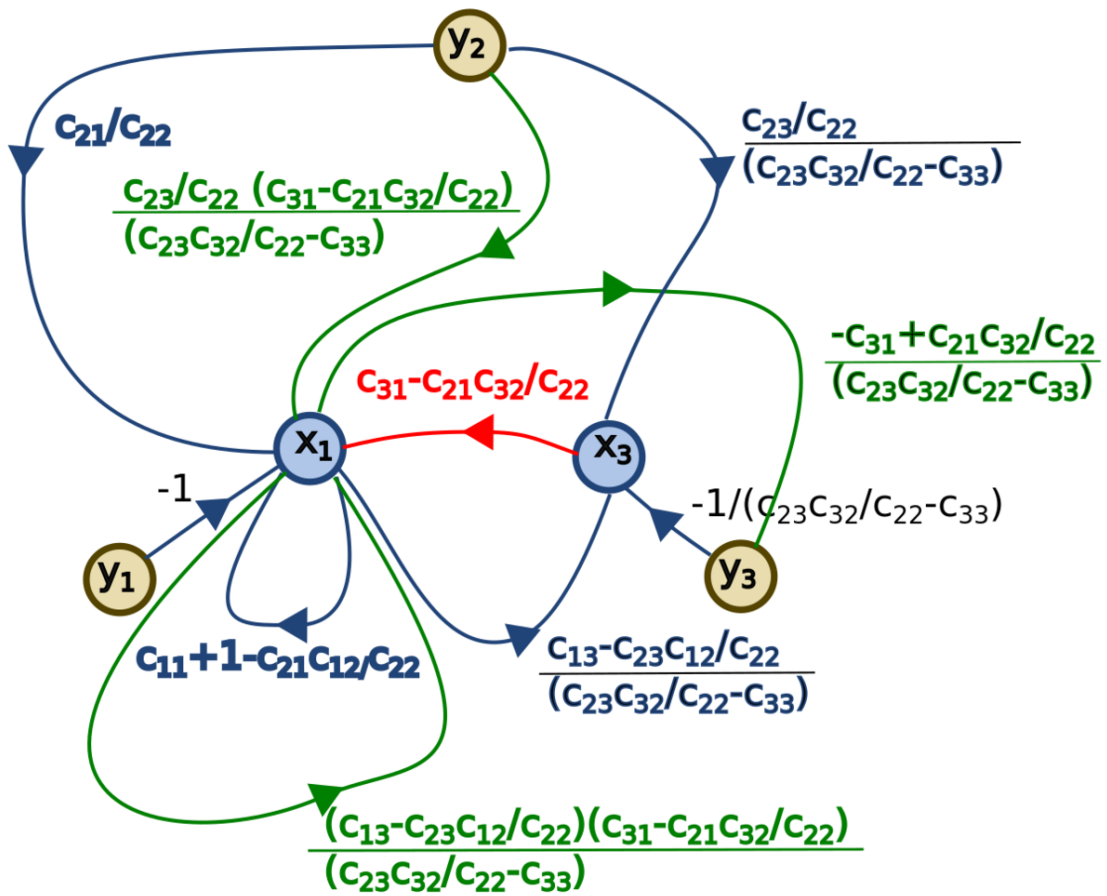
Figure 97 upright=2
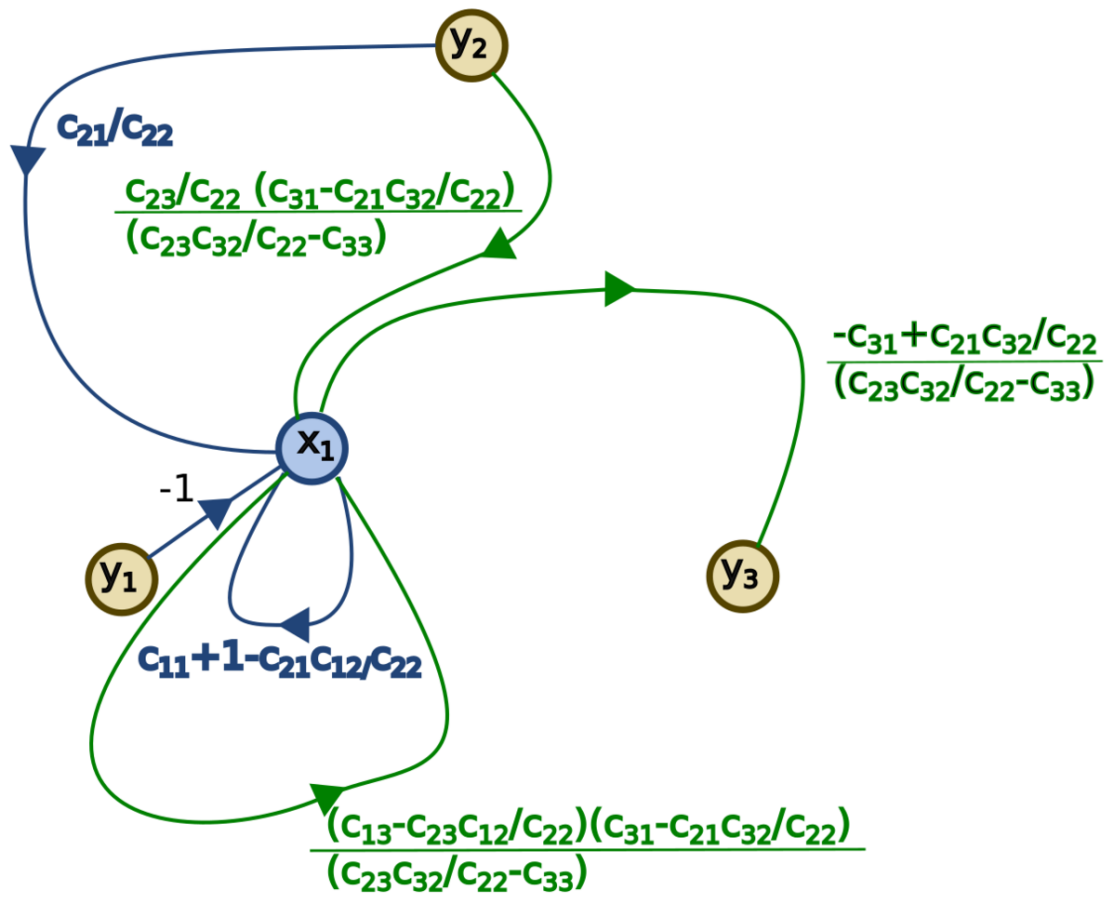
Figure 98   upright=2
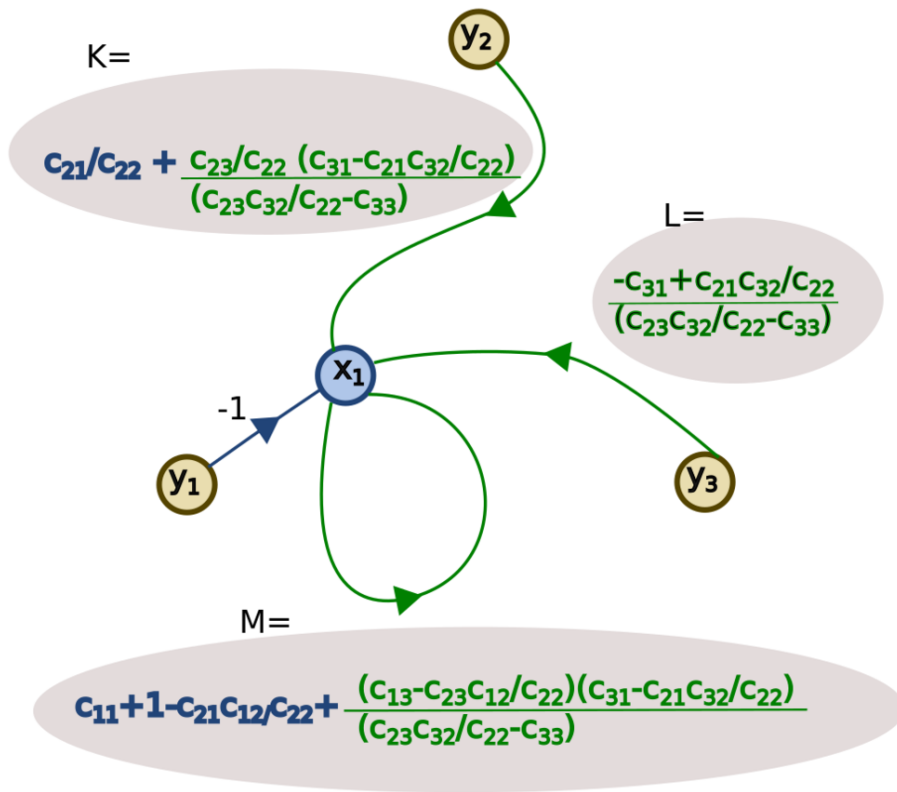
Figure 99   upright=2
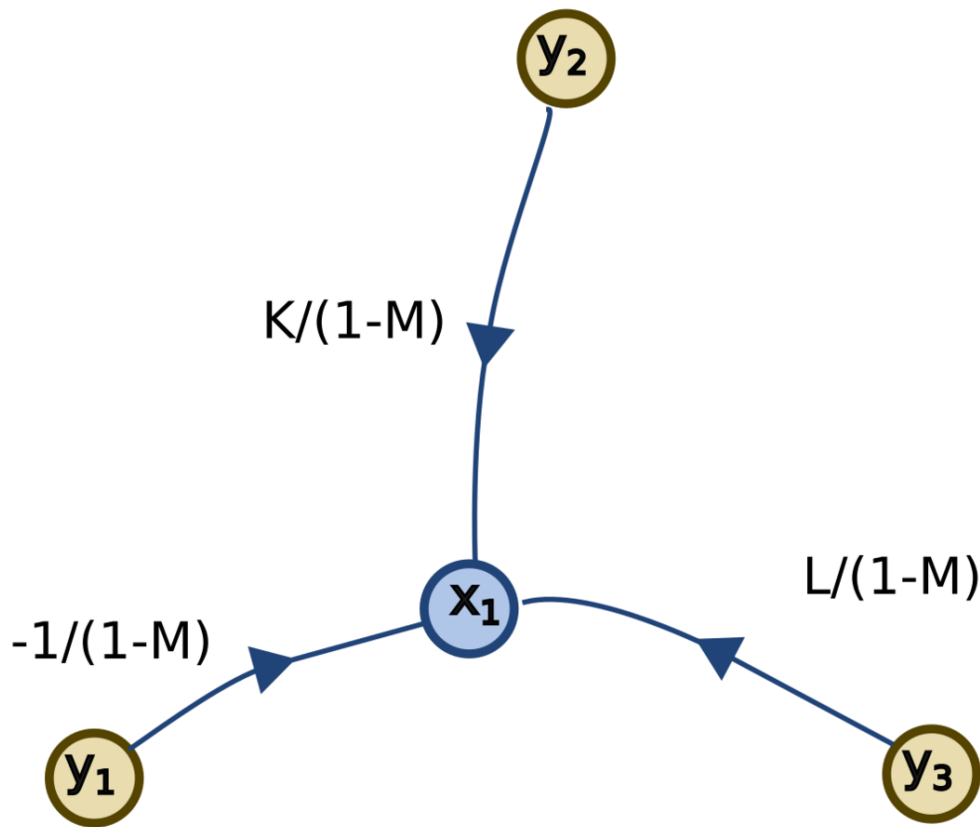
Figure 100 upright=2

22.3 Electrical engineering: Construction of a flow graph for a RC circuit
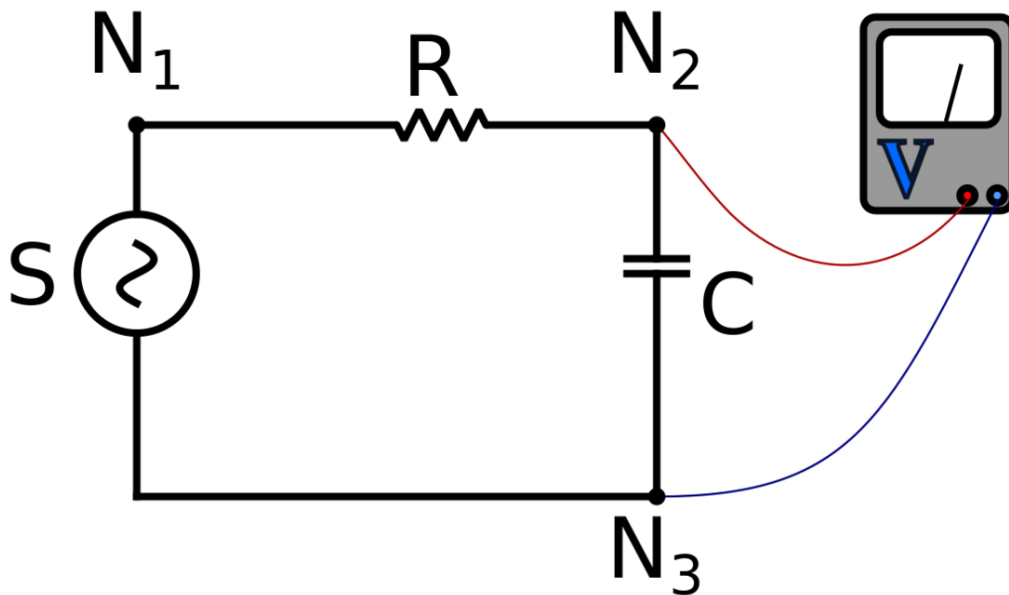
Figure 101

This illustration shows the physical connections of the circuit. Independent voltage source S is connected in series with a resistor R and capacitor C. The example is developed from the physical circuit equations and solved using signal-flow graph techniques. Polarity is important:

- S is a source with the positive terminal at N_1 and the negative terminal at N_3
- R is a resistor with the positive terminal at N_1 and the negative terminal at N_2
- C is a capacitor with the positive terminal at N_2 and the negative terminal at N_3.

The unknown variable of interest is the voltage across capacitor **C**.

Approach to the solution:

- Find the set of equations from the physical network. These equations are acausal in nature.
  - Branch equations for the capacitor and resistor. The equations will be developed as transfer functions using Laplace transforms.
  - Kirchhoff's voltage and current laws
- Build a signal-flow graph from the equations.
- Solve the signal-flow graph.

22.3.1 Branch equations
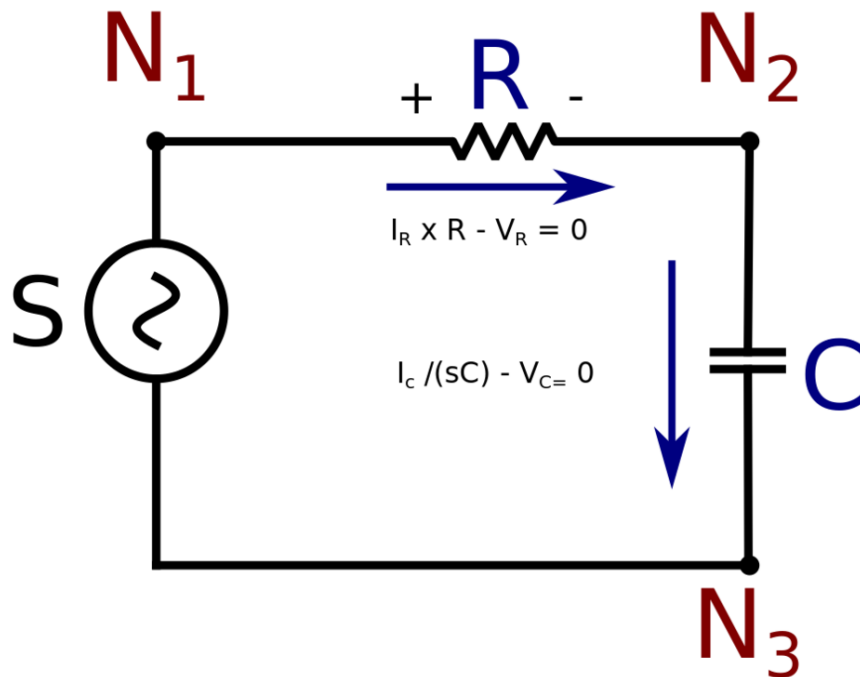
Figure 102

The branch equations are shown for R and C.

Resistor R (Branch equation B_R)

The resistor's branch equation in the time domain is:

$$V_R(t) = RI_R(t)$$

In the Laplace-transformed signal space:

$$V_R(s) = RI_R(s)$$

Capacitor C (Branch equation B_C)

The capacitor's branch equation in the time domain is:

$$V_C(t) = \frac{Q_C(t)}{C} = \frac{1}{C}\int_{t_0}^{t} I_C(\tau)\mathrm{d}\tau + V_C(t_0)$$

Assuming the capacitor is initially discharged, the equation becomes:

$$V_C(t) = \frac{Q_C(t)}{C} = \frac{1}{C}\int_{t_0}^{t} I_C(\tau)\mathrm{d}\tau$$

Taking the derivative of this and multiplying by C yields the derivative form:

$$I_C(t) = \frac{\mathrm{d}Q(t)}{\mathrm{d}t} = C\frac{\mathrm{d}V_C(t)}{\mathrm{d}t}$$

In the Laplace-transformed signal space:

$$I_C(s) = V_C(s)sC$$

22.3.2 Kirchhoff's laws equations
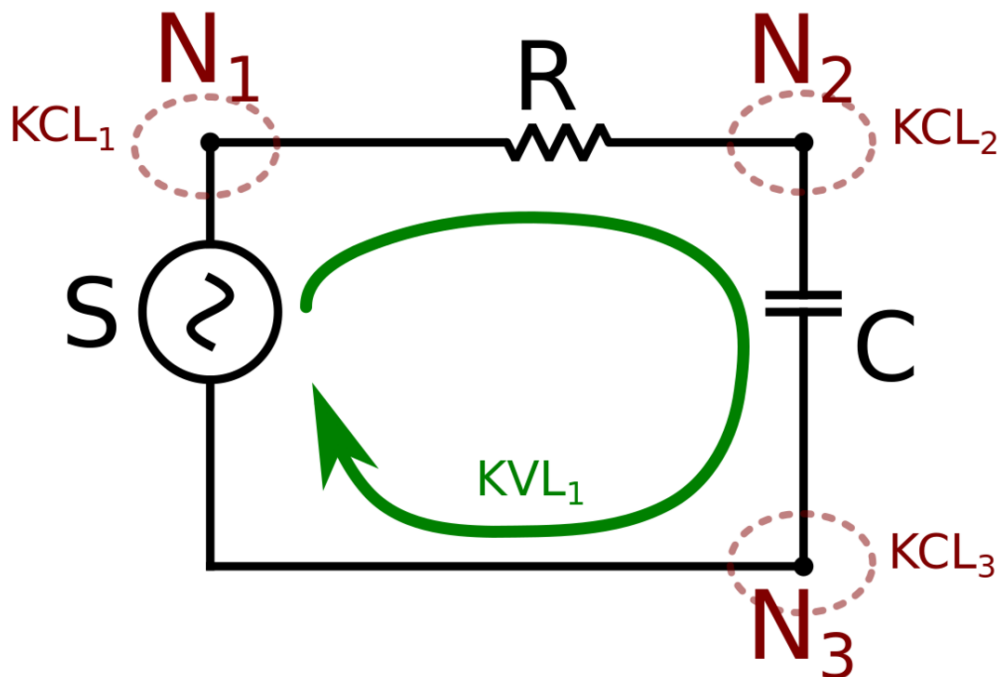
Figure 103

Kirchhoff's Voltage Law equation KVL_1

This circuit has only one independent loop. Its equation in the time domain is:

$$V_R(t) + V_C(t) - V_S(t) = 0$$

In the Laplace-transformed signal space:

$$V_R(s) + V_C(s) - V_S(s) = 0$$

Kirchhoff's Current Law equations $\text{KCL}_1, KCL_2, KCL_3$

The circuit has three nodes, thus three Kirchhoff's current equations (expresses here as the currents flowing from the nodes):

$$
\begin{aligned}
I_S(t) + I_R(t) &= 0 \quad (\text{KCL}_1) \\
I_C(t) - I_R(t) &= 0 \quad (\text{KCL}_2) \\
I_S(t) - I_C(t) &= 0 \quad (\text{KCL}_3)
\end{aligned}
$$

In the Laplace-transformed signal space:

$$
\begin{aligned}
I_S(s) + I_R(s) &= 0 \quad (\text{KCL}_1) \\
I_C(s) - I_R(s) &= 0 \quad (\text{KCL}_2) \\
I_S(s) - I_C(s) &= 0 \quad (\text{KCL}_3)
\end{aligned}
$$

A set of independent equations must be chosen. For the current laws, it is necessary to drop one of these equations. In this example, let us choose KCL_1, KCL_2.

22.3.3 Building the signal-flow graph

We then look at the inventory of equations, and the signals that each equation relates:

Equation	Signals
B_C	V_C, I_C
B_R	V_R, I_R
KVL_1	V_R, V_C, V_S
KCL_1	I_S, I_R
KCL_2	I_R, I_C

The next step consists in assigning to each equation a signal that will be represented as a node. Each independent source signal is represented in the signal-flow graph as a source node, therefore no equation is assigned to the independent source V_S. There are many possible valid signal flow graphs from this set of equations. An equation must only be used once, and the variables of interest must be represented.

Equation	Signals	Assigned signal node
B_C	V_C, I_C	I_C
B_R	V_R, I_R	V_R
KVL_1	V_R, V_C, V_S	V_C
KCL_1	I_S, I_R	I_S
KCL_2	I_R, I_C	I_R

22.3.4 The resulting flow graph is then drawn
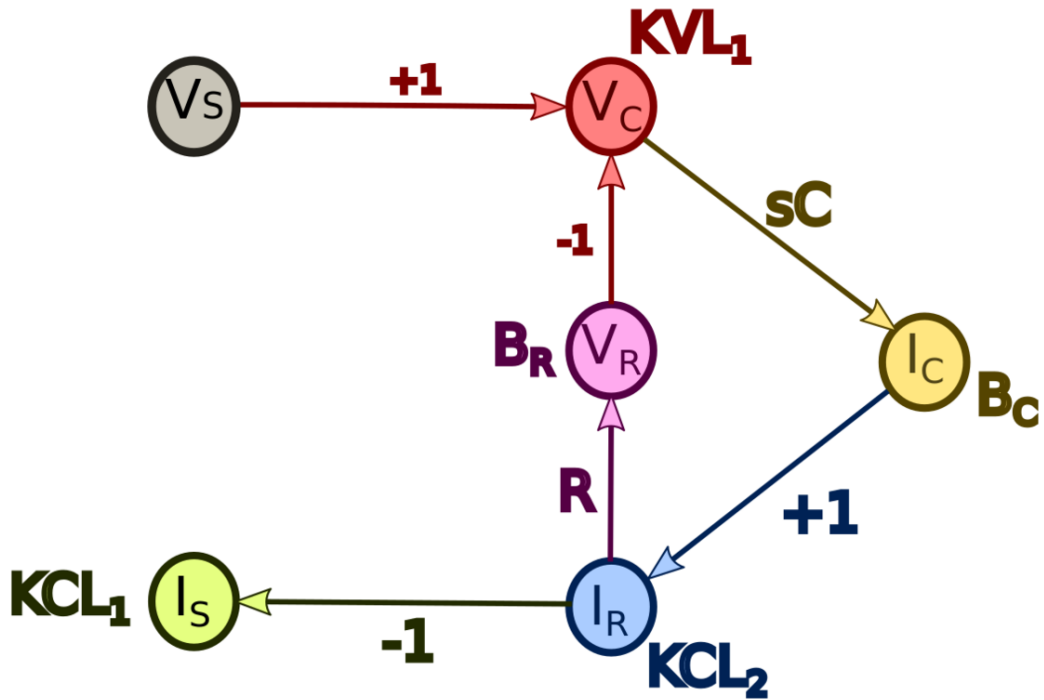
Figure 104

The next step consists in solving the signal-flow graph.

Using either Mason or systematic reduction, the resulting signal flow graph is:
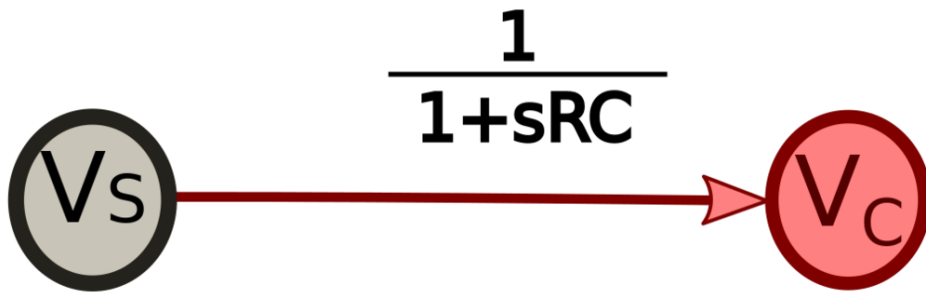
Figure 105

22.3.5 Mechatronics example
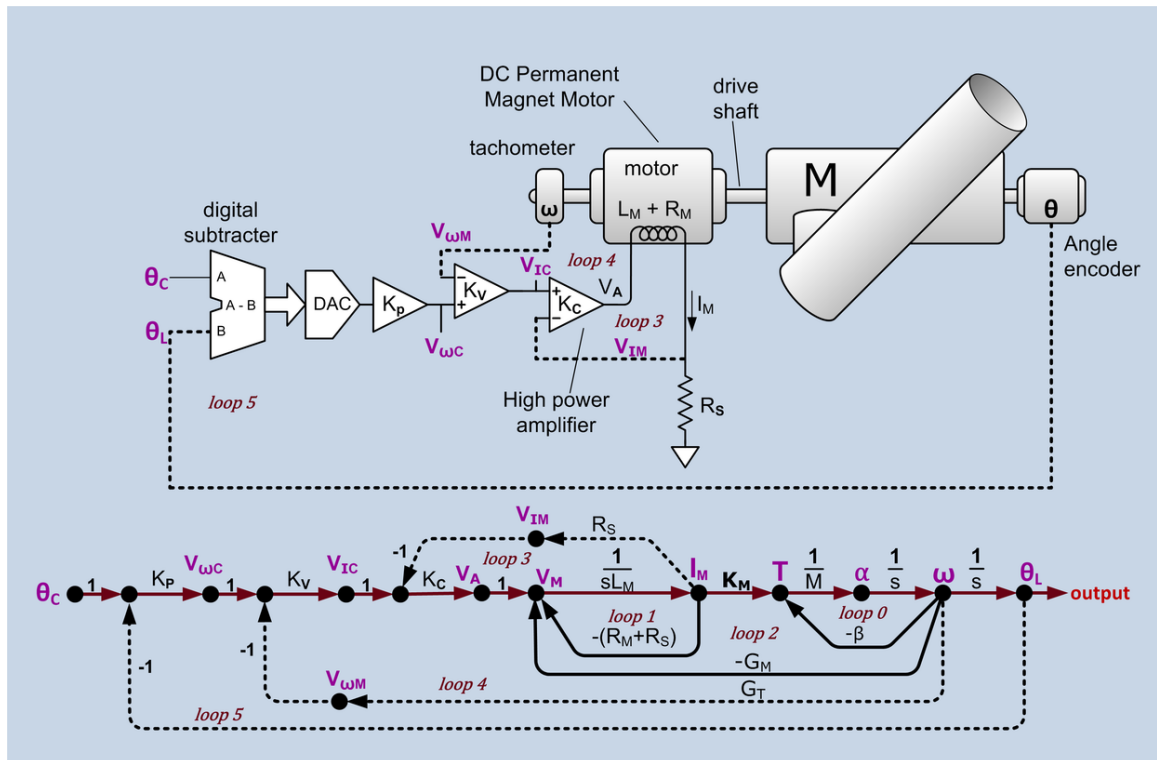
Figure 106 Angular position servo and signal flow graph. θ_C = desired angle command, θ_L = actual load angle, K_P = position loop gain, $V_{\omega C}$ = velocity command, $V_{\omega M}$ = motor velocity sense voltage, K_V = velocity loop gain, V_{IC} = current command, V_{IM} = current sense voltage, K_C = current loop gain, V_A = power amplifier output voltage, L_M = motor inductance, V_M = voltage across motor inductance, I_M = motor current, R_M = motor resistance, R_S = current sense resistance, K_M = motor torque constant (Nm/amp) , T = torque, M = momment of inertia of all rotating components α = angular acceleration, ω = angular velocity, β = mechanical damping, G_M = motor back EMF constant, G_T = tachometer conversion gain constant,. There is one forward path (shown in a different color) and six feedback loops. The drive shaft assumed to be stiff enough to not treat as a spring. Constants are shown in black and variables in purple.

23 Bode Plots

23.1 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.
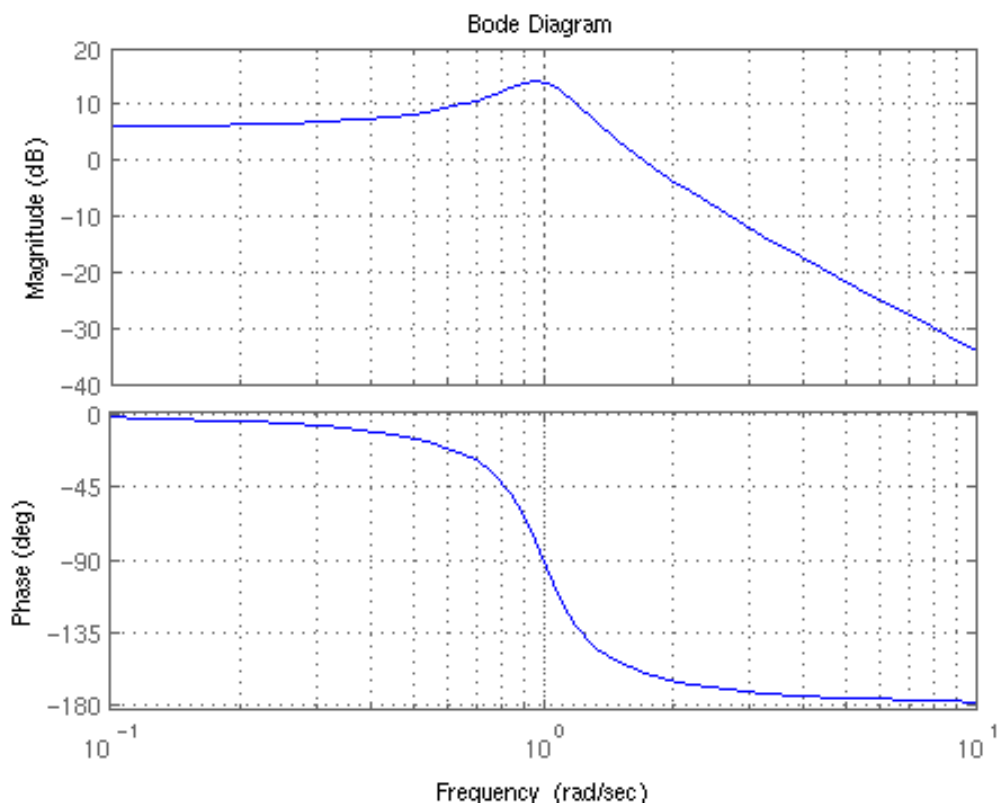
Figure 107   An example of a Bode magnitude and phase plot set. The Magnitude plot is typically on the top, and the Phase plot is typically on the bottom of the set.

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 decreases as you move to the right on the graph. (note that this

199

description doesn't match the chart above... there are 10 tickmarks per decade, not one, but since it is a log chart they are not evenly spaced).

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).

23.1.1 Decibels

A **Decibel** is a ratio between two numbers on a logarithmic scale. To express a ratio between two numbers (A and B) as a decibel we apply the following formula for numbers that represent amplitudes (numbers that represent a power measurement use a factor of 10 rather than 20):

$$dB = 20 \log\left(\frac{A}{B}\right)$$

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:

$$dB = 20 \log(C)$$

23.1.2 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 transit 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.

23.1.3 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**.

23.1.4 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 experience the largest change in direction.

Break points are sometimes also called "break frequencies", "cutoff points", or "corner points".

23.2 Bode Gain Plots

Bode Gain Plots, or **Bode Magnitude Plots** display the ratio of the system gain at each input frequency.

23.2.1 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.

23.2.2 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 $\omega = 0$ 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 $\omega = 0$ is 0 dB/Decade.

From each pole break point, the slope of the line decreases by 20 dB/Decade. The line is straight until it reaches the next break point. From each zero break point the slope of the line increases by 20 dB/Decade. Double, triple, or higher amounts of repeat poles and zeros affect the gain by multiplicative amounts. Here are some examples:

- 2 poles: -40 dB/Decade
- 10 poles: -200 dB/Decade
- 5 zeros: +100 dB/Decade

23.3 Bode Phase Plots

Bode phase plots are plots of the phase shift to an input waveform dependent 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:

$$\angle T(j\omega) = \tan^{-1}\left(\frac{X}{R}\right)$$

23.4 Bode Procedure

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 4

Use $\omega = 0$ to find the starting magnitude.

Step 5

The locations of every pole and every zero are called **break points**. At a zero break-point, 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 straight-line graph by 3dB. A zero is +3dB over the straight line, and a pole is -3dB below the straight line.

Step 7

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.

23.5 Examples

23.5.1 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 dependent 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:
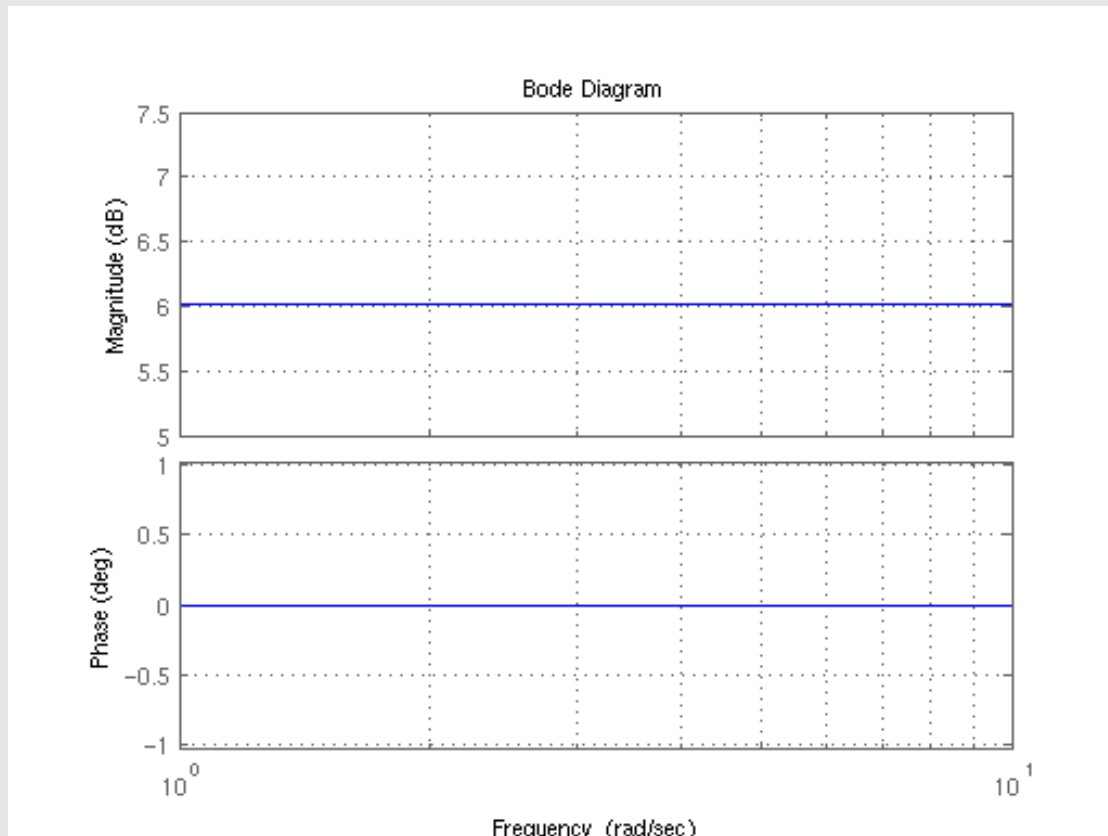
Figure 108

23.5.2 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 $\omega \to 0$, 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 calculation by plugging in $\omega \to 1$ that the second term drops out, and the magnitude at that point is 6dB. We now have the slope of the line, and a point that it intersects, and we can draw the graph:
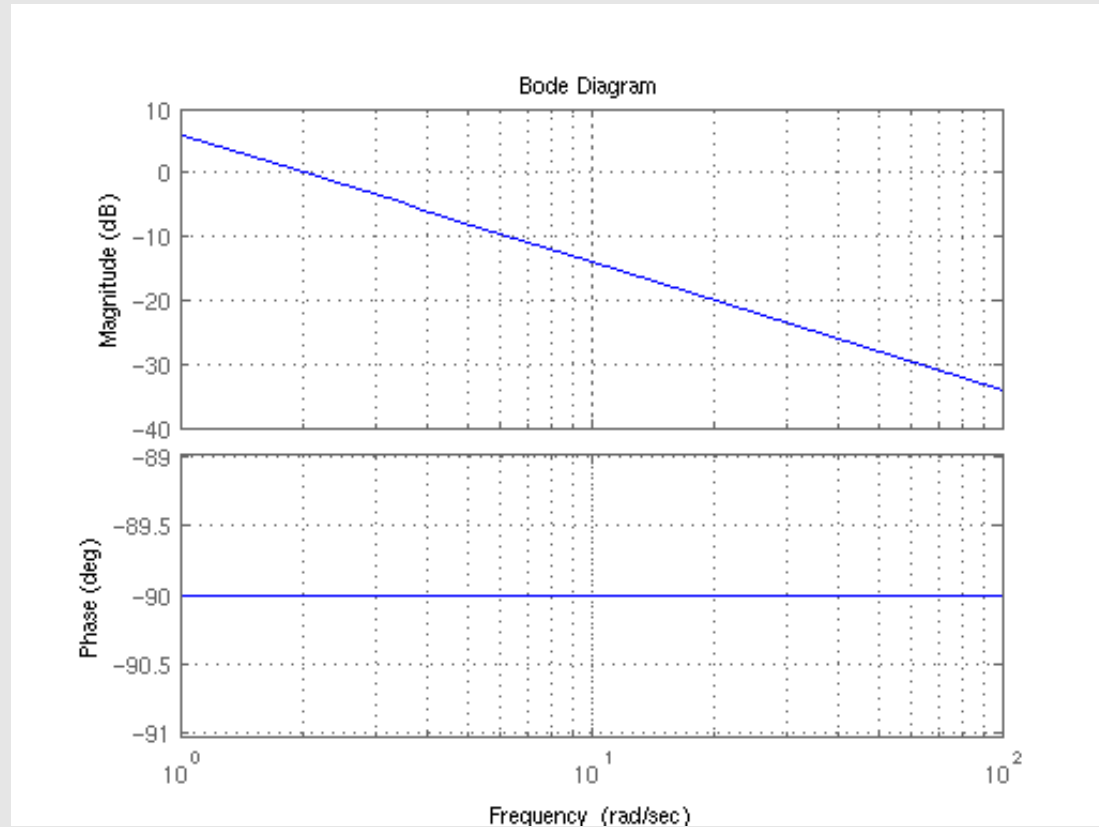
Figure 109

23.5.3 Example: Differentiator

$T(j\omega) = 2j\omega$
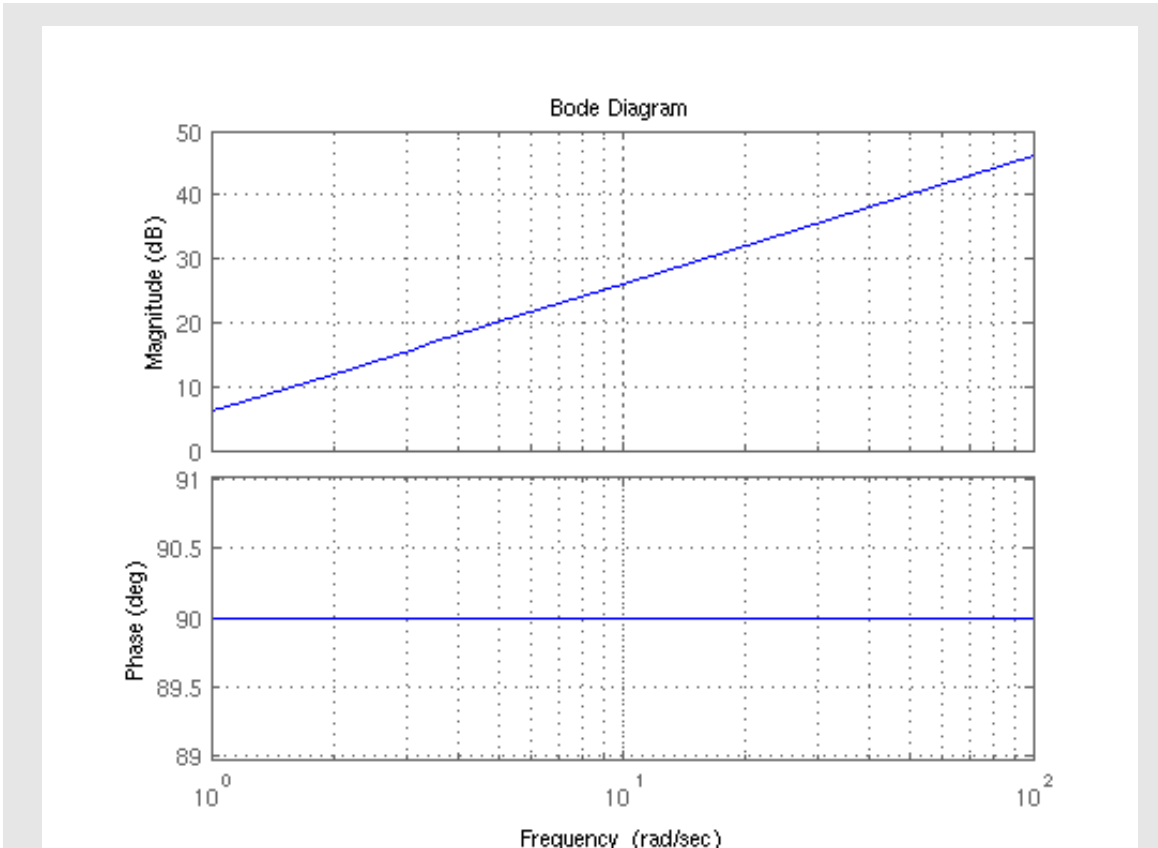
Figure 110

23.5.4 Example: 1st Order, Low-pass Filter (1 Break Point)

$T(j\omega) = \frac{-21.4}{j\omega+144}$

206
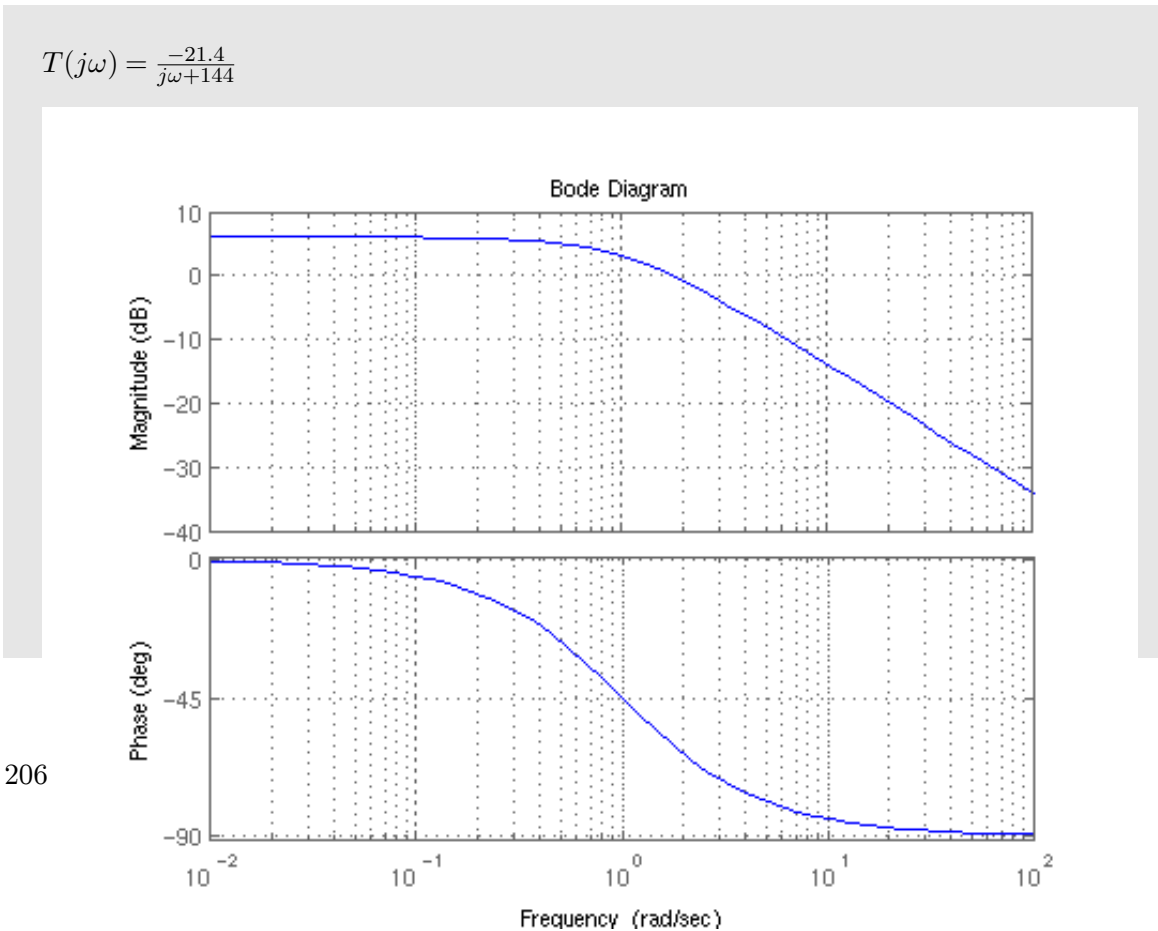
Figure 111

23.6 Further reading

- Bode plot online generator (real and asymptotic)[1]
- Circuit Theory/Bode Plots[2]
- Bode Plots on ControlTheoryPro.com[3]

1   http://www.onmyphd.com/?p=bode.plot
2   https://en.wikibooks.org/wiki/Circuit%20Theory%2FBode%20Plots
3   http://wikis.controltheorypro.com/index.php?title=Bode_Plot

24 Nichols Charts

24.1 Nichols Charts

This page will talk about the use of Nichols charts to analyze frequency-domain characteristics of control systems.

w:Nichols plot[1]

---

1   https://en.wikipedia.org/wiki/Nichols%20plot

25 Stability

25.1 Stability

When a system is unstable, the output of the system may be infinite even though the input to the system was finite. This causes a number of practical problems. For instance, a robot arm controller that is unstable may cause the robot to move dangerously. Also, systems that are unstable often incur a certain amount of physical damage, which can become costly. Nonetheless, many systems are inherently unstable - a fighter jet, for instance, or a rocket at liftoff, are examples of naturally unstable systems. Although we can design controllers that stabilize the system, it is first important to understand what stability is, how it is determined, and why it matters.

The chapters in this section are heavily mathematical, and many require a background in linear differential equations. Readers without a strong mathematical background might want to review the necessary chapters in the Calculus[1] and Ordinary Differential Equations[2] books (or equivalent) before reading this material.

For most of this chapter we will be assuming that the system is linear, and can be represented either by a set of transfer functions or in state space. Linear systems have an associated characteristic polynomial, and this polynomial tells us a great deal about the stability of the system. Negativeness of any coefficient of a characteristic polynomial indicates that the system is either unstable or at most marginally stable. If any coefficient is zero/negative then we can say that the system is unstable. It is important to note, though, that even if all of the coefficients of the characteristic polynomial are positive the system may still be unstable. We will look into this in more detail below.

25.2 BIBO Stability

A system is defined to be **BIBO Stable** if every bounded input to the system results in a bounded output over the time interval $[t_0, \infty)$. This must hold for all initial times t_o. So long as we don't input infinity to our system, we won't get infinity output.

A system is defined to be **uniformly BIBO Stable** if there exists a positive constant k that is independent of t_0 such that for all t_0 the following conditions:

$$(t)\| \leq 1$$

---

1   https://en.wikibooks.org/wiki/Calculus
2   https://en.wikibooks.org/wiki/Ordinary%20Differential%20Equations

$$t \geq t_0$$

implies that

$$(t)\| \leq k$$

There are a number of different types of stability, and keywords that are used with the topic of stability. Some of the important words that we are going to be discussing in this chapter, and the next few chapters are: **BIBO Stable**, **Marginally Stable**, **Conditionally Stable**, **Uniformly Stable**, **Asymptotically Stable**, and **Unstable**. All of these words mean slightly different things.

25.3 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:

$$-M < x \leq M$$

We apply the input x, and the arbitrary boundaries 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:

$$y_{-M} \leq y_x \leq y_M$$

If this condition is satisfied, then the system is BIBO stable.

A SISO linear time-invariant (LTI) system is BIBO stable if and only if $g(t)$ is absolutely integrable from $[0,\infty]$ or from:

$$\int_0^\infty |g(t)|\, dt \leq M < \infty$$

25.3.1 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:

$y_M = \lim_{M \to \infty} \frac{2}{M} = 0^+$

So now, we can write out our inequality:

$y_{-M} \leq y_x \leq y_M$

$0^- \leq x < 0^+$

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.

25.4 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-plane (RHP), the system becomes unstable. When the poles of the system are located in the left-half plane (LHP) and the system is not improper, 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.

If the system is a multivariable, or a MIMO system, then the system is stable if and only if *every pole of every transfer function* in the transfer function matrix has a negative real part and every transfer function in the transfer function matrix is not improper. For these systems, it is possible to use the Routh-Hurwitz, Root Locus, and Nyquist methods described later, but these methods must be performed once for each individual transfer function in the transfer function matrix.

25.5 Poles and Eigenvalues

> **Note:**
> Every pole of G(s) is an eigenvalue of the system matrix A. However, not every eigenvalue
> of A is a pole of G(s).

The poles of the transfer function, and the eigenvalues of the system matrix A are related. In fact, we can say that the eigenvalues of the system matrix A *are the poles of the transfer function* of the system. In this way, if we have the eigenvalues of a system in the state-space domain, we can use the Routh-Hurwitz, and Root Locus methods as if we had our system represented by a transfer function instead.

On a related note, eigenvalues and all methods and mathematical techniques that use eigenvalues to determine system stability *only work with time-invariant systems*. In systems which are time-variant, the methods using eigenvalues to determine system stability fail.

25.6 Transfer Functions Revisited

We are going to have a brief refesher here about transfer functions, because several of the later chapters will use transfer functions for analyzing system stability.

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:

$H_{ol}(s) = KGp(s)Gb(s)$ <---Note this definition now contradicts the updated definition in the "Feedback" section.

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:

[Characteristic Equation]

$$F(s) = 1 + H_{ol} = D(s)$$

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 than 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.

25.7 State-Space and Stability

As we have discussed earlier, the system is stable if the eigenvalues of the system matrix A have negative real parts. However, there are other stability issues that we can analyze, such as whether a system is *uniformly stable*, *asymptotically stable*, or otherwise. We will discuss all these topics in a later chapter.

25.8 Marginal Stability

When the poles of the system in the complex S-Domain exist on the complex frequency axis (the vertical axis), or when the eigenvalues of the system matrix are imaginary (no real part), 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.

We will discuss marginal stability more in the following chapters.

26 Introduction to Digital Controls

26.1 Discrete-Time Stability

The stability analysis of a discrete-time or digital system is similar to the analysis for a continuous time system. However, there are enough differences that it warrants a separate chapter.

26.2 Input-Output Stability

26.2.1 Uniform Stability

An LTI causal system is uniformly BIBO stable if there exists a positive constant L such that the following conditions:

$$x[n_0] = 0$$

$$[n]\| \leq k$$

$$k \geq 0$$

imply that

$$[n]\| \leq L$$

26.2.2 Impulse Response Matrix

We can define the **impulse response matrix** of a discrete-time system as:

[Impulse Response Matrix]

$$G[n] = \begin{cases} CA^{k-1}B & \text{if } k > 0 \\ 0 & \text{if } k \leq 0 \end{cases}$$

Or, in the general time-varying case:

$$G[n] = \begin{cases} C\phi[n,n_0]B & \text{if } k > 0 \\ 0 & \text{if } k \leq 0 \end{cases}$$

A digital system is BIBO stable if and only if there exists a positive constant L such that for all non-negative k:

$$\sum_{n=0}^{k} [n]\| \leq L$$

26.3 Stability of Transfer Function

A MIMO discrete-time system is BIBO stable if and only if every pole of every transfer function in the transfer function matrix has a magnitude less than 1. All poles of all transfer functions must exist inside the unit circle on the Z plane.

26.4 Lyapunov Stability

There is a discrete version of the Lyapunov stability theorem that applies to digital systems. Given the **discrete Lyapunov equation**:

[Digital Lypapunov Equation]

$$A^T M A - M = -N$$

We can use this version of the Lyapunov equation to define a condition for stability in discrete-time systems:

Lyapunov Stability Theorem (Digital Systems)

A digital system with the system matrix A is asymptotically stable if and only if there exists a unique matrix M that satisfies the Lyapunov Equation for every positive definite matrix N.

26.5 Poles and Eigenvalues

Every pole of G(z) is an eigenvalue of the system matrix A. Not every eigenvalue of A is a pole of G(z). Like the poles of the transfer function, all the eigenvalues of the system matrix must have magnitudes less than 1. Mathematically:

$$\sqrt{\text{Re}(z)^2 + \text{Im}(z)^2} \leq 1$$

If the magnitude of the eigenvalues of the system matrix A, or the poles of the transfer functions are greater than 1, the system is unstable.

26.6 Finite Wordlengths

Digital computer systems have an inherent problem because implementable computer systems have finite wordlengths to deal with. Some of the issues are:

1. Real numbers can only be represented with a finite precision. Typically, a computer system can only accurately represent a number to a finite number of decimal points.
2. Because of the fact above, computer systems with feedback can compound errors with each program iteration. Small errors in one step of an algorithm can lead to large errors later in the program.
3. Integer numbers in computer systems have finite lengths. Because of this, integer numbers will either **roll-over**, or **saturate**, depending on the design of the computer system. Both situations can create inaccurate results.

27 Routh-Hurwitz Criterion

27.1 Stability Criteria

The Routh-Hurwitz stability criterion provides a simple algorithm to decide whether or not the zeros of a polynomial are all in the left half of the complex plane (such a polynomial is called at times "Hurwitz"). A Hurwitz polynomial is a key requirement for a linear continuous-time invariant to be stable (all bounded inputs produce bounded outputs).

Necessary stability conditions

  Conditions that must hold for a polynomial to be Hurwitz.

If any of them fails - the polynomial is not stable. However, they may all hold without implying stability.

Sufficient stability conditions

  Conditions that if met imply that the polynomial is stable. However, a polynomial may be stable without implying some or any of them.

The Routh criteria provides condition that are both necessary and sufficient for a polynomial to be Hurwitz.

27.2 Routh-Hurwitz Criteria

The Routh-Hurwitz criteria is comprised of three separate tests that must be satisfied. If any single test fails, the system is not stable and further tests need not be performed. For this reason, the tests are arranged in order from the easiest to determine to the hardest.

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).

27.2.1 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 (equivalently all of them must be negative, with no sign change)

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 every 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.

27.2.2 The Routh's 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:

$$\begin{matrix} s^N \\ s^{N-1} \end{matrix} \begin{vmatrix} a_N & a_{N-2} & \cdots & 0 \\ a_{N-1} & a_{N-3} & \cdots & 0 \end{vmatrix}$$

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:

$$\begin{matrix} s^N \\ s^{N-1} \\ \\ \\ s^0 \end{matrix} \begin{vmatrix} a_N & a_{N-2} & \cdots & 0 \\ a_{N-1} & a_{N-3} & \cdots & 0 \\ b_{N-1} & b_{N-3} & \cdots & \\ c_{N-1} & c_{N-3} & \cdots & \\ \cdots & & & \end{vmatrix}$$

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 negate the determinant of the following matrix, and divide by the pivot element:

$$\begin{vmatrix} k & m \\ l & n \end{vmatrix}$$

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 right of the current element.
- **n** is the element one row up, and one column to the right of the current element.

In terms of **k l m n**, our equation is:

$$v = \frac{(lm) - (kn)}{l}$$

27.2.3 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}:

$$\frac{-1}{b_{N-1}} \begin{vmatrix} a_{N-1} & a_{N-5} \\ b_{N-1} & b_{N-5} \end{vmatrix} = \frac{a_{N-1}b_{N-5} - b_{N-1}a_{N-5}}{-b_{N-1}}$$

27.2.4 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:

$$
\begin{array}{c|ccc}
s^3 & 1 & 4 & 0 \\
s^2 & 2 & 3 & 0 \\
s^1 & b_{N-1} & b_{N-3} & 0 \\
s^0 & c_{N-1} & c_{N-3} & 0
\end{array}
$$

And we can calculate out all the coefficients:

$b_{N-1} = \frac{(2)(4)-(1)(3)}{2} = \frac{5}{2}$

$b_{N-3} = \frac{(2)(0)-(0)(1)}{2} = 0$

$c_{N-1} = \frac{(3)\left(\frac{5}{2}\right)-(2)(0)}{\frac{5}{2}} = 3$

$c_{N-3} = \frac{(2)(0)-\left(\frac{5}{2}\right)(0)}{\frac{5}{2}} = 0$

And filling these values into our Routh Array, we can determine whether the system is stable:

$$
\begin{array}{c|ccc}
s^3 & 1 & 4 & 0 \\
s^2 & 2 & 3 & 0 \\
s^1 & \frac{5}{2} & 0 & 0 \\
s^0 & 3 & 0 & 0
\end{array}
$$

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.

27.2.5 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 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.

The auxiliary equation is to be differentiated with respect to s and the coefficients of this equation replaces the all zero row. Routh array can be further calculated using these new values.

27.2.6 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. If the sign coefficient above the (ε) is the same as below it, this indicates a pure imaginary root.

28 Jury's Test

28.1 Routh-Hurwitz 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.

28.1.1 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.

28.1.2 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**[1].

---

1   `https://en.wikibooks.org/wiki/Control%20Systems%2FZ%20Transform%20Mappings`

28.2 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.

28.2.1 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 than 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:

$$|b_0| > |b_{N-1}|$$

$$|c_0| > |c_{N-2}|$$

$$|d_0| > |d_{N-3}|$$

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.

28.2.2 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	$\ldots$	z^N
a_0	a_1	a_2	a_3	$\ldots$	a_N
a_N	$\ldots$	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:

1)	a_0	a_1	a_2	a_3	$\ldots$	a_N
2)	a_N	$\ldots$	a_3	a_2	a_1	a_0
3)	b_0	b_1	b_2	$\ldots$	b_{N-1}
4)	b_{N-1}	$\ldots$	b_2	b_1	b_0
$\vdots$	$\vdots$	$\vdots$	$\vdots$		
$2N-3)$	v_0	v_1	v_2		

Note: The last file is the (2N-3) file, and always has 3 elements. This test doesn't have sense if N=1, but in this case you know the pole!

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.

28.2.3 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 $\rightarrow$ 1, and row d had R $\rightarrow$ 2, we can follow the pattern and for row e set R $\rightarrow$ 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:

$$e_5 = d_0 d_5 - d_{N-8} d_{N-3}$$

28.3 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[2]

---

2   https://en.wikibooks.org/wiki/Control%20Systems%2FZ%20Transform%20Mappings

29 Root Locus

29.1 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 like 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

29.2 Root-Locus

As we change gain, we notice that the system poles and zeros actually move around in the S-plane[1]. 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. As we know that a fan switch also can control the speed of the fan.

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 poles of the equation, we must set the denominator to 0, and solve the characteristic equation. In other words, 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   https://en.wikibooks.org/wiki/S-plane

$$1 + KG(s)H(s) = 0$$

$$KG(s)H(s) = -1$$

And finally by converting to polar coordinates:

$$\angle KG(s)H(s) = 180°$$

Now we have 2 equations that govern the locations of the poles of a system for all gain values:

[The Magnitude Equation]

$$1 + KG(s)H(s) = 0$$

[The Angle Equation]

$$\angle KG(s)H(s) = 180°$$

29.2.1 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:

$$\angle K\overline{GH}(z) = 180°$$

We are now left with two important equations:

[The Magnitude Equation]

$$1 + K\overline{GH}(z) = 0$$

[The Angle Equation]

$$\angle K\overline{GH}(z) = 180°$$

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.

29.3 The Root-Locus Procedure

> **Note:**
> In this section, the rules for the S-Plane and the Z-plane are the same, so we won't refer to the differences between them.

In the transform domain (see note at right), when the gain is small, 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.

If there are fewer zeros than 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:

$$KG(s)H(s) + 1 = 0 \rightarrow G(s)H(s) = \frac{-1}{K}$$

> **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.

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:

$$\frac{a(s)}{b(s)} = \frac{-1}{K}$$

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 right-hand side of the graph and traveling to the left, we draw a root-locus line on the real-axis at every point to the left of an odd number of poles or zeros 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 implicit zero), or they can travel to meet an explicit 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 implicit zeros at infinity.

29.4 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 ith pole and the ith zero, respectively. Likewise, we will use ψ_i and ρ_i to denote the angle from a given point to the ith 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 a(s) on the graph with an O. There should be a number of O's less than 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 and 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 $\to \sigma$ (no imaginary part), we can use the following equation:

$$K = -\frac{b(\sigma)}{a(\sigma)}$$

And differentiate to find the local maximum:

$$\frac{dK}{d\sigma} = \frac{d}{d\sigma} \frac{b(\sigma)}{a(\sigma)}$$

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:

$$\phi_\omega = \frac{\pi + 2N\pi}{p - z}, \quad N = 0, 1, \dots p - z - 1$$

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 (i.e., 180º).

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.

29.5 Root Locus Equations

Here are the two major equations:

[Root Locus Equations]

S-Domain Equations	Z-Domain Equations
$1 + KG(s)H(s) = 0$	$1 + K\overline{GH}(z) = 0$
$\angle KG(s)H(s) = 180^o$	$\angle K\overline{GH}(z) = 180^o$

Note that the sum of the angles of all the poles and zeros must equal to 180.

29.5.1 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:

[Number of Asymptotes]

$$N_a = P - Z$$

The angles of the asymptotes are given by:

[Angle of Asymptotes]

$$\phi_k = (2k+1)\frac{\pi}{P-Z}$$

for values of $k = [0, 1, ...N_a - 1]$.

The angles for the asymptotes are measured from the positive real-axis

---

29.5.2 Asymptote Intersection Point

The asymptotes intersect the real axis at the point:

[Origin of Asymptotes]

$$\sigma_0 = \frac{\sum_P - \sum_Z}{P - Z}$$

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.

29.5.3 Breakaway Points

The breakaway points are located at the roots of the following equation:

[Breakaway Point Locations]

$\frac{dG(s)H(s)}{ds} = 0$ or $\frac{d\overline{GH}(z)}{dz} = 0$

Once you solve for z, the real roots give you the breakaway/reentry points. Complex roots correspond to a lack of breakaway/reentry.

The breakaway point equation can be difficult to solve, so many times the actual location is approximated.

29.6 Root Locus and Stability

The root locus procedure 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	$|z| < 1$
Marginally Stable Region	The vertical axis	$\sigma = 0$	The Unit Circle	$|z| = 1$
Unstable Region	Right-Hand S Plane	$\sigma > 0$	Outside the Unit Circle,	$|z| > 1$

29.7 Examples

29.7.1 Example 1: First-Order System

Find the root-locus of the open-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:
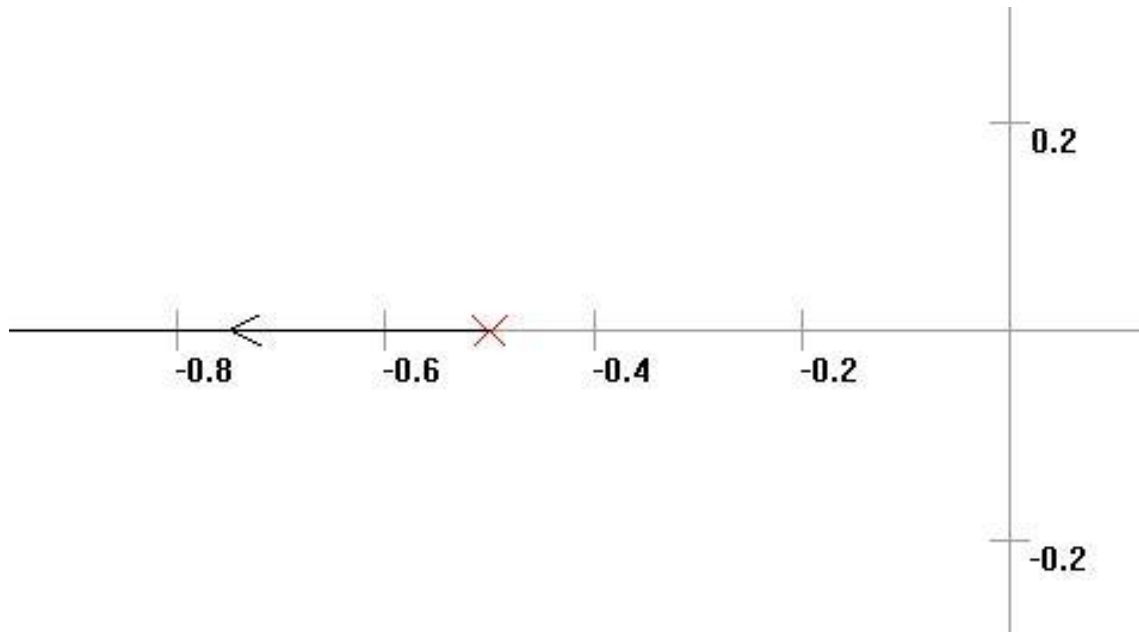
Figure 112

From this image, we can see that for *all values of gain* this system is stable.

29.7.2 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:

$\phi = \frac{180(2k+1)}{3} \text{for} k = 0, 1, 2$

We know that the breakaway occurs between the first and second poles, so we will estimate the exact breakaway point. Drawing the root-locus gives us the graph below.
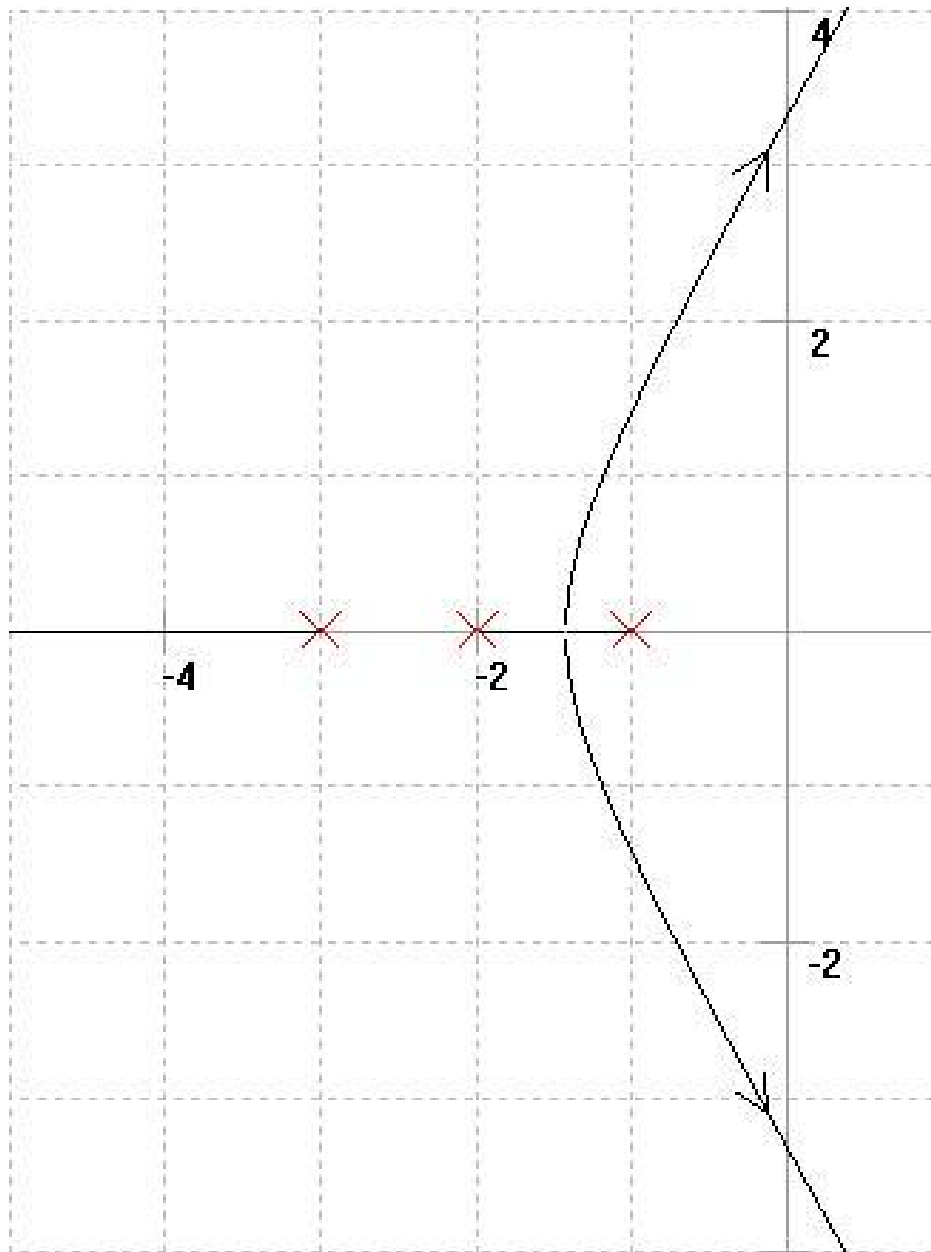
Figure 113

We can see that for low values of gain the system is stable, but for higher values of gain, the system becomes unstable.

29.7.3 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:
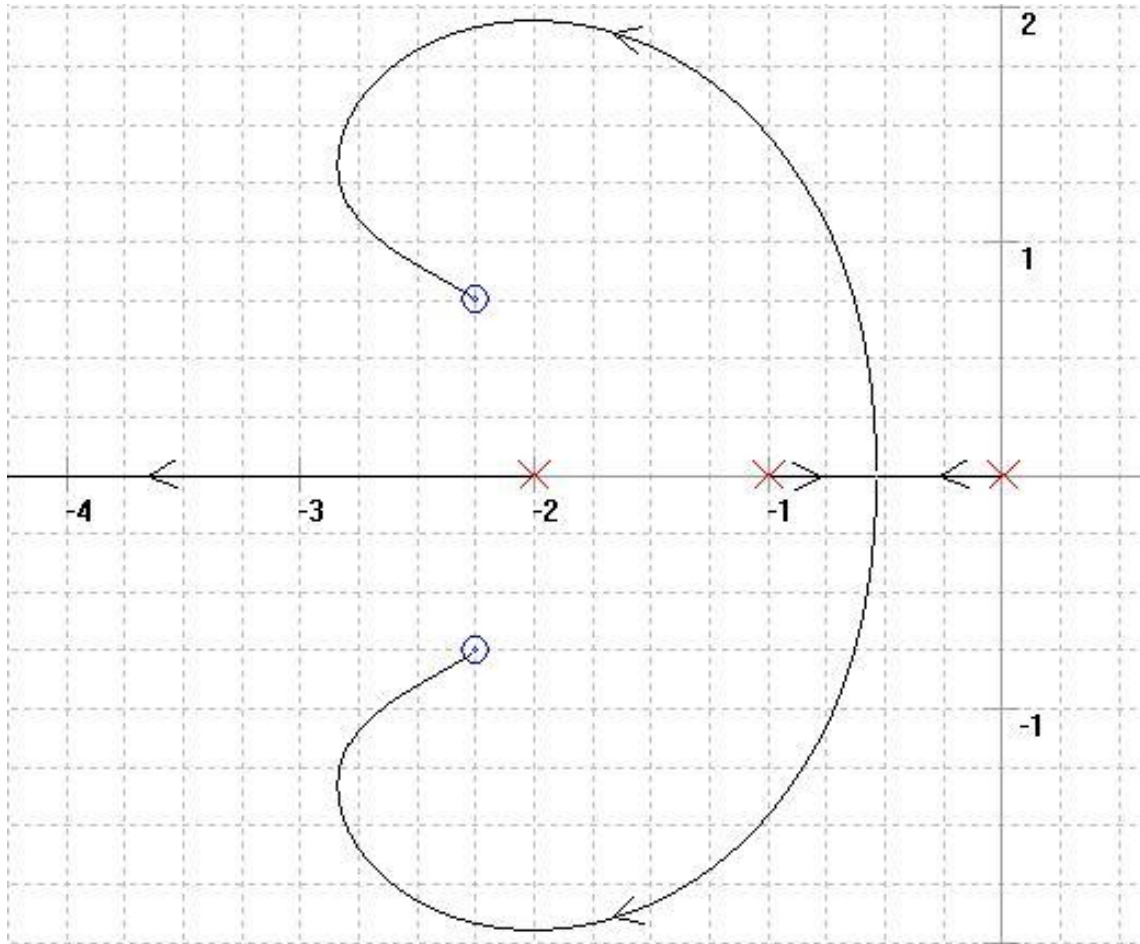
Figure 114

We can see from this graph that the system is stable for all values of K.

29.7.4 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:

$$T(s) = K \frac{s^2 + 7s + 12}{(s^2 + 3s + 6)}$$

First, we must multiply through in the denominator:

$N(s) = S^2 + 7S + 12$

$D(s) = S^2 + 3S + 2$

Now, we can generate the coefficient vectors from the numerator and denominator:

```
num = [0 1 7 12];
den = [0 1 3 2];
```

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:
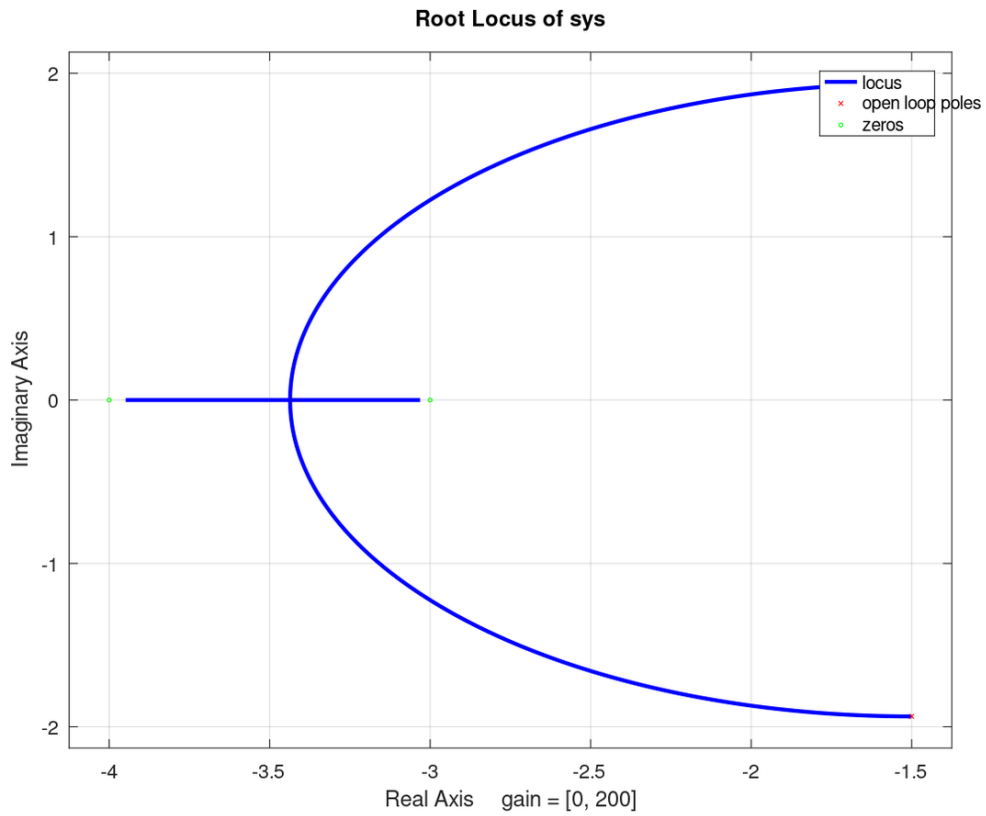
Figure 115

30 Nyquist Criterion

dhunparserurl Control Systems/Nyquist Stability Criteria

30.1 Nyquist Stability Criteria

The **Nyquist Stability Criteria** is a test for system stability, just like the Routh-Hurwitz[1] test, or the Root-Locus[2] 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 microwave radiation (from inside your microwave oven, for instance), the water will 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.

30.2 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**.

---

1   Chapter 27 on page 221
2   Chapter 29 on page 233

30.3 Argument Principle

w:Argument principle[3]

Here is the argument principle, which 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.

30.3.1 Example: First Order System

> Let's say, for instance, that Γ is a unit square contour in the complex s plane. The vertices of the square are located at points I,J,K,L, as follows:
>
> $I = 1 + j$
> $J = 1 - j$
> $K = -1 - j$
> $L = -1 + j$
>
> 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 transform function, F(s), to be the following:
>
> $F(s) = 2s + 1$
>
> We can factor the denominator of F(s), and we can show that there is one zero at s $\to$ -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 imaginary parts as:
>
> $s = \sigma + j\omega$
>
> We know that F(s) must also be complex. We will say, for reasons of simplicity, that the axes in the F(s) plane are u and v, and are related as such:
>
> $F(s) = u + vj = 2(\sigma + j\omega) + 1$
>
> 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 vertices, because they are the simplest examples:
>
> $u + vj = F(I) = 3 + 2j$

---

3   https://en.wikipedia.org/wiki/Argument%20principle

$u + vj = F(J) = 3 - 2j$

$u + vj = F(K) = -1 + 2j$

$u + vj = F(L) = -1 - 2j$

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 vertex to vertex, 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.

30.3.2 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 + 0.5j and s → -0.5 - 0.5j. We will use the same unit square contour, Γ, from above:

$I = 1 + j$

$J = 1 - j$

$K = -1 - j$

$L = -1 + j$

We can see clearly that the poles and the zero of F(s) lie within Γ. Setting F(s) to u + vj and solving, we get the following relationships:

$u + vj = F(\sigma + j\omega) = \frac{(\sigma + 0.5) + j(\omega)}{(2\sigma^2 - 2\omega^2 + 2\sigma + 1) + j(2\sigma\omega + \omega)}$

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:

a) 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.

b) We plug each segment of our contour into this equation, and simplify numerically.

30.4 The Nyquist Contour

The Nyquist contour, the contour that makes the entire nyquist criterion work, must encircle the entire unstable region of the complex plane. For analog systems, this is the right half of the complex s plane. For digital systems, this is the entire plane outside the unit circle. Remember that if a pole to the closed-loop transfer function (or equivalently a zero of the characteristic equation) lies in the unstable region of the complex plane, the system is an unstable system.

Analog Systems

The Nyquist contour for analog systems is an infinite semi-circle that encircles the entire right-half of the s plane. The semicircle travels up the imaginary axis from negative infinity to positive infinity. From positive infinity, the contour breaks away from the imaginary axis, in the clock-wise direction, and forms a giant semicircle.

Digital Systems

The Nyquist contour in digital systems is a counter-clockwise encirclement of the unit circle.

30.5 Nyquist Criteria

w:Nyquist stability criterion[4]

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 in the of the open-loop 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.

w:Nyquist plot[5]
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.

---

4   https://en.wikipedia.org/wiki/Nyquist%20stability%20criterion
5   https://en.wikipedia.org/wiki/Nyquist%20plot

30.6 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.

30.7 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 = Z - P 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.

31 State-Space Stability

dhunparserurl Control Systems/State-Space Stability

31.1 State-Space Stability

If a system is represented in the state-space domain, it doesn't make sense to convert that system to a transfer function representation (or even a transfer matrix representation) in an attempt to use any of the previous stability methods. Luckily, there are other analysis methods that can be used with the state-space representation to determine if a system is stable or not. First, let us first introduce the notion of unstability:

> **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:
>
> $$\lim_{t \to \infty} (t)\| = \infty$$

Also, a key concept when we are talking about stability of systems is the concept of an **equilibrium point**:

> **Equilibrium Point**
> Given a system f such that:
> $x'(t) = f(x(t))$
> A particular state x_e is called an **equilibrium point** if
> $f(x_e) = 0$ for all time t in the interval $[t_0, \infty)$, where t_0 is the starting time of the system.

> An equilibrium point is also known as a "stationary point", a "critical point", a "singular point", or a "rest state" in other books or literature.

The definitions below typically require that the equilibrium point be zero. If we have an equilibrium point $x_e = $ a, then we can use the following change of variables to make the equilibrium point zero:

$\bar{x} = x_e - a = 0$

We will also see below that a system's stability is defined in terms of an equilibrium point. Related to the concept of an equilibrium point is the notion of a **zero point**:

> **Zero State**

A state x_z is a **zero state** if $x_z = 0$. A zero state may or may not be an equilibrium point.

31.1.1 Stability Definitions

The equilibrium $x = 0$ of the system is stable if and only if the solutions of the zero-input state equation are bounded. Equivalently, $x = 0$ is a stable equilibrium if and only if for every initial time t_0, there exists an associated finite constant $k(t_0)$ such that:

$\sup_{t \geq t_0} \|\phi(t, t_0)\| = k(t_0) < \infty$

Where *sup* is the **supremum**, or "maximum" value of the equation. The maximum value of this equation must never exceed the arbitrary finite constant k (and therefore it may not be infinite at any point).

Uniform Stability

The system is defined to be **uniformly stable** if it is stable for all initial values of t_0:

$$\sup_{t \geq 0}[\sup_{t \geq t_0} \|\phi(t, t_0)\|] = k_0 < \infty$$

Uniform stability is a more general, and more powerful form of stability than was previously provided.

Asymptotic Stability

A system is defined to be **asymptotically stable** if:

$$\lim_{t \to \infty} \|\phi(t, t_0)\| = 0$$

A time-invariant system is asymptotically stable if all the eigenvalues of the system matrix A have negative real parts. If a system is asymptotically stable, it is also BIBO stable. However the inverse is not true: A system that is BIBO stable might not be asymptotically stable.

Uniform Asymptotic Stability

A system is defined to be **uniformly asymptotically stable** if the system is asymptotically stable for all values of t_0.

Exponential Stability

A system is defined to be **exponentially stable** if the system response decays exponentially towards zero as time approaches infinity.

For linear systems, uniform asymptotic stability is the same as **exponential stability**. This is not the case with non-linear systems.

31.1.2 Marginal Stability

Here we will discuss some rules concerning systems that are marginally stable. Because we are discussing eigenvalues and eigenvectors, these theorems only apply to time-invariant systems.

1. A time-invariant system is marginally stable if and only if all the eigenvalues of the system matrix A are zero or have negative real parts, and those with zero real parts are simple roots of the minimal polynomial of A.
2. The equilibrium $x = 0$ of the state equation is *uniformly stable* if all eigenvalues of A have non-positive real parts, and there is a complete set of distinct eigenvectors associated with the eigenvalues with zero real parts.
3. The equilibrium $x = 0$ of the state equation is *exponentially stable* if and only if all eigenvalues of the system matrix A have negative real parts.

31.2 Eigenvalues and Poles

A Linearly Time Invariant (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:

$$A^{-1} = \frac{\text{adj}(A)}{|A|}$$

We can use that definition here:

$$X(s) = \frac{\text{adj}(sI - A)BU(s)}{|(sI - A)|}$$

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 (the 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.

31.3 Impulse Response Matrix

We can define the **Impulse response matrix**, $G(t, \tau)$ in order to define further tests for stability:

[Impulse Response Matrix]

$$G(t,\tau) = \begin{cases} C(t)\phi(t,\tau)B(\tau) & \text{if } t \geq \tau \\ 0 & \text{if } t < \tau \end{cases}$$

The system is *uniformly stable* if and only if there exists a finite positive constant L such that for all time t and all initial conditions t_0 with $t \geq t_0$ the following integral is satisfied:

$\int_0^t (t,\tau)\tau \leq L$

In other words, the above integral must have a finite value, or the system is not uniformly stable.

In the time-invariant case, the impulse response matrix reduces to:

$$G(t) = \begin{cases} Ce^{At}B & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}$$

In a time-invariant system, we can use the impulse response matrix to determine if the system is uniformly BIBO stable by taking a similar integral:

$\int_0^\infty (t) \leq L$

Where L is a finite constant.

31.4 Positive Definiteness

These terms are important, and will be used in further discussions on this topic.

- f(x) is **positive definite** if f(x) > 0 for all x.
- f(x) is **positive semi-definite** if $f(x) \geq 0$ for all x, and f(x) = 0 only if x = 0.
- f(x) is **negative definite** if f(x) < 0 for all x.
- f(x) is **negative semi-definite** if $f(x) \leq 0$ for all x, and f(x) = 0 only if x = 0.
  A Hermitian matrix X is positive definite if all its principle minors are positive. Also, a matrix X is positive definite if all its 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.

31.5 Lyapunov Stability

31.5.1 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**.

[Lyapunov Equation]

$$MA + A^T M = -N$$

Where A is the system matrix, and M and N are $p \times p$ square matrices.

> **Lyapunov Stability Theorem**
>
> An LTI system $x' = Ax$ is stable if there exists a matrix M that satisfies the **Lyapunov Equation** where N is an arbitrary positive definite matrix, and M is a unique positive definite matrix.

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. Alternatively, we can write:

> **Lyapunov Stability Theorem (alternate)**
>
> If all the eigenvalues of the system matrix A have negative real parts, then the Lyapunov Equation has a unique solution M for every positive definite matrix N, and the solution can be calculated by:
>
> $$M = \int_0^\infty e^{A^T t} N e^{At} dt$$

If the matrix M can be calculated in this manner, the system is asymptotically stable.

32 Controllability and Observability

32.1 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, represents the internal workings of the system that can be separate from the regular input-output relationship of the system. This also needs to be measured, or *observed*. The term **observability** describes whether the internal state variables of the system can be externally measured.

32.2 Controllability

Complete state controllability (or simply controllability if no other context is given) describes the ability of an external input to move the internal state of a system from any initial state to any other final state in a finite time interval

We will start off with the definitions of the term **controllability**, and the related terms **reachability** and **stabilizability**.

> **Controllability**
>
> A system with internal state vector x is called **controllable** if and only if the system states can be changed by changing the system input.
>
> **Reachability**
>
> A particular state x_1 is called *reachable* if there exists an input that transfers the state of the system from the initial state x_0 to x_1 in some finite time interval $[t_0, t)$.
>
> **Stabilizability**
>
> A system is Stabilizable if all states that cannot be reached decay to zero asymptotically.

We can also write out the definition of reachability more precisely:

A state x_1 is called **reachable** at time t_1 if for some finite initial time t_0 there exists an input u(t) that transfers the state x(t) from the origin at t_0 to x_1.

A system is **reachable** at time t_1 if every state x_1 in the state-space is reachable at time t_1.

Similarly, we can more precisely define the concept of controllability:

A state x_0 is **controllable** at time t_0 if for some finite time t_1 there exists an input u(t) that transfers the state x(t) from x_0 to the origin at time t_1.

A system is called **controllable** at time t_0 if every state x_0 in the state-space is controllable.

32.2.1 Controllability Matrix

For LTI (linear time-invariant) systems, a system is reachable if and only if its **controllability matrix**, ζ, has a full row rank of p, where p is the dimension of the matrix A, and $p \times q$ is the dimension of matrix B.

$$\textbf{[Controllability Matrix]}$$

$$\zeta = \begin{bmatrix} B & AB & A^2B & \cdots & A^{p-1}B \end{bmatrix} \in R^{p \times pq}$$

A system is controllable or "Controllable to the origin" when any state x_1 can be driven to the zero state $x = 0$ in a finite number of steps.

A system is controllable when the rank of the system matrix A is p, and the rank of the controllability matrix is equal to:

$$Rank(\zeta) = Rank(A^{-1}\zeta) = p$$

If the second equation is not satisfied, the system is not .

MATLAB allows one to easily create the controllability matrix with the **ctrb** command. To create the controllability matrix ζ simply type

$$\zeta = ctrb(A, B)$$

where A and B are mentioned above. Then in order to determine if the system is controllable or not one can use the rank command to determine if it has full rank.

If

$$Rank(A) < p$$

Then controllability does not imply reachability.

- Reachability always implies controllability.
- Controllability only implies reachability when the state transition matrix is nonsingular.

32.2.2 Determining Reachability

There are four methods that can be used to determine if a system is reachable or not:

1. If the p rows of $\phi(t,\tau)B(t)$ are linearly independent over the field of complex numbers. That is, if the rank of the product of those two matrices is equal to p for all values of t and τ
2. If the rank of the controllability matrix is the same as the rank of the system matrix A.
3. If the rank of $\text{rank}[\lambda I - A, B] = p$ for all eigenvalues λ of the matrix A.
4. If the rank of the **reachability gramian** (described below) is equal to the rank of the system matrix A.
   Each one of these conditions is both necessary and sufficient. If any one test fails, all the tests will fail, and the system is not reachable. If any test is positive, then all the tests will be positive, and the system is reachable.

32.2.3 Gramians

Gramians are complicated mathematical functions that can be used to determine specific things about a system. For instance, we can use gramians to determine whether a system is controllable or reachable. Gramians, because they are more complicated than other methods, are typically only used when other methods of analyzing a system fail (or are too difficult).

All the gramians presented on this page are all matrices with dimension $p \times p$ (the same size as the system matrix A).

All the gramians presented here will be described using the general case of Linear time-variant systems. To change these into LTI (time-invariant equations), the following substitutions can be used:

$\phi(t,\tau) \to e^{A(t-\tau)}$

$\phi'(t,\tau) \to e^{A'(t-\tau)}$

Where we are using the notation X' to denote the transpose of a matrix X (as opposed to the traditional notation X^T).

32.2.4 Reachability Gramian

We can define the **reachability gramian** as the following integral:

[Reachability Gramian]

$W_r(t_0, t_1) = \int_{t_0}^{t_1} \phi(t_1,\tau)B(\tau)B'(\tau)\phi'(t_1,\tau)d\tau$

The system is reachable if the rank of the reachability gramian is the same as the rank of the system matrix:

$\text{rank}(W_r) = p$ <chemistry>/control{range}

32.2.5 Controllability Gramian

We can define the **controllability gramian** of a system (A, B) as:

[Controllability Gramian]

$$W_c(t_0, t_1) = \int_{t_0}^{t_1} \phi(t_0, \tau) B(\tau) B'(\tau) \phi'(t_0, \tau) d\tau$$

The system is controllable if the rank of the controllability gramian is the same as the rank of the system matrix:

$$\text{rank}(W_c) = p$$

If the system is time-invariant, there are two important points to be made. First, the reachability gramian and the controllability gramian reduce to be the same equation. Therefore, for LTI systems, if we have found one gramian, then we automatically know both gramians. Second, the controllability gramian can also be found as the solution to the following Lyapunov equation:

$$AW_c + W_c A' = -BB'$$

Many software packages, notably MATLAB, have functions to solve the Lyapunov equation. By using this last relation, we can also solve for the controllability gramian using these existing functions.

32.3 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:

> **Observability**
> A system with an initial state, $x(t_0)$ is **observable** if and only if the value of the initial state 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**.
> **Complete Observability**

A system is said to be **completely observable** if all the possible initial states of the system can be observed. Systems that fail this criteria are said to be **unobservable**.

Detectability

A system is **Detectable** if all states that cannot be observed decay to zero asymptotically.

Constructability

A system is **constructable** if the present state of the system can be determined from the present and past outputs and inputs to the system. If a system is observable, then it is also constructable. The relationship does not work the other way around.

A system state x_i is unobservable at a given time t_i if the zero-input response of the system is zero for all time t. If a system is observable, then the only state that produces a zero output for all time is the zero state. We can use this concept to define the term **state-observability**.

State-Observability

A system is completely **state-observable** at time t_0 or the pair (A, C) is observable at t_0 if the only state that is unobservable at t_0 is the zero state x = 0.

32.3.1 Constructability

A state x is **unconstructable** at a time t_1 if for every finite time t < t_1 the zero input response of the system is zero for all time t.

A system is completely **state constructable** at time t_1 if the only state x that is unconstructable at t_0 is $x = 0$.

If a system is observable at an initial time t_0, then it is constructable at some time t > t_0, if it is constructable at t_1.

32.3.2 Observability Matrix

The observability of the system is dependent only on the system states and the system output, so we can simplify our state equations to remove the input terms:

Matrix Dimensions:

A: $p \times p$

B: $p \times q$

C: $r \times p$

D: $r \times q$

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

$y(t) = Cx(t)$

Therefore, we can show that the observability of the system is dependent 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 **observability matrix** Q:

[Observability Matrix]

$$Q = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{p-1} \end{bmatrix}$$

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 $pr \times p$.

MATLAB allows one to easily create the observability matrix with the **obsv** command. To create the observability matrix Q simply type

Q=obsv(A,C)

where A and C are mentioned above. Then in order to determine if the system is observable or not one can use the rank command to determine if it has full rank.

32.3.3 Observability Gramian

We can define an **observability gramian** as:

[Observability Gramian]

$$W_o(t_0, t_1) = \int_{t_0}^{t_1} \phi'(\tau, t_0) C'(\tau) C(\tau) \phi(\tau, t_0) d\tau$$

A system is completely state observable at time $t_0 < t < t_1$ if and only if the rank of the observability gramian is equal to the size p of the system matrix A.

If the system (A, B, C, D) is time-invariant, we can construct the observability gramian as the solution to the Lyapunov equation:

$$A'W_o + W_o A = -C'C$$

32.3.4 Constructability Gramian

We can define a **constructability gramian** as:

[Constructability Gramian]

$$W_{cn}(t_0, t_1) = \int_{t_0}^{t_1} \phi'(\tau, t_1) C'(\tau) C(\tau) \phi(\tau, t_1) d\tau$$

A system is completely state observable at an initial time t_0 if and only if there exists a finite t_1 such that:

$$\text{rank}(W_0) = \text{rank}(W_{cn}) = p$$

Notice that the constructability and observability gramians are very similar, and typically they can both be calculated at the same time, only substituting in different values into the state-transition matrix.

32.4 Duality Principle

The concepts of controllability and observability are very similar. In fact, there is a concrete relationship between the two. We can say that a system (A, B) is controllable if and only if the system (A', C, B', D) is observable. This fact can be proven by plugging A' in for A, and B' in for C into the observability Gramian. The resulting equation will exactly mirror the formula for the controllability gramian, implying that the two results are the same.

33 System Specifications

dhunparserurl Control Systems/System Specifications

33.1 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.

33.2 Steady-State Accuracy

33.3 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:

$S_H^X(s)$

33.4 Disturbance Rejection

All physically-realized systems have to deal with a certain amount of noise and disturbance. The ability of a system to reject the noise is known as the **disturbance rejection** of the system.

33.5 Control Effort

The control effort is the amount of energy or power necessary for the controller to perform its duty.

34 Controllers and Compensators

dhunparserurl Control Systems/Controllers and Compensators

34.1 Controllers

There are a number of different standard types of control systems that have been studied extensively. These controllers, specifically the P, PD, PI, and PID controllers are very common in the production of physical systems, but as we will see they each carry several drawbacks.

34.2 Proportional Controllers
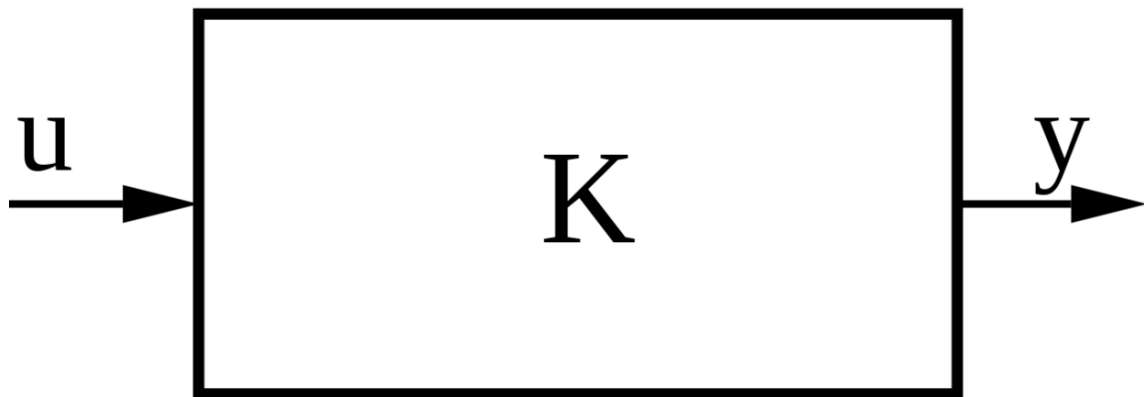
Figure 116   A Proportional controller block diagram

Proportional controllers are simply gain values. These are essentially multiplicative coefficients, usually denoted with a K. A P controller can only force the system poles to a spot on the system's root locus. A P controller cannot be used for arbitrary pole placement.

We refer to this kind of controller by a number of different names: proportional controller, gain, and zeroth-order controller.

34.3 Derivative Controllers
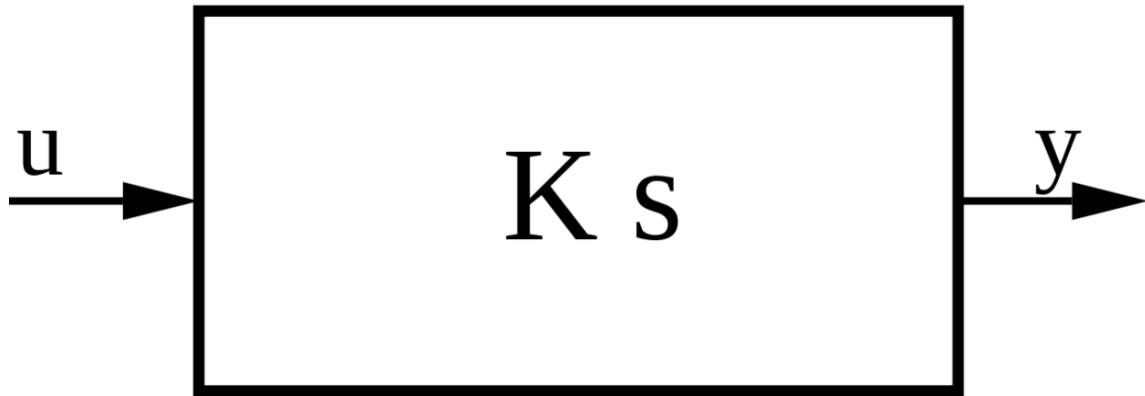
Figure 117 A Proportional-Derivative controller block diagram

In the Laplace domain, we can show the derivative of a signal using the following notation:

$D(s) = \mathcal{L}\{f'(t)\} = sF(s) - f(0)$

Since most systems that we are considering have zero initial condition, this simplifies to:

$D(s) = \mathcal{L}\{f'(t)\} = sF(s)$

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.

Notice that derivative controllers are not proper systems, in that the order of the numerator of the system is greater than the order of the denominator of the system. This quality of being a non-proper system also makes certain mathematical analysis of these systems difficult.

34.3.1 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.

34.4 Integral Controllers
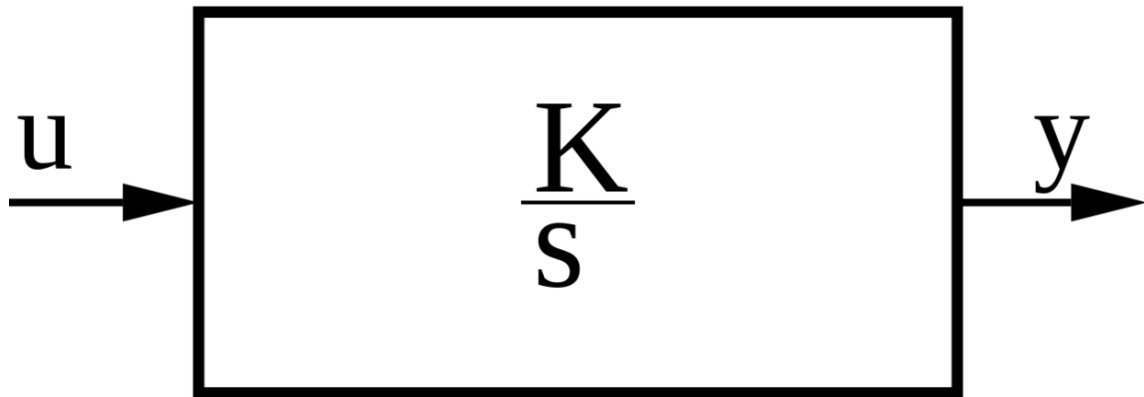
Figure 118   A Proportional-Integral Controller block diagram

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.

34.4.1 Z-Domain Integral

The integral controller can be implemented in the Z domain using the following equation:

$D(z) = \frac{z+1}{z-1}$

34.5 PID Controllers
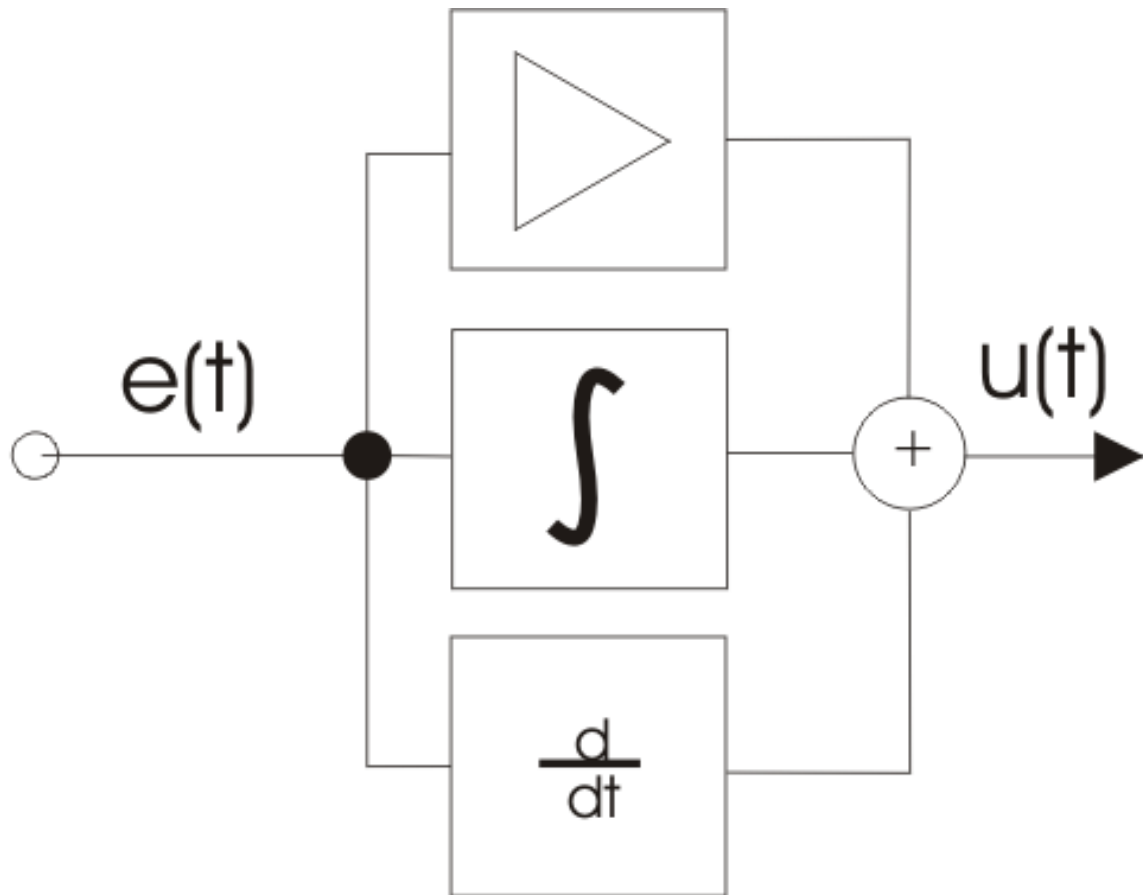
Figure 119   A block diagram of a PID controller

PID controllers are combinations of the proportional, derivative, and integral controllers. Because of this, PID controllers have large amounts of flexibility. We will see below that there are definite limites on PID control.

34.5.1 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:

[PID]

$$D(s) = K_p + \frac{K_i}{s} + K_d s$$

Notice that we can write the transfer function of a PID controller in a slightly different way:

$D(s) = \frac{A_0 + A_1 s}{B_0 + B_1 s}$

This form of the equation will be especially useful to us when we look at polynomial design.

34.5.2 PID Signal flow diagram
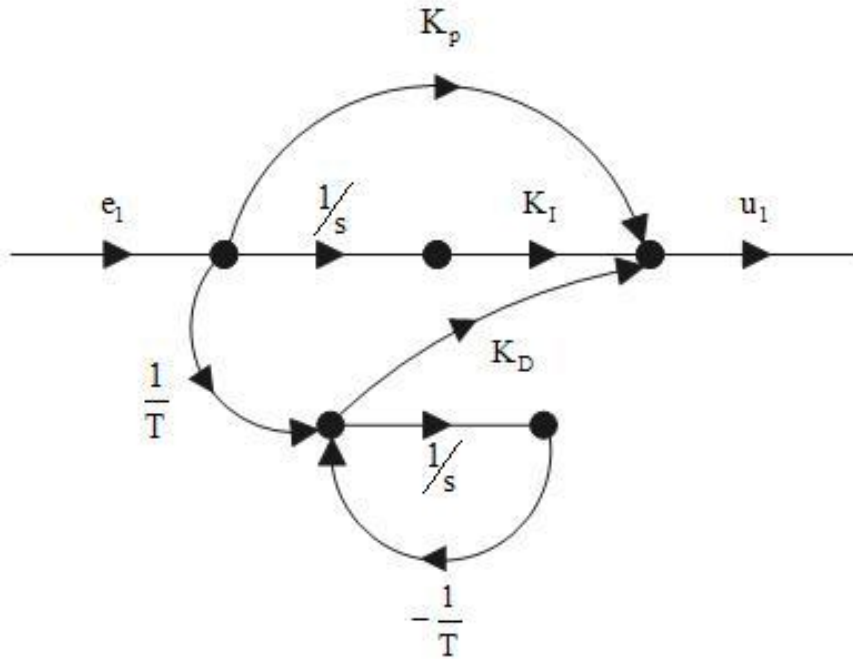
Figure 120   Signal flow diagram for a PID controller

34.5.3 PID Tuning

The process of selecting the various coefficient values to make a PID controller perform correctly is called **PID Tuning**. There are a number of different methods for determining these values:[1]

1) Direct Synthesis (DS) method

2) Internal Model Control (IMC) method

---

1   Seborg, Dale E.; Edgar, Thomas F.; Mellichamp, Duncan A. (2003). Process Dynamics and Control, Second Edition. John Wiley & Sons,Inc.
0471000779

3) Controller tuning relations

4) Frequency response techniques

5) Computer simulation

6) On-line tuning after the control system is installed

7)Trial and error

Notes:

34.5.4 Digital PID

In the Z domain, the PID controller has the following transfer function:

[Digital PID]

$$D(z) = K_p + K_i \frac{T}{2} \left[\frac{z+1}{z-1} \right] + K_d \left[\frac{z-1}{Tz} \right]$$

And we can convert this into a canonical 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[2]

34.6 Bang-Bang Controllers

Despite the low-brow sounding name of the Bang-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. Bang-bang controllers are a non-linear style of control.

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.

34.7 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. There are four major types of compensation 1. Lead compensation 2. Lag compensation 3. Lead-lag compensation 4. Lag-lead compensation

34.8 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 **Lag-lead Compensator** system.(lead-lag system is not practically realisable).

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 doesn't cause the system to become unstable. phase lead compensation:- 1 it is same as addition of zero to open loop TF since from pole zero point of view zero is nearer to origin than pole hence effect of zero dominant.

34.9 Phase Lead

The transfer function for a lead-compensator is as follows:

[Lead Compensator]

$T_{lead}(s) = \frac{s-z}{s-p}$

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 LHP. 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.

34.10 Phase Lag

The transfer function for a lag compensator is the same as the lead-compensator, and is as follows:

[Lag Compensator]

$T_{lag}(s) = \frac{s-z}{s-p}$

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.

34.11 Phase Lag-lead

w:Lag-lead compensator[3]

The transfer function of a **Lag-lead compensator** is simply a multiplication of the lead and lag compensator transfer functions, and is given as:

[Lag-lead Compensator]

$T_{Lag-lead}(s) = \frac{(s-z_1)(s-z_2)}{(s-p_1)(s-p_2)}$.

Where typically the following relationship must hold true:

$|p_1| > |z_1| > |z_2| > |p_2|$

34.12 External links

- Standard Controller Forms on ControlTheoryPro.com[4]
- PID Control on ControlTheoryPro.com[5]
- PI Control on ControlTheoryPro.com[6]

---

3   https://en.wikipedia.org/wiki/Lag-lead%20compensator
4   http://wikis.controltheorypro.com/index.php?title=Standard_Controller_Forms
5   http://wikis.controltheorypro.com/index.php?title=PID_Control
6   http://wikis.controltheorypro.com/index.php?title=PI_Control

35 Nonlinear Systems

dhunparserurl Control Systems/Nonlinear Systems

35.1 Nonlinear General Solution

w:Non-linear control[1] A nonlinear system, in general, can be defined as follows:

$x'(t) = f(t, t_0, x, x_0)$

$x(t_0) = x_0$

Where f is a nonlinear function of the time, the system state, and the initial conditions. If the initial conditions are known, we can simplify this as:

$x'(t) = f(t, x)$

The general solution of this equation (or the most general form of a solution that we can state without knowing the form of f) is given by:

$x(t) = x_0 + \int_{t_0}^{t} f(\tau, x) d\tau$

and we can prove that this is the general solution to the above equation because when we differentiate both sides we get the origin equation.

35.1.1 Iteration Method

The general solution to a nonlinear system can be found through a method of infinite iteration. We will define x_n as being an iterative family of indexed variables. We can define them recursively as such:

$x_n(t) = x_0 + \int_{t_0}^{t} f(\tau, x_{n-1}(\tau)) d\tau$

$x_1(t) = x_0$

We can show that the following relationship is true:

$x(t) = \lim_{n \to \infty} x_n(t)$

The x_n series of equations will converge on the solution to the equation as n approaches infinity.

35.1.2 Types of Nonlinearities

Nonlinearities can be of two types:

---

1   https://en.wikipedia.org/wiki/Non-linear%20control

1. **Intentional non-linearity**: The non-linear elements that are added into a system. Eg: Relay
2. **Incidental non-linearity**: The non-linear behavior that is already present in the system. Eg: Saturation

35.2 Linearization

Nonlinear systems are difficult to analyze, and for that reason one of the best methods for analyzing those systems is to find a linear approximation to the system. Frequently, such approximations are only good for certain operating ranges, and are not valid beyond certain bounds. The process of finding a suitable linear approximation to a nonlinear system is known as **linearization**.
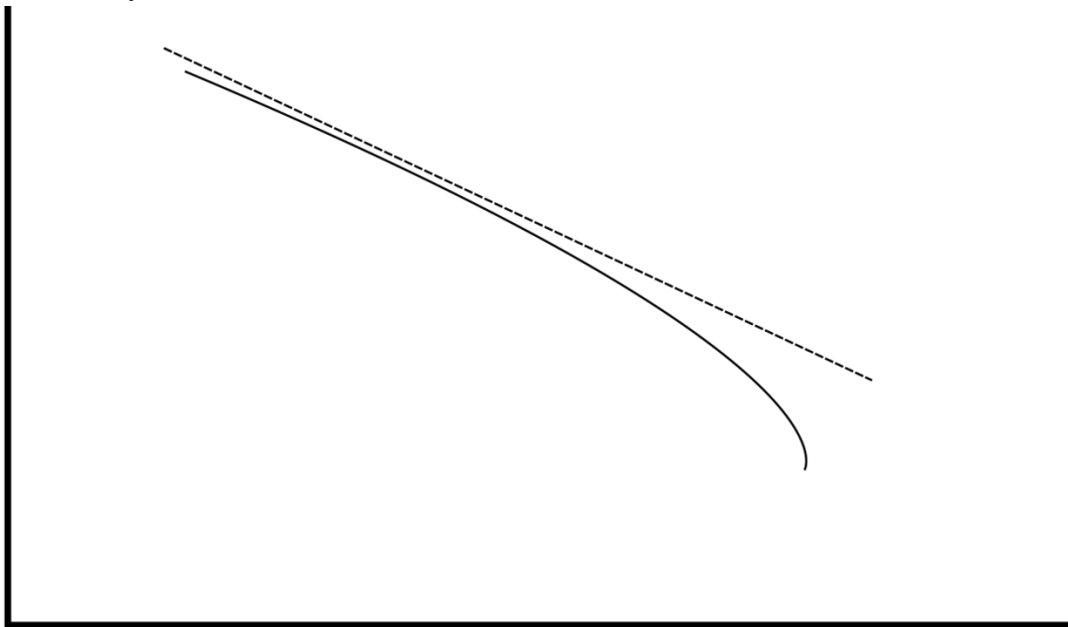
Figure 121

This image shows a linear approximation (dashed line) to a non-linear system response (solid line). This linear approximation, like most, is accurate within a certain range, but becomes more inaccurate outside that range. Notice how the curve and the linear approximation diverge towards the right of the graph.

36 Common Nonlinearities

dhunparserurl Control Systems/Common Nonlinearities

There are some nonlinearities that happen so frequently in physical systems that they are called "Common nonlinearities". These common nonlinearities include Hysteresis, Backlash, and Dead-zone.

36.1 Hysteresis

Continuing with the example of a household thermostat, let's say that your thermostat is set at 70 degrees (Fahrenheit). The furnace turns on, and the house heats up to 70 degrees, and then the thermostat dutifully turns the furnace off again. However, there is still a large amount of residual heat left in the ducts, and the hot air from the vents on the ground may not all have risen up to the level of the thermostat. This means that after the furnace turns off, the house may continue to get hotter, maybe even to uncomfortable levels.

So the furnace turns off, the house heats up to 80 degrees, and then the air conditioner turns on. The temperature of the house cools down to 70 degrees again, and the A/C turns back off. However, the house continues to cool down, and then it gets too cold, and the furnace needs to turn back on.

As we can see from this example, a bang-bang controller, if poorly designed, can cause big problems, and it can waste lots of energy. To avoid this, we implement the idea of **Hysteresis**, which is a set of threshold values that allow for overflow outputs. Implementing hysteresis, our furnace now turns off when we get to 65 degrees, and the house slowly warms up to 75 degrees, and doesn't turn on the A/C unit. This is a far preferable solution.

36.2 Backlash

E.g.: Mechanical gear.

36.3 Dead-Zone

A dead-zone is a kind of non linearity in which the system doesn't respond to the given input until the input reaches a particular level or it can refer to a condition in which output becomes zero when the input crosses certain limiting value.

36.4 Inverse Nonlinearities

36.4.1 Inverse Backlash

36.4.2 Inverse Dead-Zone

37 Noise Driven Systems

The topics in this chapter will rely heavily on topics from a calculus-based background in probability theory. There currently are no wikibooks available that contain this information. The reader should be familiar with the following concepts: Gaussian Random Variables, Mean, Expectation Operation.

37.1 Noise-Driven Systems

Systems frequently have to deal with not only the control input u, but also a random noise input v. In some disciplines, such as in a study of electrical communication systems, the noise and the data signal can be added together into a composite input $r = u + v$. However, in studying control systems, we can not combine these inputs together, for a variety of different reasons:

3. The control input works to stabilize the system, and the noise input works to destabilize the system.
4. The two inputs are independent random variables.
5. The two inputs may act on the system in completely different ways.
   As we will show in the next example, it is frequently a good idea to consider the noise and the control inputs separately:

> **Example:** Consider a moving automobile. The control signals for the automobile consist of acceleration (gas pedal) and deceleration (brake pedal) inputs acting on the wheels of the vehicle, and working to create forward motion. The noise inputs to the system can consist of wind pushing against the vertical faces of the automobile, rough pavement (or even dirt) under the tires, bugs and debris hitting the front windshield, etc. As we can see, the control inputs act on the wheels of the vehicle, while the noise inputs can act on multiple sides of the vehicle, in different ways.

37.2 Probability Refresher

We are going to have a brief refesher here for calculus-based probability, specifically focusing on the topics that we will use in the rest of this chapter.

37.2.1 Expectation

The expectation operatior, **E**, is used to find the *expected, or mean value* of a given random variable. The expectation operator is defined as:

$E[x] = \int_{-\infty}^{\infty} x f_x(x) dx$

If we have two variables that are independent of one another, the expectation of their product is zero.

37.2.2 Covariance

The **covariance** matrix, Q, is the expectation of a random vector times it's transpose:

$E[x(t)x'(t)] = Q(t)$

If we take the value of the x transpose at a different point in time, we can calculate out the covariance as:

$E[x(t)x'(s)] = Q(t)\delta(t - s)$

Where δ is the impulse function.

37.3 Noise-Driven System Description

We can define the state equation to a system incorporating a noise vector v:

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

For generality, we will discuss the case of a time-variant system. Time-invariant system results will then be a simplification of the time-variant case. Also, we will assume that v is a **gaussian random variable**. We do this because physical systems frequently approximate gaussian processes, and because there is a large body of mathematical tools that we can use to work with these processes. We will assume our gaussian process has zero-mean.

37.4 Mean System Response

We would like to find out how our system will respond to the new noisy input. Every system iteration will have a different response that varies with the noise input, but the average of all these iterations should converge to a single value.

For the system with zero control input, we have:

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

For which we know our general solution is given as:

$x(t) = \phi(t, t_0)x_0 + \int_{t_0}^{t} \phi(t, \tau)B(\tau)v(\tau)d\tau$

If we take the **expected value** of this function, it should give us the expected value of the output of the system. In other words, we would like to determine what the expected output of our system is going to be by adding a new, noise input.

$E[x(t)] = E[\phi(t, t_0)x_0] + E[\int_{t_0}^{t} \phi(t, \tau)B(\tau)v(\tau)d\tau]$

In the second term of this equation, neither φ nor B are random variables, and therefore they can come outside of the expectaion operation. Since v is zero-mean, the expectation of it is zero. Therefore, the second term is zero. In the first equation, φ is not a random variable, but x_0 does create a dependancy on the output of $x(t)$, and we need to take the expectation of it. This means that:

$$E[x(t)] = \phi(t, t_0)E[x_0]$$

In other words, the expected output of the system is, on average, the value that the output would be if there were no noise. Notice that if our noise vector v was not zero-mean, and if it was not gaussian, this result would not hold.

37.5 System Covariance

We are now going to analyze the covariance of the system with a noisy input. We multiply our system solution by its transpose, and take the expectation: (this equation is long and might break onto multiple lines)

$$E[x(t)x'(t)] = E[\phi(t, t_0)x_0 + \int_{t_0}^t \phi(\tau, t_0)B(\tau)v(\tau)d\tau]E[(\phi(t, t_0)x_0 + \int_{t_0}^t \phi(\tau, t_0)B(\tau)v(\tau)d\tau)']$$

If we multiply this out term by term, and cancel out the expectations that have a zero-value, we get the following result:

$$E[x(t)x'(t)] = \phi(t, t_0)E[x_0x_0']\phi'(t, t_0) = P$$

We call this result P, and we can find the first derivative of P by using the chain-rule:

$$P'(t) = A(t)\phi(t, t_0)P_0\phi(t, t_0) + \phi(t, t_0)P_0\phi'(t, t_0)A'(t)$$

Where

$$P_0 = E[x_0x_0']$$

We can reduce this to:

$$P'(t) = A(t)P(t) + P(t)A'(t) + B(t)Q(t)B'(t)$$

In other words, we can analyze the system *without needing to calculate the state-transition matrix*. This is a good thing, because it can often be very difficult to calculate the state-transition matrix.

37.6 Alternate Analysis

Let us look again at our general solution:

$$x(t) = \phi(t, t_0)x(t_0) + \int_{t_0}^t \phi(t, \tau)B(\tau)v(\tau)d\tau$$

We can run into a problem because in a gaussian distribution, especially systems with high variance (especially systems with infinite variance), the value of v can momentarily become undefined (approach infinity), which will cause the value of x to likewise become undefined at certain points. This is unacceptable, and makes further analysis of this problem difficult. Let us look again at our original equation, with zero control input:

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

We can multiply both sides by dt, and get the following result:

$dx = A(t)x(t)dt + B(t)v(t)dt$

> This new term, dw, is a random process known as a **Weiner Process**, which the result of transforming a gaussian process in this manner.

We can define a new differential, $dw(t)$, which is an infinitesimal function of time as:

$dw(t) = v(t)dt$

Now, we can integrate both sides of this equation:

$x(t) = x(t_0) + \int_{t_0}^{t} A(\tau)x(\tau)d\tau + \int_{t_0}^{t} B(\tau)dw(\tau)$ w:Ito Calculus[1] However, this leads us to an unusual place, and one for which we are (probably) not prepared to continue further: in the third term on the left-hand side, we are attempting to integrate with respect to a *function*, not a *variable*. In this instance, the standard Riemann integrals that we are all familiar with cannot solve this equation. There are advanced techniques known as **Ito Calculus** however that can solve this equation, but these methods are currently outside the scope of this book.

---

1   https://en.wikipedia.org/wiki/Ito%20Calculus

38 Appendix: Physical Models

dhunparserurl Control Systems/Physical Models

38.1 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 knowledgeable contributors.

38.2 Electrical Systems

For more information about electric circuits and circuit elements, see the following books:
Circuit Theory[a]
Electronics[b]

---

a   https://en.wikibooks.org/wiki/Circuit%20Theory
b   https://en.wikibooks.org/wiki/Electronics

Component	Time-Domain	Laplace	Fourier
Resistor	R	R	R
Capacitor	$i = C\frac{dv}{dt}$	$G(s) = \frac{1}{sC}$	$G(j\omega) = \frac{1}{j\omega C}$
Inductor	$v = L\frac{di}{dt}$	$G(s) = sL$	$G(j\omega) = j\omega L$

38.3 Mechanical Systems

38.4 Civil/Construction Systems

38.5 Chemical Systems

39 Appendix: Z Transform Mappings

dhunparserurl Control Systems/Z Transform Mappings

39.1 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.

39.2 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:

[Bilinear Transform]

$w = \frac{2}{T}\frac{z-1}{z+1}$

[Inverse Bilinear Transform]

$z = \frac{1+(Tw/2)}{1-(Tw/2)}$

Graphically, we can show that the bilinear transform operates as follows:
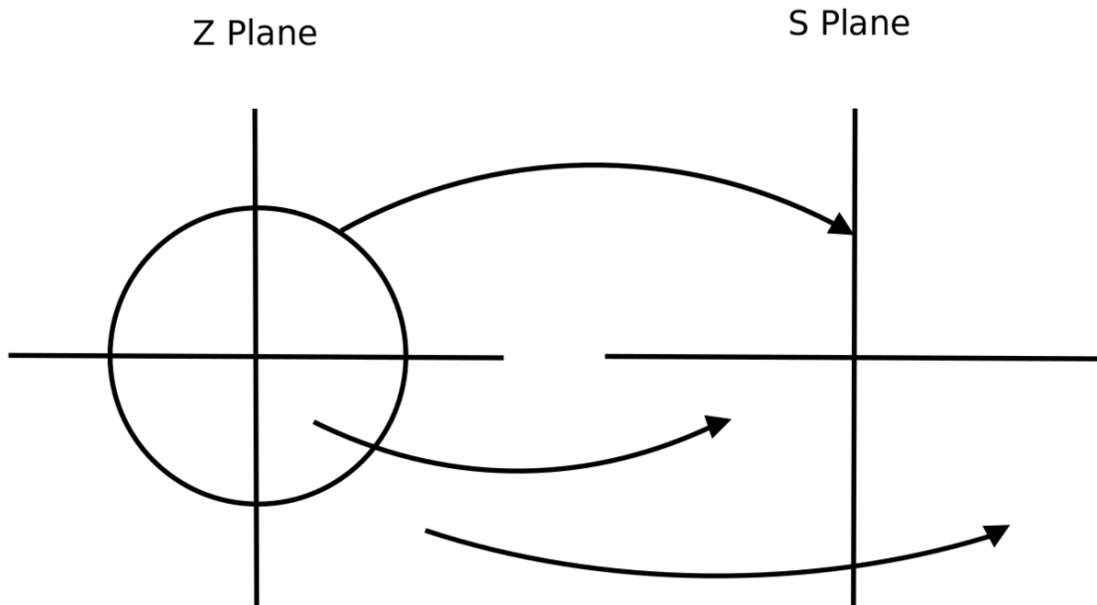
Figure 122

39.2.1 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:
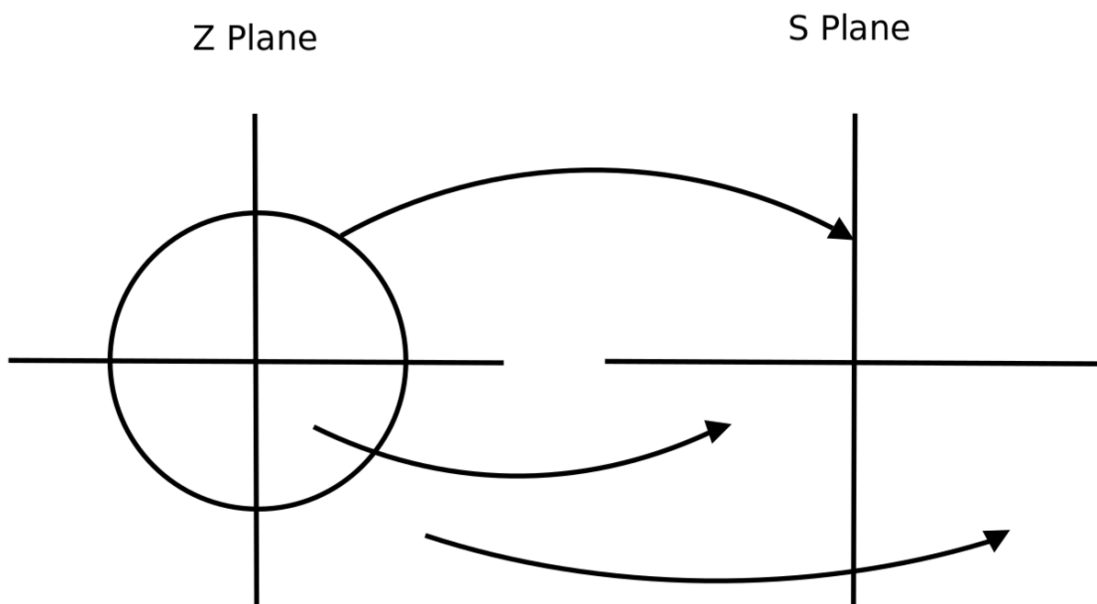
Figure 123

The shape of the graph before and after prewarping is the same as it is without prewarping. However, the destination domain is the S-domain, not the W-domain.

39.3 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:

[Matched Z Transform]

$s + \alpha = 1 - z^{-1}e^{-\alpha T}$

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.

39.4 Simpson's Rule

[Simpson's Rule]

$$s = \frac{3}{T}\frac{z^2 - 1}{z^2 + 4z^1 + 1}$$

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. It has been shown that this transform produces unstable roots (outside of unit unit circle).

39.5 (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.

39.6 Z-Forms

40 Appendix: Transforms

dhunparserurl Control Systems/Transforms Appendix

40.1 Laplace Transform

w:Laplace Transform[1]

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 Transform, below) is not used in this book.

The Laplace Transform is defined as:

[Laplace Transform]

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

And the Inverse Laplace Transform is defined as:

[Inverse Laplace Transform]

$$f(t) = \mathcal{L}^{-1}\{F(s)\} = \frac{1}{2\pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} X(s)e^{st}ds$$

40.1.1 Table of Laplace Transforms

This is a table of common laplace transforms.

dhunparserurl Engineering Tables/Laplace Transform Table

	Time Domain	**Laplace Domain**
	$x(t) = \mathcal{L}^{-1}\{X(s)\}$	$X(s) = \mathcal{L}\{x(t)\}$
1	$\frac{1}{2\pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} X(s)e^{st}ds$	$\int_{-\infty}^{\infty} x(t)e^{-st}dt$
2	$\delta(t)$	1
3	$\delta(t-a)$	e^{-as}
4	$u(t)$	$\frac{1}{s}$
5	$u(t-a)$	$\frac{e^{-as}}{s}$
6	$tu(t)$	$\frac{1}{s^2}$
7	$t^n u(t)$	$\frac{n!}{s^{n+1}}$
8	$\frac{1}{\sqrt{\pi t}}u(t)$	$\frac{1}{\sqrt{s}}$

---

1   https://en.wikipedia.org/wiki/Laplace%20Transform

	Time Domain	Laplace Domain
	$x(t) = \mathcal{L}^{-1}\{X(s)\}$	$X(s) = \mathcal{L}\{x(t)\}$
9	$e^{at}u(t)$	$\frac{1}{s-a}$
10	$t^n e^{at}u(t)$	$\frac{n!}{(s-a)^{n+1}}$
11	$\cos(\omega t)u(t)$	$\frac{s}{s^2+\omega^2}$
12	$\sin(\omega t)u(t)$	$\frac{\omega}{s^2+\omega^2}$
13	$\cosh(\omega t)u(t)$	$\frac{s}{s^2-\omega^2}$
14	$\sinh(\omega t)u(t)$	$\frac{\omega}{s^2-\omega^2}$
15	$e^{at}\cos(\omega t)u(t)$	$\frac{s-a}{(s-a)^2+\omega^2}$
16	$e^{at}\sin(\omega t)u(t)$	$\frac{\omega}{(s-a)^2+\omega^2}$
17	$\frac{1}{2\omega^3}(\sin\omega t - \omega t\cos\omega t)$	$\frac{1}{(s^2+\omega^2)^2}$
18	$\frac{t}{2\omega}\sin\omega t$	$\frac{s}{(s^2+\omega^2)^2}$
19	$\frac{1}{2\omega}(\sin\omega t + \omega t\cos\omega t)$	$\frac{s^2}{(s^2+\omega^2)^2}$

40.1.2 Properties of the Laplace Transform

This is a table of the most important properties of the laplace transform.

dhunparserurl Engineering Tables/Laplace Transform Properties

Property	Definition
Linearity	$\mathcal{L}\{af(t)+bg(t)\} = aF(s)+bG(s)$
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	$\mathcal{L}\left\{\int_0^t f(\tau)\,d\tau\right\} = \mathcal{L}\{u(t)*f(t)\} = \frac{1}{s}F(s)$
Scaling	$\mathcal{L}\{f(at)\} = \frac{1}{a}F\left(\frac{s}{a}\right)$
Initial value theorem	$f(0^+) = \lim_{s\to\infty} sF(s)$
Final value theorem	$f(\infty) = \lim_{s\to 0} sF(s)$
Frequency Shifts	$\mathcal{L}\{e^{at}f(t)\} = F(s-a)$ $\mathcal{L}^{-1}\{F(s-a)\} = e^{at}f(t)$
Time Shifts	$\mathcal{L}\{f(t-a)u(t-a)\} = e^{-as}F(s)$ $\mathcal{L}^{-1}\{e^{-as}F(s)\} = f(t-a)u(t-a)$
Convolution Theorem	$\mathcal{L}\{f(t)*g(t)\} = F(s)G(s)$

Where:

$f(t) = \mathcal{L}^{-1}\{F(s)\}$

$g(t) = \mathcal{L}^{-1}\{G(s)\}$

$s = \sigma + j\omega$

40.1.3 Convergence of the Laplace Integral

40.1.4 Properties of the Laplace Transform

40.2 Fourier Transform

w:Fourier Transform[2]

The Fourier Transform is used to break a time-domain signal into its 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:

[Fourier Transform]

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

And the Inverse Fourier Transform is defined as:

[Inverse Fourier Transform]

$$f(t) = \mathcal{F}^{-1}\{F(j\omega)\} = \frac{1}{2\pi}\int_{-\infty}^\infty F(j\omega)e^{-j\omega t}d\omega$$

40.2.1 Table of Fourier Transforms

This is a table of common fourier transforms.
dhunparserurl Engineering Tables/Fourier Transform Table

	Time Domain	Frequency Domain									
	$x(t) = \mathcal{F}^{-1}\{X(\omega)\}$	$X(\omega) = \mathcal{F}\{x(t)\}$									
1	$X(j\omega) = \int_{-\infty}^\infty x(t)e^{-j\omega t}dt$	$x(t) = \frac{1}{2\pi}\int_{-\infty}^\infty X(\omega)e^{j\omega t}d\omega$									
2	1	$2\pi\delta(\omega)$									
3	$-0.5 + u(t)$	$\frac{1}{j\omega}$									
4	$\delta(t)$	1									
5	$\delta(t-c)$	$e^{-j\omega c}$									
6	$u(t)$	$\pi\delta(\omega) + \frac{1}{j\omega}$									
7	$e^{-bt}u(t)\,(b>0)$	$\frac{1}{j\omega+b}$									
8	$\cos\omega_0 t$	$\pi[\delta(\omega+\omega_0)+\delta(\omega-\omega_0)]$									
9	$\cos(\omega_0 t + \theta)$	$\pi\left[e^{-j\theta}\delta(\omega+\omega_0)+e^{j\theta}\delta(\omega-\omega_0)\right]$									
10	$\sin\omega_0 t$	$j\pi[\delta(\omega+\omega_0)-\delta(\omega-\omega_0)]$									
11	$\sin(\omega_0 t + \theta)$	$j\pi\left[e^{-j\theta}\delta(\omega+\omega_0)-e^{j\theta}\delta(\omega-\omega_0)\right]$									
12	$\text{rect}\left(\frac{t}{\tau}\right)$	$\tau\text{sinc}\left(\frac{\tau\omega}{2\pi}\right)$									
13	$\tau\text{sinc}\left(\frac{\tau t}{2\pi}\right)$	$2\pi\text{rect}\left(\frac{\omega}{\tau}\right)$									
14	$\left(1-\frac{2|t|}{\tau}\right)\text{rect}\left(\frac{t}{\tau}\right)$	$\frac{\tau}{2}\text{sinc}^2\left(\frac{\tau\omega}{4\pi}\right)$									

2   https://en.wikipedia.org/wiki/Fourier%20Transform

	Time Domain	Frequency Domain									
	$x(t) = \mathcal{F}^{-1}\{X(\omega)\}$	$X(\omega) = \mathcal{F}\{x(t)\}$									
15	$\frac{\tau}{2}\operatorname{sinc}^2\left(\frac{\tau t}{4\pi}\right)$	$2\pi\left(1 - \frac{2|\omega|}{\tau}\right)\operatorname{rect}\left(\frac{\omega}{\tau}\right)$									
16	$e^{-a|t|}, \Re\{a\} > 0$	$\frac{2a}{a^2+\omega^2}$									

Notes:											
	1. $\operatorname{sinc}(x) = \sin(\pi x)/(\pi x)$ 2. $\operatorname{rect}\left(\frac{t}{\tau}\right)$ is the rectangular pulse function of width τ 3. $u(t)$ is the Heaviside step function 4. $\delta(t)$ is the Dirac delta function										

40.2.2 Table of Fourier Transform Properties

This is a table of common properties of the fourier transform.

dhunparserurl Engineering Tables/Fourier Transform Properties

	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	$g(at)$	$\frac{1}{|a|}G\left(\frac{\omega}{a}\right)$	$\frac{1}{|a|}G\left(\frac{f}{a}\right)$	If $|a|$ is large, then $g(at)$ is concentrated around 0 and $\frac{1}{|a|}G\left(\frac{\omega}{a}\right)$ 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	$t^n g(t)$	$i^n \frac{d^n G(\omega)}{d\omega^n}$	$\left(\frac{i}{2\pi}\right)^n \frac{d^n G(f)}{df^n}$	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	$g(t)h(t)$	$\frac{(G*H)(\omega)}{\sqrt{2\pi}}$	$(G*H)(f)$	This is the dual of 8

	Signal	Fourier transform unitary, angular frequency	Fourier transform unitary, ordinary frequency	Remarks
10	For a purely real even function $g(t)$	$G(\omega)$ is a purely real even function	$G(f)$ is a purely real even function	
11	For a purely real odd function $g(t)$	$G(\omega)$ is a purely imaginary odd function	$G(f)$ is a purely imaginary odd function	

40.2.3 Convergence of the Fourier Integral

40.2.4 Properties of the Fourier Transform

40.3 Z-Transform

w:Z-transform[3]

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:

[Z Transform]

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

40.3.1 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.

[Inverse Z Transform]

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

40.3.2 Z-Transform Tables

dhunparserurl Engineering Tables/Z Transform Table

Here:

---

3   https://en.wikipedia.org/wiki/Z-transform

- $u[n] = 1$ for $n >= 0$, $u[n] = 0$ for $n < 0$
- $\delta[n] = 1$ for $n = 0$, $\delta[n] = 0$ otherwise

	Signal, $x[n]$	**Z-transform,** $X(z)$	**ROC**
1	$\delta[n]$	1	all z
2	$\delta[n - n_0]$	z^{-n_0}	$z \neq 0$
3	$u[n]$	$\frac{1}{1-z^{-1}}$	$|z| > 1$
4	$-u[-n-1]$	$\frac{1}{1-z^{-1}}$	$|z| < 1$
5	$nu[n]$	$\frac{z^{-1}}{(1-z^{-1})^2}$	$|z| > 1$
6	$-nu[-n-1]$	$\frac{z^{-1}}{(1-z^{-1})^2}$	$|z| < 1$
7	$n^2 u[n]$	$\frac{z^{-1}(1+z^{-1})}{(1-z^{-1})^3}$	$|z| > 1$
8	$-n^2 u[-n-1]$	$\frac{z^{-1}(1+z^{-1})}{(1-z^{-1})^3}$	$|z| < 1$
9	$n^3 u[n]$	$\frac{z^{-1}(1+4z^{-1}+z^{-2})}{(1-z^{-1})^4}$	$|z| > 1$
10	$-n^3 u[-n-1]$	$\frac{z^{-1}(1+4z^{-1}+z^{-2})}{(1-z^{-1})^4}$	$|z| < 1$
11	$a^n u[n]$	$\frac{1}{1-az^{-1}}$	$|z| > |a|$
12	$-a^n u[-n-1]$	$\frac{1}{1-az^{-1}}$	$|z| < |a|$
13	$na^n u[n]$	$\frac{az^{-1}}{(1-az^{-1})^2}$	$|z| > |a|$
14	$-na^n u[-n-1]$	$\frac{az^{-1}}{(1-az^{-1})^2}$	$|z| < |a|$
15	$n^2 a^n u[n]$	$\frac{az^{-1}(1+az^{-1})}{(1-az^{-1})^3}$	$|z| > |a|$
16	$-n^2 a^n u[-n-1]$	$\frac{az^{-1}(1+az^{-1})}{(1-az^{-1})^3}$	$|z| < |a|$
17	$\cos(\omega_0 n)u[n]$	$\frac{1-z^{-1}\cos(\omega_0)}{1-2z^{-1}\cos(\omega_0)+z^{-2}}$	$|z| > 1$
18	$\sin(\omega_0 n)u[n]$	$\frac{z^{-1}\sin(\omega_0)}{1-2z^{-1}\cos(\omega_0)+z^{-2}}$	$|z| > 1$
19	$a^n \cos(\omega_0 n)u[n]$	$\frac{1-az^{-1}\cos(\omega_0)}{1-2az^{-1}\cos(\omega_0)+a^2z^{-2}}$	$|z| > |a|$
20	$a^n \sin(\omega_0 n)u[n]$	$\frac{az^{-1}\sin(\omega_0)}{1-2az^{-1}\cos(\omega_0)+a^2z^{-2}}$	$|z| > |a|$

40.4 Modified Z-Transform

w:Advanced Z Transform[4]

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]

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

---

4   https://en.wikipedia.org/wiki/Advanced%20Z%20Transform

40.5 Star Transform

w:Star Transform[5]

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:

[Star Transform]

$$F^*(s) = \mathcal{L}^*[f(t)] = \sum_{k=0}^{\infty} f(kT)e^{-skT}$$

Star transform pairs can be obtained by plugging $z = e^{sT}$ into the Z-transform pairs, above.

40.6 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 following variable transformations:

   [Bilinear Transform]

$$z = \frac{(T/2)+w}{(T/2)-w}, \quad w = \frac{2}{T}\frac{z-1}{z+1}$$

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:

$$\omega_w = \frac{2}{T}\tan\left(\frac{\omega_s T}{2}\right)$$

---

5   https://en.wikipedia.org/wiki/Star%20Transform

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 characteristic. 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.

[Bilinear Frequency Prewarping]

$\omega = \frac{2}{T}\arctan\left(\omega_a \frac{T}{2}\right).$

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**.

40.7 Wikipedia Resources

- w:Laplace transform[6]
- w:Fourier transform[7]
- w:Z-transform[8]
- w:Star transform[9]
- w:Bilinear transform[10]

---

6   https://en.wikipedia.org/wiki/Laplace%20transform

7   https://en.wikipedia.org/wiki/Fourier%20transform

8   https://en.wikipedia.org/wiki/Z-transform

9   https://en.wikipedia.org/wiki/Star%20transform

10   https://en.wikipedia.org/wiki/Bilinear%20transform

41 System Representations

dhunparserurl Control Systems/System Representations

41.1 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

41.2 General Description

These are the general external system descriptions. y is the system output, h is the system response characteristic, and x is the system input. In the time-variant cases, the general description is also known as the convolution description.

General Description	
Time-Invariant, Non-causal	$y(t) = \int_{-\infty}^{\infty} h(t-r)x(r)dr$
Time-Invariant, Causal	$y(t) = \int_0^t h(t-r)x(r)dr$
Time-Variant, Non-Causal	$y(t) = \int_{-\infty}^{\infty} h(t,r)x(r)dr$
Time-Variant, Causal	$y(t) = \int_0^t h(t,r)x(r)dr$

41.3 State-Space Equations

These are the state-space representations for a system. y is the system output, x is the internal system state, and u is the system input. The matrices A, B, C, and D are coefficient matrices.

[Analog State Equations]

State-Space Equations	
Time-Invariant	$x'(t) = Ax(t) + Bu(t)$
	$y(t) = Cx(t) + Du(t)$
Time-Variant	$x'(t) = A(t)x(t) + B(t)u(t)$
	$y(t) = C(t)x(t) + D(t)u(t)$

These are the digital versions of the equations listed above. All the variables have the same meanings, except that the systems are digital.

[Digital State Equations]

State-Space Equations	
Time-Invariant	$x'[t] = Ax[t] + Bu[t]$ $y[t] = Cx[t] + Du[t]$
Time-Variant	$x'[t] = A[t]x[t] + B[t]u[t]$ $y[t] = C[t]x[t] + D[t]u[t]$

41.4 Transfer Functions

These are the transfer function descriptions, obtained by using the Laplace Transform or the Z-Transform on the general system descriptions listed above. Y is the system output, H is the system transfer function, and X is the system input.

[Analog Transfer Function]

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

[Digital Transfer Function]

Transfer Function	
$Y(z) = H(z)X(z)$	

41.5 Transfer Matrix

This is the transfer matrix system description. This representation can be obtained by taking the Laplace or Z transforms of the state-space equations. In the SISO case, these equations reduce to the transfer function representations listed above. In the MIMO case, $\boldsymbol{Y}$ is the vector of system outputs, $\boldsymbol{X}$ is the vector of system inputs, and $\boldsymbol{H}$ is the transfer matrix that relates each input X to each output Y.

[Analog Transfer Matrix]

Transfer Matrix	
$\mathbf{Y}(s) = \mathbf{H}(s)\mathbf{X}(s)$	

[Digital Transfer Matrix]

Transfer Matrix	
$\mathbf{Y}(z) = \mathbf{H}(z)\mathbf{X}(z)$	

42 Matrix Operations

dhunparserurl Control Systems/Matrix Operations

> For more about this subject, see:
> **Linear Algebra**[a]
> and
> **Engineering Analysis**[b]

---

[a]   https://en.wikibooks.org/wiki/Linear%20Algebra
[b]   https://en.wikibooks.org/wiki/Engineering%20Analysis

42.1 Laws of Matrix Algebra

Matrices must be compatible sizes in order for an operation to be valid:

Addition

Matrices must have the same dimensions (same number of rows, same number of columns).
Matrix addition is commutative:

$$A + B = B + A$$

Multiplication

Matrices must have the same inner dimensions (the number of columns of the first matrix must equal the number of rows in the second matrix). For instance, if matrix A is $n \times m$, and matrix B is $m \times k$, then we can multiply:

$$AB = C$$

Where C is an $n \times k$ matrix. Matrix multiplication is not commutative:

$$AB \neq BA$$

Because it is not commutative, the differentiation must be made between "multiplication on the left", and "multiplication on the right".

Division

There is no such thing as division in matrix algebra, although multiplication of the matrix inverse performs the same basic function. To find an inverse, a matrix must be nonsingular, and must have a non-zero determinant.

42.2 Transpose Matrix

The transpose of a matrix, denoted by:

X^T

is the matrix where the rows and columns of X are interchanged. In some instances, the transpose of a matrix is denoted by:

X'

This shorthand notation is used when the superscript T applied to a large number of matrices in a single equation, and the notation would become too crowded otherwise. When this notation is used in the book, derivatives will be denoted explicitly with:

$\frac{d}{dt}X(t)$

42.3 Determinant

The determinant of a matrix it is a scalar value. It is denoted similarly to absolute-value in scalars:

$|X|$

A matrix has an inverse if the matrix is square, and if the determinant of the matrix is non-zero.

42.4 Inverse

The inverse of a matrix A, which we will denote here by "B" is any matrix that satisfies the following equation:

$AB = BA = I$

Matrices that have such a companion are known as "invertible" matrices, or "non-singular" matrices. Matrices which do not have an inverse that satisfies this equation are called "singular" or "non-invertable".

An inverse can be computed in a number of different ways:

1. Append the matrix A with the Identity matrix of the same size. Use row-reductions to make the left side of the matrice an identity. The right side of the appended matrix will then be the inverse:

$$[A|I] \rightarrow [I|B]$$

2. The inverse matrix is given by the adjoint matrix divided by the determinant. The adjoint matrix is the transpose of the cofactor matrix.

$$A^{-1} = \frac{\text{adj}(A)}{|A|}$$

3. The inverse can be calculated from the Cayley-Hamilton Theorem.

42.5 Eigenvalues

The eigenvalues of a matrix, denoted by the Greek letter lambda λ, are the solutions to the characteristic equation of the matrix:

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

Eigenvalues only exist for square matrices. Non-square matrices do not have eigenvalues. If the matrix X is a real matrix, the eigenvalues will either be all real, or else there will be complex conjugate pairs.

42.6 Eigenvectors

The eigenvectors of a matrix are the nullspace solutions of the characteristic equation:

$$(X - \lambda_i I)v_i = 0$$

There is at least one distinct eigenvector for every distinct eigenvalue. Multiples of an eigenvector are also themselves eigenvectors. However, eigenvalues that are not linearly independent are called "non-distinct" eigenvectors, and can be ignored.

42.7 Left-Eigenvectors

Left Eigenvectors are the right-hand nullspace solutions to the characteristic equation:

$$w_i(A - \lambda_i I) = 0$$

These are also the rows of the inverse transition matrix.

42.8 Generalized Eigenvectors

In the case of repeated eigenvalues, there may not be a complete set of n distinct eigenvectors (right or left eigenvectors) associated with those eigenvalues. Generalized eigenvectors can be generated as follows:

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

Because generalized eigenvectors are formed in relation to another eigenvector or generalize eigenvectors, they constitute an ordered set, and should not be used outside of this order.

42.9 Transformation Matrix

The transformation matrix is the matrix of all the eigenvectors, or the ordered sets of generalized eigenvectors:

$$T = [v_1 v_2 \cdots v_n]$$

The inverse transition matrix is the matrix of the left-eigenvectors:

$$T^{-1} = \begin{bmatrix} w'_1 \\ w'_2 \\ \cdots \\ w'_n \end{bmatrix}$$

A matrix can be diagonalized by multiplying by the transition matrix:

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

Or:

$$T^{-1}AT = D$$

If the matrix has an incomplete set of eigenvectors, and therefore a set of generalized eigenvectors, the matrix cannot be diagonalized, but can be converted into Jordan canonical form:

$$T^{-1}AT = J$$

42.10 MATLAB

The MATLAB programming environment was specially designed for matrix algebra and manipulation. The following is a brief refresher about how to manipulate matrices in MATLAB:

Addition

To add two matrices together, use a plus sign (”+”): C = A + B;

Multiplication

To multiply two matrices together use an asterisk (”*”): C = A * B;

If your matrices are not the correct dimensions, MATLAB will issue an error.

Transpose

To find the transpose of a matrix, use the apostrophe (” ' ”): C = A';

Determinant

To find the determinant, use the **det** function: d = det(A);

Inverse

To find the inverse of a matrix, use the function **inv**: C = inv(A);

Eigenvalues and Eigenvectors

To find the eigenvalues and eigenvectors of a matrix, use the **eig** command: [E, V] = eig(A);

Where E is a square matrix with the eigenvalues of A in the diagonal entries, and V is the matrix comprised of the corresponding eigenvectors. If the eigenvalues are not distinct, the eigenvectors will be repeated. MATLAB will not calculate the generalized eigenvectors.

Left Eigenvectors

To find the left eigenvectors, assuming there is a complete set of distinct right-eigenvectors, we can take the inverse of the eigenvector matrix: [E, V] = eig(A); C = inv(V); The rows of C will be the left-eigenvectors of the matrix A.

For more information about MATLAB, see the wikibook MATLAB Programming[1].

---

1   https://en.wikibooks.org/wiki/MATLAB%20Programming

43 Appendix: MATLAB

dhunparserurl Control Systems/MATLAB

> ⚠ **Warning**
>
> 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.

43.1 MATLAB

This page assumes a prior knowledge of the fundamentals of MATLAB. For more information about MATLAB, see MATLAB Programming[a].

---

[a]   https://en.wikibooks.org/wiki/MATLAB%20Programming

MATLAB is a programming language that is specially designed for the manipulation of matrices. Because of its 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. MATLAB has a number of plugin modules called "Toolboxes". Nearly all the functions described below are located in the **control systems toolbox**. If your system has the control systems toolbox installed, you can get more information about the toolbox by typing `help control` at the MATLAB prompt.

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.

MATLAB is a copyrighted product produced by **The Mathworks**. For more information about MATLAB and The Mathworks, see Control Systems/Resources[1].

43.1.1 Input-Output Isolation

In a MIMO system, typically it can be important to isolate a single input-output pair for analysis. Each input corresponds to a single row in the B matrix, and each output corresponds to a single column in the C matrix. For instance, to isolate the 2nd input and the 3rd output, we can create a system:

sys = ss(A, B(:,2), C(3,:), D);

This page will refer to this technique as "input-output isolation".

43.2 Step Response

This operation can be performed using this MATLAB command: `step`

First, let's take a look at the classical approach, with the following system:

$G(s) = \frac{5s+10}{s^2+4s+5}$

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.

Here is a sample screenshot:

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.

This operation can be performed using this MATLAB command: `plot`

If we didn't get an automatic plot, and we want to produce our own, we type:

[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:

$$x' = Ax + Bu$$

$$y = Cx + Du$$

43.3 Classical ↔ Modern

This operation can be performed using this MATLAB command: `tf2ss`

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:

[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.

This operation can be performed using this MATLAB command: `ss2tf`

In a similar vein, we can convert from the Laplace domain back to the state-space representation using the **ss2tf** function, as such:

[NUM, DEN] = ss2tf(A, B, C, D);

Or, if we have more than 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 than 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.

43.4 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);

43.5 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.

43.6 Root Locus Plots

This operation can be performed using this MATLAB command: `rlocus`

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:

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);

The **rlocus** command cannot be used with MIMO systems, so if your system is a MIMO system, you must separate out your coefficient matrices to isolate each separate Input-output pair, and graph each individually.

Here is a sample screenshot:

43.7 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.

43.8 Bode Plots

This operation can be performed using this MATLAB command: `bode`

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:

[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 doesn't require explanation.

This operation can be performed using this MATLAB command: `logspace`

When talking about Bode plots in decibels, it makes the most sense (and is the most common occurrence) to also use a logarithmic frequency scale. To create such a logarithmic sequence in omega, we use the **logspace** command, as such:

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.

The **bode** command, if used with a MIMO system, will use subplots to produce all the input-output relationship graphs on a single plot window. for a system with multiple inputs and multiple outputs, this can become difficult to see clearly. In these cases, it is typically better to separate out your coefficient matrices to isolate each individual input-output pair.

Here is a sample screenshot:

43.9 Nyquist Plots

This operation can be performed using this MATLAB command: `nyquist`

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):

[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.

Like the **bode** command, the **nyquist** command will use subplots to display the input-output relations of MIMO systems on a single plot window. If there are multiple input-output pairs, it can be difficult to see the individual graphs.

Here is a sample screenshot:

43.10 Lyapunov Equations

43.11 Controllability

A controllability matrix can be constructed using the **ctrb** command. The controllability gramian can be constructed using the **gram** command.

43.12 Observability

An observability matrix can be constructed using the command **obsv**

43.13 Empirical Gramians

Empirical gramians can be computed for linear and also nonlinear control systems. The empirical gramian framework **emgr** allows the computation of the controllability, observability and cross gramian; it is compatible with **MATLAB** and **OCTAVE** and does not require the *control systems toolbox.*

43.14 Further reading

- Ogata, Katsuhiko, "Solving Control Engineering Problems with MATLAB", Prentice Hall, New Jersey, 1994.
  0130459070
- MATLAB Programming[2].
- http://octave.sourceforge.net/
- MATLAB Category on ControlTheoryPro.com[3]
- Empirical Gramian Framework[4]

---

2   https://en.wikibooks.org/wiki/MATLAB%20Programming
3   http://wikis.controltheorypro.com/index.php?title=Category:MATLAB
4   http://gramian.de

44 Glossary and List of Equations

dhunparserurl Control Systems/Glossary
The following is a listing of some of the most important terms from the book, along with a short definition or description.

44.1 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.

Adaptive Gain

when control gain is varied depending on system state or condition, such as a disturbance

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.

ARMA

Autoregressive Moving Average, see `http://en.wikipedia.org/wiki/Autoregressive_moving_average_model`

ATO

Analog Timed Output. Control loop output is correlated to a timed contact closure.

A/M

Auto-Manual. Control modes, where auto typically means output is computer-driven, calculated while manual can be field-driven or merely using a static setpoint.

Bilinear Transform

a variant of the Z-transform, see `http://en.wikibooks.org/wiki/Digital_Signal_Processing/Bilinear_Transform`

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 log scale 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**.

Cascade

When the output of a control loop is fed to/from another loop.

Causal

A system whose output 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.

Closed Loop

a controlled system using feedback or feedforward

Compensator

A Control System that augments the shortcomings of another system.

Condition Number

Conditional Stability

A system with variable gain is conditionally stable if it is BIBO stable for certain values of gain, but not BIBO stable for other values of gain.

Continuous-Time

A system or signal that is defined at all points t.

Control Rate

the rate at which control is computed and any appropriate output sent. Lower bound is sample rate.

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.

CQI

Control Quality Index, $= 1 - abs(PV - SP)/max[PVmax - SP, SP - PVmin]$, 1 being ideal.

CV

Controlled variable

44.2 D, E, F

Damping Ratio

A constant that determines the damping properties of a system.

Deadtime

time shift between the output change and the related effect (typ. at least one control sample). One sees "Lag" used for this action sometimes.

Digital

A system that is both **discrete-time**, and **quantized**.

Direct action

target output increase is required to bring the process variable (PV) to setpoint (SP) when PV is below SP. Thus, PV increases with output increase directly.

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 doesn't have memory. See **Instantaneous**, **Memory**.

Eigenvalues

Solutions to the characteristic equation of a matrix. If the matrix is itself a function of time, the eigenvalues might be functions of time. In this case, they are frequently called **eigenfunctions**.

Eigenvectors

The nullspace vectors of the characteristic equation for particular eigenvalues. Used to determine state-transitions, among other things. See `http://en.wikibooks.org/wiki/Control_Systems/Eigenvalues_and_Eigenvectors`

Euler's Formula

An equation that relates complex exponentials to complex sinusoids.

Exponential Weighted Average (EWA)

Apportions fractional weight to new and existing data to form a working average. Example EWA=0.70*EWA+0.30*latest, see Filtering.

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.

Feedforward

whwn apriori knowledge is used to forecast at least part of the control response.

Filtering (noise)

Use of signal smoothing techniques to reject undesirable components like noise. Can be as simple as using exponential weighted averaging on the input.

Final Value Theorem

A theorem that allows the steady-state value of a system to be determined from the transfer function.

FOH

First order hold

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. See `http://en.wikibooks.org/wiki/Waves/Fourier_Transforms`

44.3 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 multiplier in a system that is typically implemented as an amplifier or attenuator. Gain can be changed, but is typically not a function of time. Adaptive control can use time-adaptive gains that change with 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

Property of a system whose scaled input results in an equally scaled output.

Hybrid Systems

Systems which have both analog and digital components.

Impulse

A function denoted $\delta(t)$, 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 doesn't have memory, and if the current output of the system is only dependent on the current input. See **Dynamic**, **Memory**.

Integrated Absolute Error (IAE)

absolute error (ideal vs actual performance) is integrated over the analysis period.

Integrated Squared Error (ISE)

squared error (ideal vs actual performance) is integrated over the analysis period.

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.

44.4 J, K, L

Lag

The observed process impact from an output is slower than the control rate.

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

Left-hand nullspace solutions to the characteristic equation of a matrix for given eigenvalues. The rows of the inverse transition matrix.

Linear

A system that satisfies the **superposition principle**. See **Additive** and **Homogeneous**.

Linear Time-Invariant

LTI. See **Linear**, and **Time-Invariant**.

Low Clamp

User-applied lower bound on control output signal.

L/R

Local/Remote operation.

LQR

Linear Quadratic Regulator.

Lumped

A system with a finite number of states, or a finite number of state variables.

44.5 M, N, O

Magnitude

the gain component of frequency response. This is often all that is considered in saying a discrete filter's response is well matched to the analog's. It is the DC gain at 0 frequency.

Marginal Stability

A system has an oscillatory response, as determined by having imaginary poles or imaginary eigenvalues.

Mason's Rule

see http://en.wikipedia.org/wiki/Mason's_rule

MATLAB

Commercial software having a Control Systems toolbox. Also see Octave.

Memory

A system has memory if its current output is dependent on previous and current inputs.

MFAC

Model Free Adaptive Control.

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.

MPC

Model Predictive Control.

MRAC

Model Reference Adaptive Control.

MV

can denote Manipulated variable or Measured variable (not the same)

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.

OCTAVE

Open-source software having a Control Systems toolbox. Also see MATLAB.

Offset

The discrepancy between desired and actual value after settling. P-only control can give offset.

Oliver Heaviside

Electrical Engineer, Introduced the Laplace Transform as a tool for control engineering.

Open Loop

when the system is not closed, its behavior has a free-running component rather than controlled

Optimal Control

A branch of control engineering that deals with the minimization of system cost, or maximization of system performance.

Order

The order of a polynomial is the highest exponent of the independent 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.

Overshoot

measures the extent of system response against desired (setpoint tracking).

44.6 P, Q, R

Parabolic

A parabolic input is defined by the equation $\frac{1}{2}t^2 u(t)$.

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.

Phase

the directional component of frequency response, not typically well-matched between a discrete filter equivalent to the analog version, especially as frequency approaches the Nyquist limit. The final value in the limit drives system stability, and stems from the poles and zeros of the characteristic equation.

PID

Proportional-Integral-Derivative

Plant

A central system which has been provided, and must be analyzed or controlled.

PLC

Programmable Logic Controller

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.

PSD

The power spectral density which shows the distribution of power in the spectrum of a particular signal.

Pulse Response

The response of a digital system to a unit step input, in terms of the transfer matrix.

PV

Process variable

Quantized

A system is quantized if it can only output certain discrete values.

Quarter-decay

the time or number of control rates required for process overshoot to be limited to within 1/4 of the maximum peak overshoot (PO) after a SP change. If the PO is 25% at sample time N, this would be time N+k when subsequent PV remains < SP*1.0625, presuming the process is settling.

Raise-Lower

Output type that works from present position rather than as a completely new computed spanned output. For R/L, the % change should be applied to the working clamps i.e. 5%(hi clamp-lo clamp).

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.

Reverse action

target output decrease is required to bring the process variable (PV) to setpoint (SP) when PV is below SP. Thus, PV decreases with output increase.

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.

44.7 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. Reflects the lower bound for Control rate.

SCADA

Supervisory Control and Data Acquisition.

S-Domain

The domain of the Laplace Transform of a signal or system.

Second-order 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.

SISO

Single input, single output.

Span

the designed operation region of the item,=high range-low range. Working span can be smaller if output clamps are used.

Stability

Typically "BIBO Stability", a system with a well-behaved input will result in a well-behaved output. "Well-behaved" in this sense is arbitrary.

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

A coefficient matrix, or a matrix function that relates how the system state changes in response to the system input. In time-invariant systems, the state-transition matrix is the matrix exponential of the system 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 output equation. See http://en.wikibooks.org/wiki/Control_Systems/State-Space_Equations

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. A unit step is a setpoint change for setpoint tracking.

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 homogeneous.

System Identification

method of trying to identify the system characterization , typically through least squares analysis of input,output and noise data vectors. May use ARMA type framework.

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 its 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.

Uniform Stability

Also "Uniform BIBO Stability", a system where an input signal in the range [0, 1] results in a finite output from the initial time until infinite time. See `http://en.wikibooks.org/wiki/Control_Systems/Stability`.

Unit Step

An input defined by $u(t)$. Practically, a setpoint change.

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.

44.8 W, X, Y, Z

W-plane

Reference plane used in the bilinear transform.

Wind-up

when the numerics of computed control adjustment can "wind-up", yielding control correction with an inappropriate component unless prevented. An example is the "I" contribution of PID if output has been disconnected during PID calculation

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

The response of a system with zero external input. Relies only on the value of the system state to produce output.

Zero State Response

The response of the system with zero system state. The output of the system depends only on the system input.

ZOH

Zero order hold.

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. See `http://en.wikibooks.org/wiki/Digital_Signal_Processing/Z_Transform`

45 List of Equations

dhunparserurl Control Systems/List of Equations

The following is a list of the important equations from the text, arranged by subject. For more information about these equations, including the meaning of each variable and symbol, the uses of these functions, or the derivations of these equations, see the relevant pages in the main text.

45.1 Fundamental Equations

[Euler's Formula]

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

[Convolution]

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

[Convolution Theorem]

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

[Characteristic Equation]

$$|A - \lambda I| = 0$$
$$Av = \lambda v$$
$$wA = \lambda w$$

[Decibels]

$$dB = 20\log(C)$$

45.2 Basic Inputs

[Unit Step Function]

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

[Unit Ramp Function]

$$r(t) = tu(t)$$

$$p(t) = \tfrac{1}{2}t^2 u(t)$$

45.3 Error Constants

[Position Error Constant]

$$K_p = \lim_{s \to 0} G(s)$$
$$K_p = \lim_{z \to 1} G(z)$$

[Velocity Error Constant]

$$K_v = \lim_{s \to 0} sG(s)$$
$$K_v = \lim_{z \to 1} (z-1)G(z)$$

[Acceleration Error Constant]

$$K_a = \lim_{s \to 0} s^2 G(s)$$
$$K_a = \lim_{z \to 1} (z-1)^2 G(z)$$

45.4 System Descriptions

[General System Description]

$$y(t) = \int_{-\infty}^{\infty} g(t,r)x(r)dr$$

[Convolution Description]

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

[Transfer Function Description]

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

[State-Space Equations]

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

[Transfer Matrix]

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

[Transfer Matrix Description]

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

$$M = \frac{y_{out}}{y_{in}} = \sum_{k=1}^{N} \frac{M_k \Delta_k}{\Delta}$$

45.5 Feedback Loops

[Closed-Loop Transfer Function]

$$H_{cl}(s) = \frac{KGp(s)}{1 + KGp(s)Gb(s)}$$

[Open-Loop Transfer Function]

$$H_{ol}(s) = KGp(s)Gb(s)$$

[Characteristic Equation]

$$F(s) = 1 + H_{ol}$$

45.6 Transforms

[Laplace Transform]

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

[Inverse Laplace Transform]

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

[Fourier Transform]

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

[Inverse Fourier Transform]

$$f(t) = \mathcal{F}^{-1}\{F(j\omega)\} = \frac{1}{2\pi} \int_{-\infty}^\infty F(j\omega)e^{-j\omega t}d\omega$$

[Star Transform]

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

[Z Transform]

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

[Inverse Z Transform]

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

[Modified Z Transform]

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

45.7 Transform Theorems

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

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

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

45.8 State-Space Methods

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

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

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

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

$$x(t) = \phi(t, t_0) x(t_0) + \int_{t_0}^{t} \phi(\tau, t_0) B(\tau) u(\tau) d\tau$$

$$x[n] = \phi[n, n_0] x[t_0] + \sum_{m=n_0}^{n} \phi[n, m+1] B[m] u[m]$$

$$G(t, \tau) = \begin{cases} C(\tau)\phi(t,\tau)B(\tau) & \text{if } t \geq \tau \\ 0 & \text{if } t < \tau \end{cases}$$

$$G[n] = \begin{cases} CA^{k-1}N & \text{if } k > 0 \\ 0 & \text{if } k \leq 0 \end{cases}$$

45.9 Root Locus

$$1 + KG(s)H(s) = 0$$

$$1 + K\overline{GH}(z) = 0$$

$$\angle KG(s)H(s) = 180°$$

$$\angle K\overline{GH}(z) = 180°$$

$N_a = P - Z$

<div align="right">**[Angle of Asymptotes]**</div>

$\phi_k = (2k+1)\frac{\pi}{P-Z}$

<div align="right">**[Origin of Asymptotes]**</div>

$\sigma_0 = \frac{\sum_P - \sum_Z}{P-Z}$

<div align="right">**[Breakaway Point Locations]**</div>

$\frac{G(s)H(s)}{ds} = 0$ or $\frac{\overline{GH}(z)}{dz} = 0$

45.10 Lyapunov Stability

<div align="right">**[Lyapunov Equation]**</div>

$MA + A^T M = -N$

45.11 Controllers and Compensators

<div align="right">**[PID]**</div>

$D(s) = K_p + \frac{K_i}{s} + K_d s$

$D(z) = K_p + K_i \frac{T}{2}\left[\frac{z+1}{z-1}\right] + K_d \left[\frac{z-1}{Tz}\right]$

46 Resources and Further Reading

dhunparserurl Control Systems/Resources

46.1 Wikibooks

A number of wikibooks exist on topics that are (a) prerequisites to this book (b) companion pieces 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.

46.1.1 Prerequisite Books

- Linear algebra[1]
- Linear Algebra with Differential Equations[2]
- Complex Numbers[3]
- Calculus[4]
- Signals and Systems[5]

46.1.2 Companion Books

- Engineering Analysis[6]
- Engineering Tables[7]
- Analog and Digital Conversion[8]
- MATLAB Programming[9]

46.1.3 Books for Further Reading

- Signal Processing[10]
- Digital Signal Processing[11]

---

1   https://en.wikibooks.org/wiki/Linear%20algebra
2   https://en.wikibooks.org/wiki/Linear%20Algebra%20with%20Differential%20Equations
3   https://en.wikibooks.org/wiki/Algebra%2FComplex%20Numbers
4   https://en.wikibooks.org/wiki/Calculus
5   https://en.wikibooks.org/wiki/Signals%20and%20Systems
6   https://en.wikibooks.org/wiki/Engineering%20Analysis
7   https://en.wikibooks.org/wiki/Engineering%20Tables
8   https://en.wikibooks.org/wiki/Analog%20and%20Digital%20Conversion
9   https://en.wikibooks.org/wiki/MATLAB%20Programming
10   https://en.wikibooks.org/wiki/Signal%20Processing
11   https://en.wikibooks.org/wiki/Digital%20Signal%20Processing

- Communication Systems[12]
- Embedded Control Systems Design[13]

46.2 Wikiversity

v:Control Systems[14]

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:Control Systems[15]

Wikiversity is also a place to host learning materials, such as assignments, tests, and reading plans. It is the goal of the authors of this book to create such materials for use in conjunction with this book. As such materials are added to wikiversity, they will be referenced here.

46.3 Wikipedia

There are a number of Wikipedia articles on the topics covered in this book, and those articles will be linked to from the appropriate pages of this book. However, some of the articles that are of general use to the book are:

- w:Control theory[16]
- w:Control engineering[17]
- w:Process control[18]
  A complete listing of all Wikipedia articles related to this topic can be found at:
- w:Category:Control theory[19].

46.4 Software

46.4.1 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://web.archive.org/20041230124431/www.geocities.com/aseldawy/root_
locus.html
```

Explicit permission has been granted by the author of the program to include screenshots on wikibooks. Images generated from the Root-Locus program should be included in Category:Root Locus Images[20], and appropriately tagged as a screenshot of a free software program.

---

12   https://en.wikibooks.org/wiki/Communication%20Systems
13   https://en.wikibooks.org/wiki/Embedded%20Control%20Systems%20Design
14   https://en.wikiversity.org/wiki/Control%20Systems
15   https://en.wikiversity.org/wiki/Control%20Systems
16   https://en.wikipedia.org/wiki/Control%20theory
17   https://en.wikipedia.org/wiki/Control%20engineering
18   https://en.wikipedia.org/wiki/Process%20control
19   https://en.wikipedia.org/wiki/Category%3AControl%20theory
20   https://en.wikibooks.org/wiki/%3ACategory%3ARoot%20Locus%20Images

46.4.2 MATLAB

MATLAB, Simulink, the Control Systems Toolbox and the Symbolic Toolbox are trademarks of The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders. For more information about MATLAB, or to purchase a copy, visit:

http://www.themathworks.com

For information about the proper way to refer to MATLAB, please see:

http://www.mathworks.com/company/pressroom/editorial_guidelines.html

All MATLAB code appearing in this book has been released under the terms of the GFDL by the respective authors. All screenshots, graphs, and images relating to MATLAB have been produced in **Octave**, with changes to the original MATLAB code made as necessary.

46.4.3 Octave

Octave is a free open-source alternative program to MATLAB. Octave utilizes a scripting language that is very similar to that of MATLAB, although there are several differences. Most of the basic examples described in this book will work equally well in MATLAB or Octave, with no changes or only minor changes. For more information, or to download a copy of Octave, visit:

http://octave.sourceforge.net

46.4.4 Commercial Vendors

The following are some common vendors of control-related hardware and software. These links are for personal interest only, and do not constitute an official endorsement of the companies by Wikibooks.

- Systems Technology, Inc[21]
- SimApp - Dynamic Simulation Made Easy[22]

46.5 External Publications

The following books and resources were used as reference works in the creation of this wikibook (books listed in alphabetical order).
- Brogan, William L, *Modern Control Theory*, 3rd Edition, 1991.
  0135897637
- Chen, Chi-Tsong, *Linear System Theory and Design*, 3rd Edition, 1999.
  0195117778
- Dorf and Bishop, *Modern Control Systems*, 10th Edition, Prentice Hall, 2005.
  0131277650

---

21   http://www.programcc.com/about.asp
22   http://www.simapp.com

- Hamming, Richard, *Numerical Methods for Scientists and Engineers*, 2nd edition, Dover, 1987.
  0486652416
- Kalman, R. E., *When is a linear control system optimal*, ASME Transactions, Journal of Basic Engineering, 1964
- Kalman, R. E., *On the General Theory of Control Systems*, IRE Transactions on Automatic Control, Volume 4, Issue 3, p110, 1959. ISSN 0096199X
- Ogata, Katsuhiko, *Solving Control Engineering Problems with MATLAB*, Prentice Hall, New Jersey, 1994.
  0130459070
- Phillips and Nagle, *Digital Control System Analysis and Design*, 3rd Edition, Prentice Hall, 1995.
  013309832X
  The following books and resources are suitable for further reading.
- DiStefano, Stubberud, Williams, *Schaum's Outline Series Feedback and Control Systems*, 2nd Edition, 1997.
  0070170479
- Franklin, Powell, Workman, *Digital Control of Dynamic Systems*, 3rd Edition, 1997.
  9780201820546
- Brosilow, Joseph, *Techniques of Model-Based Control*, 2002.
  013028078X

46.6 External Resources

- IEEE Control Systems Society[23]
- ControlTheoryPro.com[24] A place for controls theory, application, and modeling help.
- University of Michigan Chemical Engineering Process Dynamics and Controls Open Textbook[25]

---

23 http://www.ieeecss.org/
24 http://wikis.ControlTheoryPro.com
25 http://controls.engin.umich.edu/wiki/index.php/Main_Page

47 Contributors

Edits	User
2	1989[1]
1	1997kB[2]
1	1sfoerster[3]
1	Abletried~enwikibooks[4]
16	Adrignola[5]
1	Agentx3r~enwikibooks[6]
1	Ah3kal[7]
1	Alej27[8]
2	Amorini[9]
2	Anith 555[10]
2	Anon1713[11]
1	Antonysigma[12]
1	Arjunazadi[13]
5	Atcovi[14]
25	Avicennasis[15]
1	Avijit525[16]
3	Az1568[17]
4	Bestable[18]
66	Billymac00[19]
2	Bo Dorku~enwikibooks[20]

1   https://en.wikibooks.org/wiki/User:1989
2   https://en.wikibooks.org/wiki/User:1997kB
3   https://en.wikibooks.org/wiki/User:1sfoerster
4   https://en.wikibooks.org/w/index.php%3ftitle=User:Abletried~enwikibooks&action=edit&redlink=1
5   https://en.wikibooks.org/wiki/User:Adrignola
6   https://en.wikibooks.org/w/index.php%3ftitle=User:Agentx3r~enwikibooks&action=edit&redlink=1
7   https://en.wikibooks.org/wiki/User:Ah3kal
8   https://en.wikibooks.org/w/index.php%3ftitle=User:Alej27&action=edit&redlink=1
9   https://en.wikibooks.org/w/index.php%3ftitle=User:Amorini&action=edit&redlink=1
10   https://en.wikibooks.org/w/index.php%3ftitle=User:Anith_555&action=edit&redlink=1
11   https://en.wikibooks.org/w/index.php%3ftitle=User:Anon1713&action=edit&redlink=1
12   https://en.wikibooks.org/w/index.php%3ftitle=User:Antonysigma&action=edit&redlink=1
13   https://en.wikibooks.org/w/index.php%3ftitle=User:Arjunazadi&action=edit&redlink=1
14   https://en.wikibooks.org/wiki/User:Atcovi
15   https://en.wikibooks.org/wiki/User:Avicennasis
16   https://en.wikibooks.org/w/index.php%3ftitle=User:Avijit525&action=edit&redlink=1
17   https://en.wikibooks.org/wiki/User:Az1568
18   https://en.wikibooks.org/w/index.php%3ftitle=User:Bestable&action=edit&redlink=1
19   https://en.wikibooks.org/w/index.php%3ftitle=User:Billymac00&action=edit&redlink=1
20   https://en.wikibooks.org/w/index.php%3ftitle=User:Bo_Dorku~enwikibooks&action=edit&redlink=1

1	Cbarlog[21]
1	Chazz[22]
1	Chih WANG[23]
1	CommonsDelinker[24]
1	Congquaner[25]
1	Constant314[26]
1	DSP-user[27]
1	Dagonet Iestwyn[28]
1	Danny B.[29]
1	DannyS712[30]
1	Davetcoleman[31]
1	DavidCary[32]
1	Deepcyan[33]
2	Derby County FC[34]
8	Dirk Hünniger[35]
1	Discostu5[36]
1	Doleszki~enwikibooks[37]
1	Druzhnik[38]
2	Ducleotide[39]
1	Edwardvear[40]
1	Empato[41]
6	Ennui~enwikibooks[42]
9	Esj88[43]
1	Esteban16[44]

21 https://en.wikibooks.org/w/index.php%3ftitle=User:Cbarlog&action=edit&redlink=1
22 https://en.wikibooks.org/wiki/User:Chazz
23 https://en.wikibooks.org/w/index.php%3ftitle=User:Chih_WANG&action=edit&redlink=1
24 https://en.wikibooks.org/wiki/User:CommonsDelinker
25 https://en.wikibooks.org/w/index.php%3ftitle=User:Congquaner&action=edit&redlink=1
26 https://en.wikibooks.org/w/index.php%3ftitle=User:Constant314&action=edit&redlink=1
27 https://en.wikibooks.org/wiki/User:DSP-user
28 https://en.wikibooks.org/w/index.php%3ftitle=User:Dagonet_Iestwyn&action=edit&redlink=1
29 https://en.wikibooks.org/wiki/User:Danny_B.
30 https://en.wikibooks.org/wiki/User:DannyS712
31 https://en.wikibooks.org/w/index.php%3ftitle=User:Davetcoleman&action=edit&redlink=1
32 https://en.wikibooks.org/wiki/User:DavidCary
33 https://en.wikibooks.org/w/index.php%3ftitle=User:Deepcyan&action=edit&redlink=1
34 https://en.wikibooks.org/w/index.php%3ftitle=User:Derby_County_FC&action=edit&redlink=1
35 https://en.wikibooks.org/wiki/User:Dirk_H%25C3%25BCnniger
36 https://en.wikibooks.org/wiki/User:Discostu5
37 https://en.wikibooks.org/w/index.php%3ftitle=User:Doleszki~enwikibooks&action=edit&redlink=1
38 https://en.wikibooks.org/w/index.php%3ftitle=User:Druzhnik&action=edit&redlink=1
39 https://en.wikibooks.org/w/index.php%3ftitle=User:Ducleotide&action=edit&redlink=1
40 https://en.wikibooks.org/w/index.php%3ftitle=User:Edwardvear&action=edit&redlink=1
41 https://en.wikibooks.org/w/index.php%3ftitle=User:Empato&action=edit&redlink=1
42 https://en.wikibooks.org/w/index.php%3ftitle=User:Ennui~enwikibooks&action=edit&redlink=1
43 https://en.wikibooks.org/w/index.php%3ftitle=User:Esj88&action=edit&redlink=1
44 https://en.wikibooks.org/wiki/User:Esteban16

3   Federhalter[45]

2   Feraudyh[46]

1   Fishpi[47]

1   Foreign1~enwikibooks[48]

2   Fragapanagos[49]

1   Frigotoni[50]

1   Glaisher[51]

2   Hagindaz[52]

2   Hammer of Moradin[53]

1   Helptry[54]

2   Herbythyme[55]

1   Herbythyme is the Antichrist of Mumfum[56]

1   HethrirBot[57]

1   Hippias[58]

1   Hyiltiz[59]

3   Hypergeek14[60]

8   Inductiveload[61]

1   Istevie[62]

5   JackBot[63]

3   JackPotte[64]

1   Jakec[65]

1   Javalenok[66]

1   Jawnsy[67]

2   JenVan[68]

1   Jfmantis[69]

---

45   https://en.wikibooks.org/wiki/User:Federhalter

46   https://en.wikibooks.org/wiki/User:Feraudyh

47   https://en.wikibooks.org/w/index.php%3ftitle=User:Fishpi&action=edit&redlink=1

48   https://en.wikibooks.org/w/index.php%3ftitle=User:Foreign1~enwikibooks&action=edit&
redlink=1

49   https://en.wikibooks.org/w/index.php%3ftitle=User:Fragapanagos&action=edit&redlink=1

50   https://en.wikibooks.org/wiki/User:Frigotoni

51   https://en.wikibooks.org/wiki/User:Glaisher

52   https://en.wikibooks.org/wiki/User:Hagindaz

53   https://en.wikibooks.org/w/index.php%3ftitle=User:Hammer_of_Moradin&action=edit&
redlink=1

54   https://en.wikibooks.org/w/index.php%3ftitle=User:Helptry&action=edit&redlink=1

55   https://en.wikibooks.org/wiki/User:Herbythyme

56   https://en.wikibooks.org/w/index.php%3ftitle=User:Herbythyme_is_the_Antichrist_of_
Mumfum&action=edit&redlink=1

57   https://en.wikibooks.org/wiki/User:HethrirBot

58   https://en.wikibooks.org/w/index.php%3ftitle=User:Hippias&action=edit&redlink=1

59   https://en.wikibooks.org/wiki/User:Hyiltiz

60   https://en.wikibooks.org/w/index.php%3ftitle=User:Hypergeek14&action=edit&redlink=1

61   https://en.wikibooks.org/wiki/User:Inductiveload

62   https://en.wikibooks.org/w/index.php%3ftitle=User:Istevie&action=edit&redlink=1

63   https://en.wikibooks.org/wiki/User:JackBot

64   https://en.wikibooks.org/wiki/User:JackPotte

65   https://en.wikibooks.org/wiki/User:Jakec

66   https://en.wikibooks.org/w/index.php%3ftitle=User:Javalenok&action=edit&redlink=1

67   https://en.wikibooks.org/w/index.php%3ftitle=User:Jawnsy&action=edit&redlink=1

68   https://en.wikibooks.org/w/index.php%3ftitle=User:JenVan&action=edit&redlink=1

69   https://en.wikibooks.org/wiki/User:Jfmantis

25	Jguk[70]
2	Jianhui67[71]
1	Jmah[72]
4	Joe Schmedley[73]
12	Jomegat[74]
2	Josephkiran[75]
1	Kayau[76]
2	Kevinp2[77]
5	Leaderboard[78]
3	Leisuresuitwally[79]
2	Livingthingdan[80]
11	Lpkeys[81]
4	Macsdev~enwikibooks[82]
1	Markcc[83]
1	Martin Urbanec[84]
1	Mike.lifeguard[85]
1	Minorax[86]
2	Mintz l[87]
1	Mreiki[88]
4	Mrjulesd[89]
3	Mstachowsky[90]
1	Murughendra[91]
1	N.masarone[92]
1	Napalm Llama[93]
1	Neils51[94]

70   https://en.wikibooks.org/wiki/User:Jguk
71   https://en.wikibooks.org/wiki/User:Jianhui67
72   https://en.wikibooks.org/w/index.php%3ftitle=User:Jmah&action=edit&redlink=1
73   https://en.wikibooks.org/wiki/User:Joe_Schmedley
74   https://en.wikibooks.org/wiki/User:Jomegat
75   https://en.wikibooks.org/w/index.php%3ftitle=User:Josephkiran&action=edit&redlink=1
76   https://en.wikibooks.org/wiki/User:Kayau
77   https://en.wikibooks.org/w/index.php%3ftitle=User:Kevinp2&action=edit&redlink=1
78   https://en.wikibooks.org/wiki/User:Leaderboard
79   https://en.wikibooks.org/w/index.php%3ftitle=User:Leisuresuitwally&action=edit&redlink=1
80   https://en.wikibooks.org/w/index.php%3ftitle=User:Livingthingdan&action=edit&redlink=1
81   https://en.wikibooks.org/w/index.php%3ftitle=User:Lpkeys&action=edit&redlink=1
82   https://en.wikibooks.org/w/index.php%3ftitle=User:Macsdev~enwikibooks&action=edit&redlink=1
83   https://en.wikibooks.org/w/index.php%3ftitle=User:Markcc&action=edit&redlink=1
84   https://en.wikibooks.org/wiki/User:Martin_Urbanec
85   https://en.wikibooks.org/wiki/User:Mike.lifeguard
86   https://en.wikibooks.org/wiki/User:Minorax
87   https://en.wikibooks.org/wiki/User:Mintz_l
88   https://en.wikibooks.org/w/index.php%3ftitle=User:Mreiki&action=edit&redlink=1
89   https://en.wikibooks.org/wiki/User:Mrjulesd
90   https://en.wikibooks.org/w/index.php%3ftitle=User:Mstachowsky&action=edit&redlink=1
91   https://en.wikibooks.org/w/index.php%3ftitle=User:Murughendra&action=edit&redlink=1
92   https://en.wikibooks.org/w/index.php%3ftitle=User:N.masarone&action=edit&redlink=1
93   https://en.wikibooks.org/w/index.php%3ftitle=User:Napalm_Llama&action=edit&redlink=1
94   https://en.wikibooks.org/w/index.php%3ftitle=User:Neils51&action=edit&redlink=1

1	Ninjastrikers[95]
5	Nithinvgeorge[96]
1	Nonstandard[97]
2	Nostraticispeak[98]
14	Nrs135[99]
1	Ojan[100]
1	Okashi no kuni[101]
8	Panic2k4[102]
1	Pedro Fonini[103]
9	Pi zero[104]
34	Pierre5018[105]
45	PokestarFan[106]
18	QuiteUnusual[107]
2	Razr Nation[108]
19	Recent Runes[109]
2	Redberry76[110]
1	Renamed user wZdBmxEPfs[111]
1	Rmaax[112]
2	Ro890Z[113]
1	Roman123~enwikibooks[114]
1	Rotlink[115]
1	SPat[116]
3	Sagie[117]
2	Satyabrata[118]

95   https://en.wikibooks.org/wiki/User:Ninjastrikers
96   https://en.wikibooks.org/w/index.php%3ftitle=User:Nithinvgeorge&action=edit&redlink=1
97   https://en.wikibooks.org/w/index.php%3ftitle=User:Nonstandard&action=edit&redlink=1
98   https://en.wikibooks.org/w/index.php%3ftitle=User:Nostraticispeak&action=edit&redlink=1
99   https://en.wikibooks.org/w/index.php%3ftitle=User:Nrs135&action=edit&redlink=1
100 https://en.wikibooks.org/w/index.php%3ftitle=User:Ojan&action=edit&redlink=1
101 https://en.wikibooks.org/w/index.php%3ftitle=User:Okashi_no_kuni&action=edit&redlink=1
102 https://en.wikibooks.org/wiki/User:Panic2k4
103 https://en.wikibooks.org/wiki/User:Pedro_Fonini
104 https://en.wikibooks.org/wiki/User:Pi_zero
105 https://en.wikibooks.org/w/index.php%3ftitle=User:Pierre5018&action=edit&redlink=1
106 https://en.wikibooks.org/wiki/User:PokestarFan
107 https://en.wikibooks.org/wiki/User:QuiteUnusual
108 https://en.wikibooks.org/wiki/User:Razr_Nation
109 https://en.wikibooks.org/wiki/User:Recent_Runes
110 https://en.wikibooks.org/wiki/User:Redberry76
111 https://en.wikibooks.org/w/index.php%3ftitle=User:Renamed_user_wZdBmxEPfs&action=edit&redlink=1
112 https://en.wikibooks.org/w/index.php%3ftitle=User:Rmaax&action=edit&redlink=1
113 https://en.wikibooks.org/w/index.php%3ftitle=User:Ro890Z&action=edit&redlink=1
114 https://en.wikibooks.org/w/index.php%3ftitle=User:Roman123~enwikibooks&action=edit&redlink=1
115 https://en.wikibooks.org/w/index.php%3ftitle=User:Rotlink&action=edit&redlink=1
116 https://en.wikibooks.org/wiki/User:SPat
117 https://en.wikibooks.org/w/index.php%3ftitle=User:Sagie&action=edit&redlink=1
118 https://en.wikibooks.org/w/index.php%3ftitle=User:Satyabrata&action=edit&redlink=1

1	Sberkane[119]
3	ScaAr[120]
1	Scruff323[121]
1	Sdayal[122]
1	Shaffers21[123]
4	Simoneau[124]
1	Sonia[125]
1	Sonicwave32[126]
10	Spradlig[127]
1	Strange quark[128]
4	Stïnger[129]
1	Supermackin˜enwikibooks[130]
2	Syum90[131]
1	Tdenewiler[132]
6	Texvc2LaTeXBot[133]
4	Thenub314[134]
1	Tickle55[135]
4	Tim.greatrex[136]
2	Tropicalkitty[137]
1	Trunks ishida[138]
1	Turkmen[139]
3	Ubigene[140]
1	Upul[141]
1	Van der Hoorn[142]
1	Vito Genovese[143]

119 https://en.wikibooks.org/w/index.php%3ftitle=User:Sberkane&action=edit&redlink=1
120 https://en.wikibooks.org/w/index.php%3ftitle=User:ScaAr&action=edit&redlink=1
121 https://en.wikibooks.org/w/index.php%3ftitle=User:Scruff323&action=edit&redlink=1
122 https://en.wikibooks.org/w/index.php%3ftitle=User:Sdayal&action=edit&redlink=1
123 https://en.wikibooks.org/w/index.php%3ftitle=User:Shaffers21&action=edit&redlink=1
124 https://en.wikibooks.org/wiki/User:Simoneau
125 https://en.wikibooks.org/wiki/User:Sonia
126 https://en.wikibooks.org/wiki/User:Sonicwave32
127 https://en.wikibooks.org/wiki/User:Spradlig
128 https://en.wikibooks.org/wiki/User:Strange_quark
129 https://en.wikibooks.org/wiki/User:St%25C3%25AFnger
130 https://en.wikibooks.org/wiki/User:Supermackin˜enwikibooks
131 https://en.wikibooks.org/w/index.php%3ftitle=User:Syum90&action=edit&redlink=1
132 https://en.wikibooks.org/w/index.php%3ftitle=User:Tdenewiler&action=edit&redlink=1
133 https://en.wikibooks.org/wiki/User:Texvc2LaTeXBot
134 https://en.wikibooks.org/wiki/User:Thenub314
135 https://en.wikibooks.org/w/index.php%3ftitle=User:Tickle55&action=edit&redlink=1
136 https://en.wikibooks.org/w/index.php%3ftitle=User:Tim.greatrex&action=edit&redlink=1
137 https://en.wikibooks.org/wiki/User:Tropicalkitty
138 https://en.wikibooks.org/w/index.php%3ftitle=User:Trunks_ishida&action=edit&redlink=1
139 https://en.wikibooks.org/wiki/User:Turkmen
140 https://en.wikibooks.org/w/index.php%3ftitle=User:Ubigene&action=edit&redlink=1
141 https://en.wikibooks.org/wiki/User:Upul
142 https://en.wikibooks.org/wiki/User:Van_der_Hoorn
143 https://en.wikibooks.org/wiki/User:Vito_Genovese

1	Wargo[144]
1	Warlock31415[145]
1823	Whiteknight[146]
1	WikiBayer[147]
4	Wknight8111[148]
1	XMollioTKs[149]
7	Xania[150]
5	Xris[151]
1	YMS[152]
1	Yeus[153]
1	Z35~enwikibooks[154]
3	Zhaopatrick[155]
3	Zoomzoom~enwikibooks[156]
2	~riley[157]

144 https://en.wikibooks.org/wiki/User:Wargo
145 https://en.wikibooks.org/w/index.php%3ftitle=User:Warlock31415&action=edit&redlink=1
146 https://en.wikibooks.org/wiki/User:Whiteknight
147 https://en.wikibooks.org/wiki/User:WikiBayer
148 https://en.wikibooks.org/wiki/User:Wknight8111
149 https://en.wikibooks.org/w/index.php%3ftitle=User:XMollioTKs&action=edit&redlink=1
150 https://en.wikibooks.org/wiki/User:Xania
151 https://en.wikibooks.org/w/index.php%3ftitle=User:Xris&action=edit&redlink=1
152 https://en.wikibooks.org/wiki/User:YMS
153 https://en.wikibooks.org/w/index.php%3ftitle=User:Yeus&action=edit&redlink=1
154 https://en.wikibooks.org/w/index.php%3ftitle=User:Z35~enwikibooks&action=edit&
redlink=1
155 https://en.wikibooks.org/w/index.php%3ftitle=User:Zhaopatrick&action=edit&redlink=1
156 https://en.wikibooks.org/w/index.php%3ftitle=User:Zoomzoom~enwikibooks&action=edit&
redlink=1
157 https://en.wikibooks.org/wiki/User:~riley

List of Figures

- GFDL: Gnu Free Documentation License. `http://www.gnu.org/licenses/fdl.html`

- cc-by-sa-3.0: Creative Commons Attribution ShareAlike 3.0 License. `http://creativecommons.org/licenses/by-sa/3.0/`

- cc-by-sa-2.5: Creative Commons Attribution ShareAlike 2.5 License. `http://creativecommons.org/licenses/by-sa/2.5/`

- cc-by-sa-2.0: Creative Commons Attribution ShareAlike 2.0 License. `http://creativecommons.org/licenses/by-sa/2.0/`

- cc-by-sa-1.0: Creative Commons Attribution ShareAlike 1.0 License. `http://creativecommons.org/licenses/by-sa/1.0/`

- cc-by-2.0: Creative Commons Attribution 2.0 License. `http://creativecommons.org/licenses/by/2.0/`

- cc-by-2.0: Creative Commons Attribution 2.0 License. `http://creativecommons.org/licenses/by/2.0/deed.en`

- cc-by-2.5: Creative Commons Attribution 2.5 License. `http://creativecommons.org/licenses/by/2.5/deed.en`

- cc-by-3.0: Creative Commons Attribution 3.0 License. `http://creativecommons.org/licenses/by/3.0/deed.en`

- GPL: GNU General Public License. `http://www.gnu.org/licenses/gpl-2.0.txt`

- LGPL: GNU Lesser General Public License. `http://www.gnu.org/licenses/lgpl.html`

- PD: This image is in the public domain.

- ATTR: The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted.

- EURO: This is the common (reverse) face of a euro coin. The copyright on the design of the common face of the euro coins belongs to the European Commission. Authorised is reproduction in a format without relief (drawings, paintings, films) provided they are not detrimental to the image of the euro.

- LFK: Lizenz Freie Kunst. `http://artlibre.org/licence/lal/de`

- CFR: Copyright free use.

- EPL: Eclipse Public License. `http://www.eclipse.org/org/documents/epl-v10.php`

Copies of the GPL, the LGPL as well as a GFDL are included in chapter Licenses[158]. Please note that images in the public domain do not require attribution. You may click on the image numbers in the following table to open the webpage of the images in your webbrower.

---

158 Chapter 48 on page 355

1	BrightRaven, Gabor commonswiki, Jacek Halicki, Jarekt-Bot, Luestling commonswiki, Olivier, Umherirrender	
2	Jules Boilly	
3	Whiteknight[159] at English Wikibooks[160]	
4	Whiteknight[161] at English Wikibooks[162]	
5	Whiteknight[163] at English Wikibooks[164]	
6	Whiteknight[165] at English Wikibooks[166]	
7	Whiteknight[167] at English Wikibooks[168]	
8	Whiteknight[169] at English Wikibooks[170]	
9	Whiteknight[171] at English Wikibooks[172]	
10	Whiteknight[173] at English Wikibooks[174]	
11	Whiteknight[175] at English Wikibooks[176]	
12	Whiteknight[177] at English Wikibooks[178]	
13	Whiteknight[179] at English Wikibooks[180]	
14	Whiteknight[181] at English Wikibooks[182]	
15	Inductiveload[183], Inductiveload[184]	
16	Spradlig[185]	
17	Ea91b3dd, Emijrpbot, EugeneZelenko, Glenn, Hazard-Bot, Ilmari Karonen, JarektBot, MGA73bot2, Severino666 commonswiki	
18	Whiteknight[186] at English Wikibooks[187]	
19	Whiteknight[188] at English Wikibooks[189]	

---

159 https://en.wikibooks.org/wiki/User:Whiteknight
160 https://en.wikibooks.org/wiki/
161 https://en.wikibooks.org/wiki/User:Whiteknight
162 https://en.wikibooks.org/wiki/
163 https://en.wikibooks.org/wiki/User:Whiteknight
164 https://en.wikibooks.org/wiki/
165 https://en.wikibooks.org/wiki/User:Whiteknight
166 https://en.wikibooks.org/wiki/
167 https://en.wikibooks.org/wiki/User:Whiteknight
168 https://en.wikibooks.org/wiki/
169 https://en.wikibooks.org/wiki/User:Whiteknight
170 https://en.wikibooks.org/wiki/
171 https://en.wikibooks.org/wiki/User:Whiteknight
172 https://en.wikibooks.org/wiki/
173 https://en.wikibooks.org/wiki/User:Whiteknight
174 https://en.wikibooks.org/wiki/
175 https://en.wikibooks.org/wiki/User:Whiteknight
176 https://en.wikibooks.org/wiki/
177 https://en.wikibooks.org/wiki/User:Whiteknight
178 https://en.wikibooks.org/wiki/
179 https://en.wikibooks.org/wiki/User:Whiteknight
180 https://en.wikibooks.org/wiki/
181 https://en.wikibooks.org/wiki/User:Whiteknight
182 https://en.wikibooks.org/wiki/
183 http://commons.wikimedia.org/wiki/User:Inductiveload
184 https://commons.wikimedia.org/wiki/User:Inductiveload
185 https://en.wikibooks.org/wiki/en:User:Spradlig
186 https://en.wikibooks.org/wiki/User:Whiteknight
187 https://en.wikibooks.org/wiki/
188 https://en.wikibooks.org/wiki/User:Whiteknight
189 https://en.wikibooks.org/wiki/

20	Whiteknight[190] at English Wikibooks[191]	
21	Whiteknight[192] at English Wikibooks[193]	
22	Whiteknight[194] at English Wikibooks[195]	
23	Whiteknight[196] at English Wikibooks[197]	
24	No machine-readable author provided. Rbj[198] assumed (based on copyright claims)., No machine-readable author provided. Rbj[199] assumed (based on copyright claims).	
25	image source obtained from en:User:Petr.adamek[200] (with permission) and previously saved as PD in PNG format. touched up a little and converted to SVG by en:User:Rbj[201]	
26	No machine-readable author provided. Rbj[202] assumed (based on copyright claims)., No machine-readable author provided. Rbj[203] assumed (based on copyright claims).	
27	No machine-readable author provided. Rbj[204] assumed (based on copyright claims)., No machine-readable author provided. Rbj[205] assumed (based on copyright claims).	
28	No machine-readable author provided. Rbj[206] assumed (based on copyright claims)., No machine-readable author provided. Rbj[207] assumed (based on copyright claims).	
29	No machine-readable author provided. Rbj[208] assumed (based on copyright claims)., No machine-readable author provided. Rbj[209] assumed (based on copyright claims).	
30	Whiteknight[210] at English Wikibooks[211]	
31	Whiteknight[212] at English Wikibooks[213]	
32	Inductiveload[214], Inductiveload[215]	

---

190 https://en.wikibooks.org/wiki/User:Whiteknight
191 https://en.wikibooks.org/wiki/
192 https://en.wikibooks.org/wiki/User:Whiteknight
193 https://en.wikibooks.org/wiki/
194 https://en.wikibooks.org/wiki/User:Whiteknight
195 https://en.wikibooks.org/wiki/
196 https://en.wikibooks.org/wiki/User:Whiteknight
197 https://en.wikibooks.org/wiki/
198 http://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
199 https://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
200 https://en.wikipedia.org/wiki/User:Petr.adamek
201 https://en.wikipedia.org/wiki/User:Rbj
202 http://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
203 https://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
204 http://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
205 https://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
206 http://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
207 https://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
208 http://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
209 https://commons.wikimedia.org/w/index.php?title=User:Rbj&action=edit&redlink=1
210 https://en.wikibooks.org/wiki/User:Whiteknight
211 https://en.wikibooks.org/wiki/
212 https://en.wikibooks.org/wiki/User:Whiteknight
213 https://en.wikibooks.org/wiki/
214 http://commons.wikimedia.org/wiki/User:Inductiveload
215 https://commons.wikimedia.org/wiki/User:Inductiveload

33	Inductiveload[216], Inductiveload[217]	
34	Inductiveload[218], Inductiveload[219]	
35	Inductiveload[220], Inductiveload[221]	
36	Inductiveload[222], Inductiveload[223]	
37	Inductiveload[224], Inductiveload[225]	
38	Inductiveload[226], Inductiveload[227]	
39	Inductiveload[228], Inductiveload[229]	
40	Inductiveload[230], Inductiveload[231]	
41	Inductiveload[232], Inductiveload[233]	
42	Inductiveload[234], Inductiveload[235]	
43	Inductiveload[236], Inductiveload[237]	
44	Inductiveload[238], Inductiveload[239]	
45	Inductiveload[240], Inductiveload[241]	
46	Inductiveload[242], Inductiveload[243]	
47	Inductiveload[244], Inductiveload[245]	
48	Inductiveload[246], Inductiveload[247]	
49	Inductiveload[248], Inductiveload[249]	

---

216 http://commons.wikimedia.org/wiki/User:Inductiveload
217 https://commons.wikimedia.org/wiki/User:Inductiveload
218 http://commons.wikimedia.org/wiki/User:Inductiveload
219 https://commons.wikimedia.org/wiki/User:Inductiveload
220 http://commons.wikimedia.org/wiki/User:Inductiveload
221 https://commons.wikimedia.org/wiki/User:Inductiveload
222 http://commons.wikimedia.org/wiki/User:Inductiveload
223 https://commons.wikimedia.org/wiki/User:Inductiveload
224 http://commons.wikimedia.org/wiki/User:Inductiveload
225 https://commons.wikimedia.org/wiki/User:Inductiveload
226 http://commons.wikimedia.org/wiki/User:Inductiveload
227 https://commons.wikimedia.org/wiki/User:Inductiveload
228 http://commons.wikimedia.org/wiki/User:Inductiveload
229 https://commons.wikimedia.org/wiki/User:Inductiveload
230 http://commons.wikimedia.org/wiki/User:Inductiveload
231 https://commons.wikimedia.org/wiki/User:Inductiveload
232 http://commons.wikimedia.org/wiki/User:Inductiveload
233 https://commons.wikimedia.org/wiki/User:Inductiveload
234 http://commons.wikimedia.org/wiki/User:Inductiveload
235 https://commons.wikimedia.org/wiki/User:Inductiveload
236 http://commons.wikimedia.org/wiki/User:Inductiveload
237 https://commons.wikimedia.org/wiki/User:Inductiveload
238 http://commons.wikimedia.org/wiki/User:Inductiveload
239 https://commons.wikimedia.org/wiki/User:Inductiveload
240 http://commons.wikimedia.org/wiki/User:Inductiveload
241 https://commons.wikimedia.org/wiki/User:Inductiveload
242 http://commons.wikimedia.org/wiki/User:Inductiveload
243 https://commons.wikimedia.org/wiki/User:Inductiveload
244 http://commons.wikimedia.org/wiki/User:Inductiveload
245 https://commons.wikimedia.org/wiki/User:Inductiveload
246 http://commons.wikimedia.org/wiki/User:Inductiveload
247 https://commons.wikimedia.org/wiki/User:Inductiveload
248 http://commons.wikimedia.org/wiki/User:Inductiveload
249 https://commons.wikimedia.org/wiki/User:Inductiveload

250 http://commons.wikimedia.org/wiki/User:Inductiveload
251 https://commons.wikimedia.org/wiki/User:Inductiveload
252 http://commons.wikimedia.org/wiki/User:Inductiveload
253 https://commons.wikimedia.org/wiki/User:Inductiveload
254 http://commons.wikimedia.org/wiki/User:Inductiveload
255 https://commons.wikimedia.org/wiki/User:Inductiveload
256 http://commons.wikimedia.org/wiki/User:Inductiveload
257 https://commons.wikimedia.org/wiki/User:Inductiveload
258 http://commons.wikimedia.org/wiki/User:Inductiveload
259 https://commons.wikimedia.org/wiki/User:Inductiveload
260 http://commons.wikimedia.org/wiki/User:Inductiveload
261 https://commons.wikimedia.org/wiki/User:Inductiveload
262 http://commons.wikimedia.org/wiki/User:Inductiveload
263 https://commons.wikimedia.org/wiki/User:Inductiveload
264 http://commons.wikimedia.org/wiki/User:Inductiveload
265 https://commons.wikimedia.org/wiki/User:Inductiveload
266 http://commons.wikimedia.org/wiki/User:Inductiveload
267 https://commons.wikimedia.org/wiki/User:Inductiveload
268 http://commons.wikimedia.org/wiki/User:Inductiveload
269 https://commons.wikimedia.org/wiki/User:Inductiveload
270 http://commons.wikimedia.org/wiki/User:Inductiveload
271 https://commons.wikimedia.org/wiki/User:Inductiveload
272 http://commons.wikimedia.org/wiki/User:Inductiveload
273 https://commons.wikimedia.org/wiki/User:Inductiveload
274 http://commons.wikimedia.org/wiki/User:Inductiveload
275 https://commons.wikimedia.org/wiki/User:Inductiveload
276 https://en.wikipedia.org/wiki/User:Ap
277 https://en.wikibooks.org/wiki/User:Whiteknight
278 https://en.wikibooks.org/wiki/
279 https://en.wikibooks.org/wiki/User:Whiteknight
280 https://en.wikibooks.org/wiki/
281 https://en.wikibooks.org/wiki/User:Nithinvgeorge
282 https://en.wikibooks.org/wiki/

68	Pierre5018[283], Pierre5018[284]	
69	Pierre5018[285], Pierre5018[286]	
70	Pierre5018[287], Pierre5018[288]	
71	Pierre5018[289], Pierre5018[290]	
72	Pierre5018[291], Pierre5018[292]	
73	Pierre5018[293], Pierre5018[294]	
74	Pierre5018[295], Pierre5018[296]	
75	Pierre5018[297], Pierre5018[298]	
76	Pierre5018[299], Pierre5018[300]	
77	Pierre5018[301], Pierre5018[302]	
78	Pierre5018[303], Pierre5018[304]	
79	Pierre5018[305], Pierre5018[306]	
80	Pierre5018[307], Pierre5018[308]	
81	Pierre5018[309], Pierre5018[310]	
82	Pierre5018[311], Pierre5018[312]	
83	Pierre5018[313], Pierre5018[314]	
84	Pierre5018[315], Pierre5018[316]	

---

283 http://commons.wikimedia.org/wiki/User:Pierre5018
284 https://commons.wikimedia.org/wiki/User:Pierre5018
285 http://commons.wikimedia.org/wiki/User:Pierre5018
286 https://commons.wikimedia.org/wiki/User:Pierre5018
287 http://commons.wikimedia.org/wiki/User:Pierre5018
288 https://commons.wikimedia.org/wiki/User:Pierre5018
289 http://commons.wikimedia.org/wiki/User:Pierre5018
290 https://commons.wikimedia.org/wiki/User:Pierre5018
291 http://commons.wikimedia.org/wiki/User:Pierre5018
292 https://commons.wikimedia.org/wiki/User:Pierre5018
293 http://commons.wikimedia.org/wiki/User:Pierre5018
294 https://commons.wikimedia.org/wiki/User:Pierre5018
295 http://commons.wikimedia.org/wiki/User:Pierre5018
296 https://commons.wikimedia.org/wiki/User:Pierre5018
297 http://commons.wikimedia.org/wiki/User:Pierre5018
298 https://commons.wikimedia.org/wiki/User:Pierre5018
299 http://commons.wikimedia.org/wiki/User:Pierre5018
300 https://commons.wikimedia.org/wiki/User:Pierre5018
301 http://commons.wikimedia.org/wiki/User:Pierre5018
302 https://commons.wikimedia.org/wiki/User:Pierre5018
303 http://commons.wikimedia.org/wiki/User:Pierre5018
304 https://commons.wikimedia.org/wiki/User:Pierre5018
305 http://commons.wikimedia.org/wiki/User:Pierre5018
306 https://commons.wikimedia.org/wiki/User:Pierre5018
307 http://commons.wikimedia.org/wiki/User:Pierre5018
308 https://commons.wikimedia.org/wiki/User:Pierre5018
309 http://commons.wikimedia.org/wiki/User:Pierre5018
310 https://commons.wikimedia.org/wiki/User:Pierre5018
311 http://commons.wikimedia.org/wiki/User:Pierre5018
312 https://commons.wikimedia.org/wiki/User:Pierre5018
313 http://commons.wikimedia.org/wiki/User:Pierre5018
314 https://commons.wikimedia.org/wiki/User:Pierre5018
315 http://commons.wikimedia.org/wiki/User:Pierre5018
316 https://commons.wikimedia.org/wiki/User:Pierre5018

85	Pierre5018[317], Pierre5018[318]	
86	Pierre5018[319], Pierre5018[320]	
87	Pierre5018[321], Pierre5018[322]	
88	Pierre5018[323], Pierre5018[324]	
89	Pierre5018[325], Pierre5018[326]	
90	Pierre5018[327], Pierre5018[328]	
91	Pierre5018[329], Pierre5018[330]	
92	Pierre5018[331], Pierre5018[332]	
93	Pierre5018[333], Pierre5018[334]	
94	Pierre5018[335], Pierre5018[336]	
95	Pierre5018[337], Pierre5018[338]	
96	Pierre5018[339], Pierre5018[340]	
97	Pierre5018[341], Pierre5018[342]	
98	Pierre5018[343], Pierre5018[344]	
99	Pierre5018[345], Pierre5018[346]	
100	Pierre5018[347], Pierre5018[348]	
101	Pierre5018[349], Pierre5018[350]	

317 http://commons.wikimedia.org/wiki/User:Pierre5018
318 https://commons.wikimedia.org/wiki/User:Pierre5018
319 http://commons.wikimedia.org/wiki/User:Pierre5018
320 https://commons.wikimedia.org/wiki/User:Pierre5018
321 http://commons.wikimedia.org/wiki/User:Pierre5018
322 https://commons.wikimedia.org/wiki/User:Pierre5018
323 http://commons.wikimedia.org/wiki/User:Pierre5018
324 https://commons.wikimedia.org/wiki/User:Pierre5018
325 http://commons.wikimedia.org/wiki/User:Pierre5018
326 https://commons.wikimedia.org/wiki/User:Pierre5018
327 http://commons.wikimedia.org/wiki/User:Pierre5018
328 https://commons.wikimedia.org/wiki/User:Pierre5018
329 http://commons.wikimedia.org/wiki/User:Pierre5018
330 https://commons.wikimedia.org/wiki/User:Pierre5018
331 http://commons.wikimedia.org/wiki/User:Pierre5018
332 https://commons.wikimedia.org/wiki/User:Pierre5018
333 http://commons.wikimedia.org/wiki/User:Pierre5018
334 https://commons.wikimedia.org/wiki/User:Pierre5018
335 http://commons.wikimedia.org/wiki/User:Pierre5018
336 https://commons.wikimedia.org/wiki/User:Pierre5018
337 http://commons.wikimedia.org/wiki/User:Pierre5018
338 https://commons.wikimedia.org/wiki/User:Pierre5018
339 http://commons.wikimedia.org/wiki/User:Pierre5018
340 https://commons.wikimedia.org/wiki/User:Pierre5018
341 http://commons.wikimedia.org/wiki/User:Pierre5018
342 https://commons.wikimedia.org/wiki/User:Pierre5018
343 http://commons.wikimedia.org/wiki/User:Pierre5018
344 https://commons.wikimedia.org/wiki/User:Pierre5018
345 http://commons.wikimedia.org/wiki/User:Pierre5018
346 https://commons.wikimedia.org/wiki/User:Pierre5018
347 http://commons.wikimedia.org/wiki/User:Pierre5018
348 https://commons.wikimedia.org/wiki/User:Pierre5018
349 http://commons.wikimedia.org/wiki/User:Pierre5018
350 https://commons.wikimedia.org/wiki/User:Pierre5018

102	Pierre5018[351], Pierre5018[352]	
103	Pierre5018[353], Pierre5018[354]	
104	Pierre5018[355], Pierre5018[356]	
105	Pierre5018[357], Pierre5018[358]	
106	Constant314[359], Constant314[360]	
107	User:Netnet[361], User:Netnet[362]	
108	User:Netnet[363], User:Netnet[364]	
109	User:Netnet[365], User:Netnet[366]	
110	User:Netnet[367], User:Netnet[368]	
111	User:Netnet[369], User:Netnet[370]	
112	The original uploader was Whiteknight[371] at English Wikibooks[372].	
113	The original uploader was Whiteknight[373] at English Wikibooks[374].	
114	The original uploader was Whiteknight[375] at English Wikibooks[376].	
115	Warlock31415[377], Warlock31415[378]	
116	User:Netnet[379], User:Netnet[380]	
117	User:Netnet[381], User:Netnet[382]	

351 http://commons.wikimedia.org/wiki/User:Pierre5018
352 https://commons.wikimedia.org/wiki/User:Pierre5018
353 http://commons.wikimedia.org/wiki/User:Pierre5018
354 https://commons.wikimedia.org/wiki/User:Pierre5018
355 http://commons.wikimedia.org/wiki/User:Pierre5018
356 https://commons.wikimedia.org/wiki/User:Pierre5018
357 http://commons.wikimedia.org/wiki/User:Pierre5018
358 https://commons.wikimedia.org/wiki/User:Pierre5018
359 http://commons.wikimedia.org/wiki/User:Constant314
360 https://commons.wikimedia.org/wiki/User:Constant314
361 http://commons.wikimedia.org/wiki/User:Netnet
362 https://commons.wikimedia.org/wiki/User:Netnet
363 http://commons.wikimedia.org/wiki/User:Netnet
364 https://commons.wikimedia.org/wiki/User:Netnet
365 http://commons.wikimedia.org/wiki/User:Netnet
366 https://commons.wikimedia.org/wiki/User:Netnet
367 http://commons.wikimedia.org/wiki/User:Netnet
368 https://commons.wikimedia.org/wiki/User:Netnet
369 http://commons.wikimedia.org/wiki/User:Netnet
370 https://commons.wikimedia.org/wiki/User:Netnet
371 https://en.wikibooks.org/wiki/User:Whiteknight
372 https://en.wikibooks.org/wiki/
373 https://en.wikibooks.org/wiki/User:Whiteknight
374 https://en.wikibooks.org/wiki/
375 https://en.wikibooks.org/wiki/User:Whiteknight
376 https://en.wikibooks.org/wiki/
377 http://commons.wikimedia.org/w/index.php?title=User:Warlock31415&action=edit&redlink=1
378 https://commons.wikimedia.org/w/index.php?title=User:Warlock31415&action=edit&redlink=1
379 http://commons.wikimedia.org/wiki/User:Netnet
380 https://commons.wikimedia.org/wiki/User:Netnet
381 http://commons.wikimedia.org/wiki/User:Netnet
382 https://commons.wikimedia.org/wiki/User:Netnet

383 http://commons.wikimedia.org/wiki/User:Netnet
384 https://commons.wikimedia.org/wiki/User:Netnet
385 https://sk.wikipedia.org/wiki/user:robo
386 https://en.wikibooks.org/wiki/User:Whiteknight
387 https://en.wikibooks.org/wiki/
388 https://en.wikibooks.org/wiki/User:Whiteknight
389 https://en.wikibooks.org/wiki/
390 https://en.wikibooks.org/wiki/User:Whiteknight
391 https://en.wikibooks.org/wiki/

48 Licenses

48.1 GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date. * b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices". * c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. * d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. * b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge. * c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. * d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a

different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. * e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or * b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or * c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or * d) Limiting the use for publicity purposes of names of licensors or authors of the material; or * e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or * f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way. 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates

your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy

both those terms and this License would be to refrain entirely from conveying the Program. 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

48.2 GNU Free Documentation License

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.

The "publisher" means any person or entity that distributes copies of the Document to the public.

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 under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document. 11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

48.3 GNU Lesser General Public License

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below. 0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work. 1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL. 2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

* a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or * b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

* a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

* a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the Combined Work with a copy of the GNU GPL and this license document. * c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document. * d) Do one of the following: o 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source. o 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version. * e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

* a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License. * b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.